CS202-001 OS

Review Session 3 TA: Jinli Xiao

☑ 0. Record + Attendence

☑ 1. Process & Threads

☑ 2. What is Concurrency?

☑ 3. Concurrency Commandments

☑ 4. C++ Primer

☑ 5. Lab 3

☑ 6. Sequential Consistency

□ 7. Q & A

---

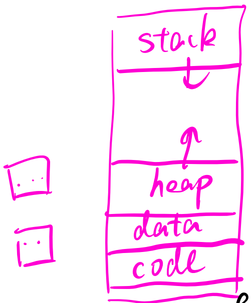
1. Process and Thread

Process:

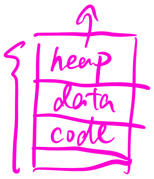
- An instance of a program.
- Has its own memory space & system resources.

Thread:

- A unit of execution within process
- Each process contains one or more threads
- If a thread crashes, it can cause the entire process to crash.



memory space of a single process

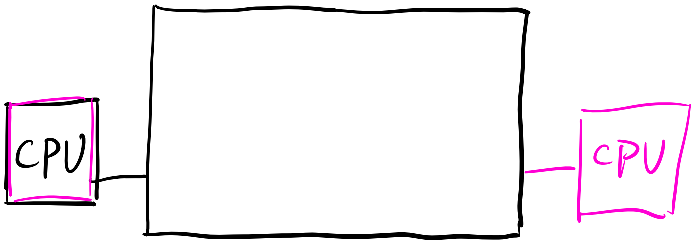


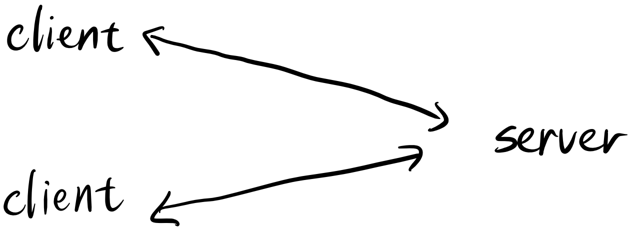
2. What is concurrency? Why?

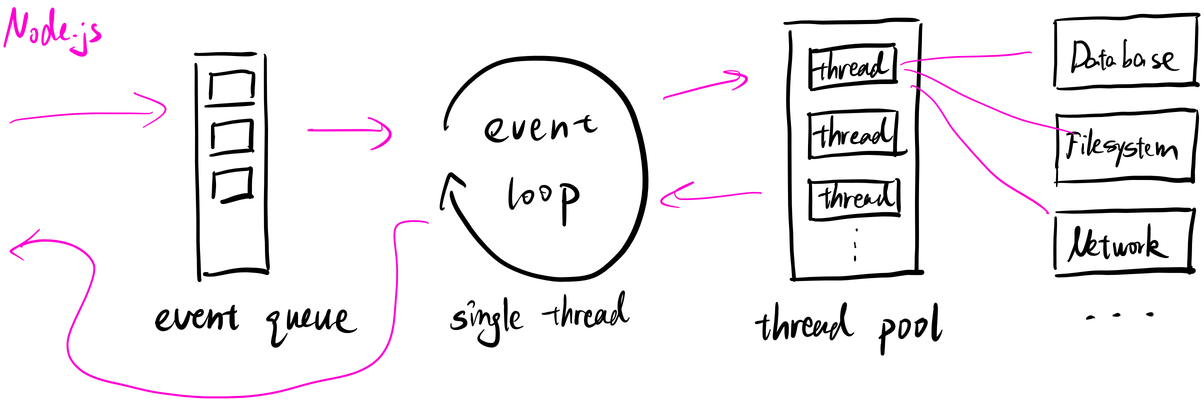
"It's when there are multiple threads?" ✗

"It's when things execute at the same time"

Eg.







But programs sometimes need to share resources. This could easily cause problems if multiple threads read/modify the same data concurrently.

