Ruprecht-Karls University of Heidelberg
Institute of Computer Engineering

Holger Fröning
Bernhard Klein

Advanced Parallel Computing
Summer Term 2019

Exercise 5

- **Return electronically until Tuesday, May 28, 9:00 am**
- **Include name on the top sheet.**
- **A maximum of two students is allowed to work jointly on the exercises.**

5.1 Reading

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

- Shucai Xiao (Virginia Tech, US); Wu-chun Feng (Virginia Tech, US), Inter-Block GPU Communication via Fast Barrier Synchronization, IPDPS 2010.
- Torsten Hoefler, Torsten Mehlan, Frank Mietke and Wolfgang Rehm, Fast Barrier Synchronization for InfiniBand, CAC 2006, co-located with IPDPS 2006.

(25 points)

5.2 Barrier implementation and performance analysis

In this exercise, the goal is to develop a scalable barrier with minimal overhead. Refer to these references:

- John M. Mellor-Crummey, Michael L. Scott. Algorithms for Scalable Synchronization on Shared-Memory Multiprocessors. ACM Trans. on Computer Systems, 1991.
- Holger Fröning et al, Highly Scalable Barriers for Future High-Performance Computing Clusters, HiPC 2011.

First, implement the barrier using a central counter that is incremented using atomic operations. Besides the *PTHREAD* barrier, this serves as a reference.

Then, implement a barrier that is optimized according to the two references above. In particular, choose one from the alternatives including *tournament*, *dissemination* and *tree structures*, or combinations thereof.

Write a test program in which each thread performs a certain number of barrier operations. Ensure that your barriers are working correctly, for instance by arbitrarily slowing down a single thread. Each thread should successfully pass the same number of barriers.

Ruprecht-Karls University of Heidelberg
Institute of Computer Engineering

Holger Fröning
Bernhard Klein

Conduct a performance assessment by comparing average barrier latency against the *PTHREAD* barrier implementation. Increase the number of threads participating in the barrier operation according to the following table. Report average barrier latency in the table and in a graphical representation. Perform a sufficient large number of barrier operations in order to obtain stable results. Derive average barrier latency by dividing the overall execution time by the number of barrier operations performed.

	1. PTHREAD BARRIER	2. COUNTER BARRIER	3. OPTIMIZED BARRIER 1	4. OPTIMIZED BARRIER 2	5. OPTIMIZED BARRIER 3
Thread Count	Average barrier latency	Average barrier latency	Average barrier latency	Average barrier latency	Average barrier latency
1					
2					
4					
8					
12					
16					
24					
32					
40					
48					

(60 points)

Total: 85 points