Advanced Parallel Programming Exercise 7

Fabian Czappa

TECHNISCHE UNIVERSITÄT DARMSTADT

Summer term 2025
10.07.2025

Please solve the following tasks by 10.07.2025. The results are not graded, but a solution is discussed on 10.07.2025.

Task 1: An extended game

In this task, we will extend a game to help unterstand the memory model in C++. We will firstly prepare a mental model, and then implement a demo.

A grumpy secretary (think of "Einwohnermeldeamt") has a queue of tasks they need to finish. As their employer, you are able to submit new tasks and query the 'latest' queued task. However, latest here means: They keep a post-it sticker with your name next to the 'latest' task they told you about, and for now, they have no incentive to tell you the latest one on the list. They also do the same for the clients. You want to keep your employee honest and at the end of the day, you get the list of all tasks and can check their progress.

1. Based on just the list they keep, in what way can you force the secretary to be honest and not change the list at the end of the day (think rearranging the tasks to move yours to the back)? I.e., what information can you ask the secretary to also provide when you give a task?
2. Based on just the list they keep, in what way can you force the secretary to be honest about the current latest task (think rearranging the tasks to move the 'latest' one to a nicer position)? I.e., what information can you ask the secretary to also provide when you query the 'latest' task?
3. Based on the defined interactions above, you are never able to add a direct follow-up task (think: You query the latest task, change it, push it again—but there are now multiple new tasks inbetween). How would you formulate a third interaction to remedy this?

Implement a simple program with an atomic integer and multiple threads (you can try 32). The task of the threads: For a given number (you can try 1000), read the value of the variable, increment the value by one, and write the value back. What do you observe?

Solution:

1. You ask the secretary for the current task and its position in the list.
2. Same.
3. You can say "update the latest task to also include the following ...".

```cpp
#include <atomic>
#include <iostream>
#include <thread>
#include <vector>

std::atomic<unsigned int> counter(0);

void increment_counter(const unsigned int iterations) {
  for (unsigned int i = 0; i < iterations; ++i) {
```

```
    const auto old_value = counter.load(std::memory_order_relaxed);
    const auto new_value = old_value + 1;
    counter.store(new_value, std::memory_order_relaxed);
  }
}

int main()
{
  const auto number_iterations = 1000U;
  const auto number_threads = 32U;

  auto threads = std::vector<std::thread>();
  threads.reserve(number_threads);

  for (unsigned int i = 0; i < number_threads; ++i) {
    threads.emplace_back(increment_counter, number_iterations);
  }

  for (auto& thread : threads) {
    thread.join();
  }

  std::cout << "Actual counter value: " << counter.load() << '\n';
  std::cout << "Expected counter value: " << (number_iterations * number_threads) << '\n';

  return 0;
}
```

Listing 1: Incrementing 32x1000. On my machine, this results in missing around 1000 increments.

Task 2: Atomic Minimum and Maximum Functions

In this task, we want to fill a hole in the C++ standard: There are no atomic minimum and maximum functions available.

Firstly, make yourself clear what kind such an atomic operation is. Is it read-after-write, write-after-read, write, read, or something else? If there are multiple suiting categories, what drawbacks do they have when compared to one another?

Implement each category. You can restrict yourself to one the two functions.

Solution:

The solution is taken from the proposal to the C++ standard **Grant, Kozicki, Northover; P0493R3**.

```
#pragma once

#include <algorithm>
#include <atomic>

template <typename T>
T atomic_fetch_max_read_modify_write(std::atomic<T>* pv,
  typename std::atomic<T>::value_type v,
  std::memory_order m) noexcept {
  auto t = pv->load(m);
  while (!pv->compare_exchange_weak(t, std::max(v, t), m, m))
    ;
  return t;
}

template <typename T>
T atomic_fetch_max_read_and_conditional_store(std::atomic<T>* pv,
  typename std::atomic<T>::value_type v,
  std::memory_order m) noexcept {
  auto t = pv->load(m);
  while (std::max(v, t) != t) {
    if (pv->compare_exchange_weak(t, v, m, m))
```

```
      break;
  }
  return t;
}

// notes:
// T::value_type is whatever it has been defined as in the type T. For instance if T is a vector<int> then
     value_type will be int.
// noexcept specifier: Specifies whether a function could throw exceptions.
```

Listing 2: Atomic minimum and maximum functions

An atomic minimum/ maximum can be a read-after-write instruction, as well as a read-and-conditional-store instruction. The later comes into play if the atomic value is already smaller respectively larger than the new value, and thus no update has to be performed.

Task 3: A curious execution order

Your classmate wrote the program below, which produces different results depending on the execution order.

Example: Your classmate found that the result **a: 4, b: 6** is produced by the following order of execution:
$\mathrm{a} = 1 \;\rightarrow\; \mathrm{b} = {+}{+}\mathrm{a} \;\rightarrow\; \mathrm{b} \mathrel{+}= 2 \;\rightarrow\; \mathrm{a} = 4 \;\rightarrow\; \mathrm{b} = 6$.

However, your classmate noticed that other executions orders do also occur. Please give a possible execution order of the program for each of the following two results, in the same notation as the example.

Hint: You may need to introduce a temporary variable to give the execution order.

- **a: 5, b: 7**
- **a: 4, b: 4**

```
std::atomic<int> a(0); std::atomic<int> b(0);

void workerA() {
        a = 1;
        b += 2;
}
void workerB() {
        b = ++a;
}

int main() {
        auto threadA = std::thread(workerA);
        auto threadB = std::thread(workerB);

        a = 4;
        b = 6;

        threadA.join();
        threadB.join();
        std::cout << "a: " << a << ", b: " << b;
}
```

Solution:

a: 5 b: 7:
$a = 1 \quad \rightarrow \quad a = 4 \quad \rightarrow \quad t = {+}{+}a \quad \rightarrow \quad b = 6 \quad \rightarrow \quad b = t \quad \rightarrow \quad b{+}= 2$
$a = 1 \quad \rightarrow \quad a = 4 \quad \rightarrow \quad b = 6 \quad \rightarrow \quad t = {+}{+}a \quad \rightarrow \quad b = t \quad \rightarrow \quad b{+}= 2$

a: 4 b: 4:
$a = 1 \quad \rightarrow \quad t = ++a \quad \rightarrow \quad a = 4 \quad \rightarrow \quad b = 6 \quad \rightarrow \quad b = t \quad \rightarrow \quad b+ = 2$