

Falsesharing

optimisations

When one variable shares the same headline

• The CP will effectively block access to the memory until it has been used by a single core

Head/tail counters

Optimisations

False Sharing

- When one or more variables share the same cache line
- The CPU will effectively lock access to the memory until it has been used by a single core
- Head/tail counters

```

template<typename T>
class drow_queue_v5
{
public:
    drow_queue_v5 (size_t capacity_)
        : capacity (std::bit_ceil (capacity_))
    {}

    bool try_push (const T& v)
    {
        size_t current_tail = tail.load (std::memory_order_relaxed);
        size_t current_head = head.load (std::memory_order_acquire);

        size_t size = current_tail - current_head;

        if (size >= (capacity - 1)) // full
            return false;

        size_t index = current_tail & (capacity - 1);
        data[index] = v;
        tail.store (current_tail + 1, std::memory_order_release);

        return true;
    }

    bool try_pop (T& v)
    {
        size_t current_head = head.load (std::memory_order_relaxed);
        size_t current_tail = tail.load (std::memory_order_acquire);

        if (current_head == current_tail) // empty
            return false;

        size_t index = current_head & (capacity - 1);
        v = data[index];
        head.store (current_head + 1, std::memory_order_release);

        return true;
    }

private:
    size_t capacity = 0;
    std::vector<T> data { std::vector<T> (capacity) };
    alignas(hardware_destructive_interference_size) std::atomic<size_t> head { 0 };
    alignas(hardware_destructive_interference_size) std::atomic<size_t> tail { 0 };
};

```

