Spinlock vs. Mutex

It’s very interesting to compare the active waiting of a spinlock with the passive waiting of a mutex. Let's continue our discussing from the previous lesson and make a comparison between these two.

What will happen to the CPU load if the function `workOnResource` locks the spinlock for 2 seconds (lines 24 - 26)?

```cpp
// spinLockSleep.cpp

#include <iostream>
#include <atomic>
#include <thread>

class Spinlock{
  std::atomic_flag flag;
public:
  Spinlock(): flag(ATOMIC_FLAG_INIT){}

  void lock(){
    while( flag.test_and_set() );
  }

  void unlock(){
    flag.clear();
  }
};

Spinlock spin;

void workOnResource(){
  spin.lock();
  std::this_thread::sleep_for(std::chrono::milliseconds(2000));
  spin.unlock();
  std::cout << "Work done" << std::endl;
}


int main(){

  std::thread t(workOnResource);
  std::thread t2(workOnResource);

  t.join();
  t2.join();

}
```

According to the theory, one of the four cores of PC will be fully utilized, and that's exactly what happened as the load of one core reaches 100% on my PC. Each time, a different core performs busy waiting.

Now, I will use a mutex instead of a spinlock. Let's see what happens.

```
// mutex.cpp
#include <iostream>
#include <mutex>
#include <thread>

std::mutex mut;

void workOnResource(){
  mut.lock();
  std::this_thread::sleep_for(std::chrono::milliseconds(5000));
  mut.unlock();
  std::cout << "Work done" << std::endl;
}

int main(){

  std::thread t(workOnResource);
  std::thread t2(workOnResource);

  t.join();
  t2.join();

}
```

Although I executed the program several times, I did not observe a significant load on any of the cores.

In the next lesson, let's go one step further from the basic building block `std::atomic_flag` to the more advanced atomics: the class template `std::atomic`. The various partial and full specializations for bools, integral types, and pointers provide a more powerful interface than `std::atomic_flag`. The downside is that you do not have the guarantee that these specializations are lock-free.