Department of Computer Science, ETH Zürich

Supporting Parallelism in Operating Systems & Programming Languages

Spring 2012

Exercise 3: Synchronization

1 Multi-threaded task scheduling

The goal of this exercise is to extend the single-threaded task scheduling system from the previous exercise to utilize multiple processors. Specifically, your system should create a number of threads, each of which maintains its own task queue. Use work stealing to balance the workload.

For your intermediate report:

- Give an overview of your design
- Provide a synchronization scheme for your design. Structure this discussion based on the data-structures you are using and how you protect them from concurrent accesses.
- Provide an API for your system. If you are using the same API you specified in the previous exercise, duplicate it here for convenience. If you made changes to the API, discuss your reasoning.
- Briefly discuss at a high-level (e.g., with a few bullet points) how would you implement each API operation.

For your final report:

- Implement your design
- Provide an updated discussion of your implementation as in your intermediate report. Mention changes, if any.
- Use a simple program (e.g., the recursive Fibonacci number computation) to evaluate the performance of your system. Present and discuss the results. What is the part of your system that has the biggest overhead?

2 Locking strategy

Adaptive mutexes are mutexes that combine a spinlock with an ordinary mutex: instead of blocking immediately when the critical section is locked, adaptive mutexes spin for some time and then block.

For your final report:

- Describe a situation where a normal mutex would be preferable over an adaptive mutex and justify your answer.
- **For bonus points:** implement the situation you described and verify that this is indeed the case.

3 Transactional Memory

For your final report:

- Describe the characteristics of an application for which transactional memory synchronization is a good fit. Justify your answer.

4 Dates & Deliverables

02.11.2012: Intermediate report
16.11.2012: Final version of the code & final report