Concurrency and memory consistency Lab

J. Daniel García Sánchez (coordinator)
David Expósito Singh
Javier García Blas

Computer Architecture
ARCOS Group
Computer Science and Engineering Department
University Carlos III of Madrid

1. Objective

The main goal of this lab is to familiarize the student with lock free programming and to allow understanding the impact it may have in an application's performance. Drawbacks and advantages of atomics versus lock based techniques will be evaluated.

2. Description

This lab will evaluate different approaches to implement a bounded circular buffer. To evaluate thread safety two techniques will be used: lock based programming and lock-free programming. Both cases will be programmed through standard ISO/IEC 14882:2011 (C++11).

2.1. Provided material

To perform the lab a bounded circular buffer basic implementation is provided in file **seq_buffer.h**.

2.2. Performance evaluation

To complete performance evaluation, you may use some of the following methods:

- Measuring time through the C++ standards library (namespace **chrono**).
- Access to Linux kernel module `perf`.

3. Tasks

3.1. Initial study

Study provided source code and analyze how it works.
Include a section in your report with an detailed explanation of the data structure.

- Identify under which conditions are exceptions thrown.

- Which is the utility of data members **next_read_** and **next_write_**.
- Which is the maximum number of elements that can be stored?

Also analyze the main program and explain in detail how it works.

3.2. Locked concurrent execution

Write a *thread-safe* version of the data structure for the case of single reader and single writer. Use in the implementation **mutex** and condition variables. This version must replace exceptions by blocking waits.

Also write a version of the main program in which a thread acts as producer and a different thread acts as consumer.

3.3. Lock-free concurrent execution

Write a lock-free and *thread-safe* version of the data structure using atomics. This version must replace exceptions by busy waiting using a memory consistency model. Consider the use of sequential consistency versus more relaxed models.

Also write a version of the main program in which a thread acts as producer and a different thread acts as consumer.

3.4. Performance evaluation

Evaluate the three solutions in a computer with at least two real cores. Conduct the evaluations that you consider more appropriate to determine advantages and drawbacks from different approaches.