Ruprecht-Karls University of Heidelberg
Institute of Computer Engineering

Holger Fröning
Dominik Zosgornik

Advanced Parallel Computing
Summer term 2017

Exercise 4

- **Return electronically until Tuesday, May 16, 14:00**
- **Include name on the top sheet.**
- **A maximum of two students is allowed to work jointly on the exercises.**

4.1 Reading

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

- Alain Kägi, Doug Burger, and James R. Goodman. 1997. Efficient synchronization: let them eat QOLB. In Proceedings of the 24th annual international symposium on Computer architecture (ISCA '97). ACM, New York, NY, USA, 170-180. DOI=http://dx.doi.org/10.1145/264107.264166
- Jose L. Abellán, Juan Fernández, and Manuel E. Acacio. 2011. GLocks: Efficient Support for Highly-Contended Locks in Many-Core CMPs. In Proceedings of the 2011 IEEE International Parallel & Distributed Processing Symposium (IPDPS '11). IEEE Computer Society, Washington, DC, USA, 893-905. DOI=http://dx.doi.org/10.1109/IPDPS.2011.87

(25 points)

4.2 List-based Queue Locks

Start with the program from exercise 3.3. Now, implement the lock primitive using the MCS lock algorithm from Mellor-Crummey/Scott 1991.

- John M. Mellor-Crummey, Michael L. Scott. Algorithms for Scalable Synchronization on Shared-Memory Multiprocessors. ACM Trans. on Computer Systems, 1991

Develop your programs and perform initial testing on one of the `creek` nodes. Validate the correctness of these two new programs with the same methodology as in exercise 3.3. i.e., for a varying C and N, ensure that after execution the counter matches C, i.e. there are no race conditions anymore.

(35 points)

4.3 Lock performance analysis

Now, measure the overall execution time using suitable functions (e.g. `clock_gettime` or `gettimeofday`). Report the overall execution time and the derived number of updates per second for a varying number of threads (1-48) and sufficiently large number of updates (providing stable results). For this experiment, use the computer `moore` (48 cores in total). As `moore` is often heavily used, please ensure that you only use it for performance experiments.

	PTHREAD MUTEX		ATOMIC INCREMENT		LOCK_RMW		QUEUE LOCK	
Thread Count	Execution Time	Updates per second	Execution Time	Updates per second	Execution Time	Updates per second	Execution Time	Updates per second
1								
2								
4								
8								
12								
16								
24								
32								
40								
48								

Re-use the data for experiments 1.-3. Run the experiment for the MCS Lock and report results. Include a graphical representation here (varying number of threads on x-axis with updates per second on y-axis).

(15 points)

Total: 75 points