3. Concurrency Commandments

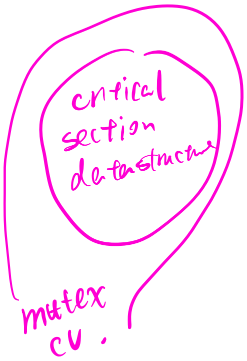
Rule 1. acquire/release lock at beginning/end of method

Rule 2. hold lock when doing conditional variable operations

Rule 3. prepare threads to wake anytime

```
while (cond)
    wait (&cv, &mutex);
```

signal ( ) v.s. broadcast ( )



4. C++ Primer

(a) Destructor

`TaskQueue::~TaskQueue()` opposite of construction

- free resources
- destroy mutex

(b) Freeing Dynamically Allocated Memory

`free(void *ptr)`   `delete`

`free(ptr_to_queue)`   `delete ptr_to_queue`

(c) Printing

`printf(...)`   `std::cout << ...`

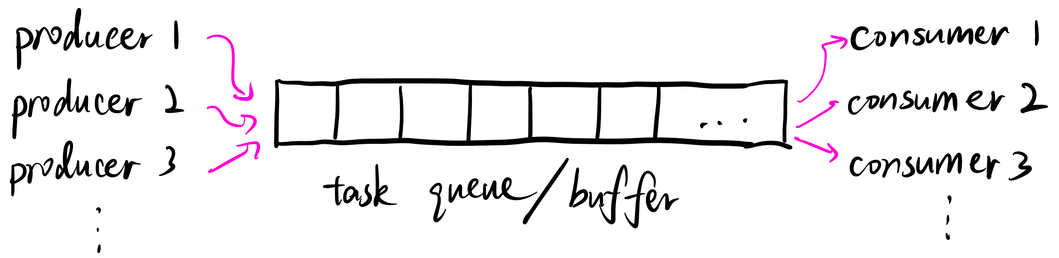
(d) Initializer List

```
class Item {
    public:
    bool valid;

    Item();
    ~Item();
}
```

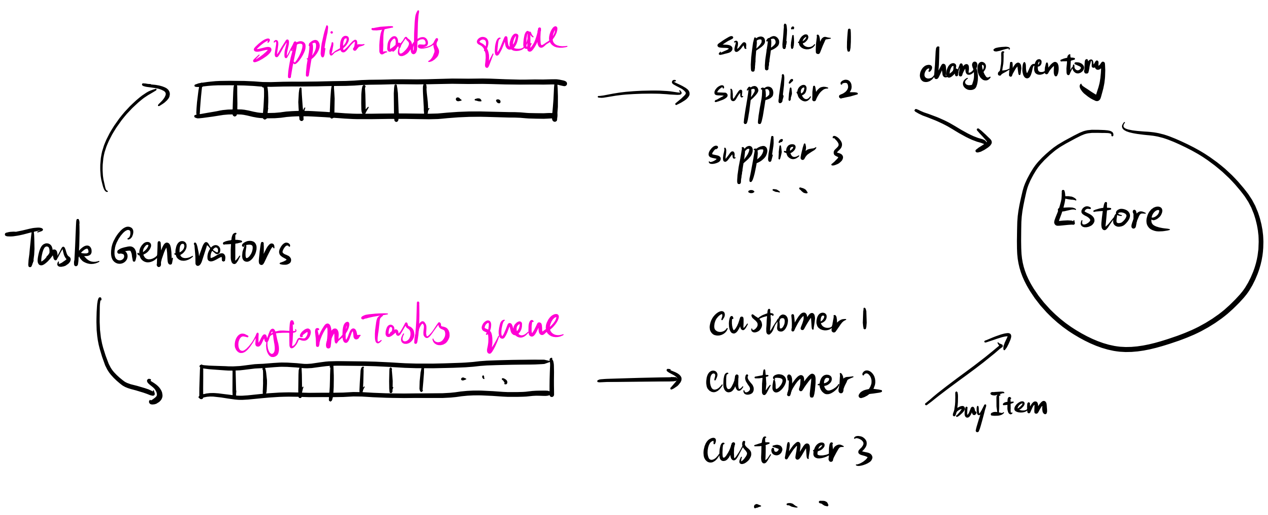
```
Item::Item() : valid(false) {}
Item::~Item() {}

Item::Item() {
    valid = false;
}
```

5. Lab 3 Overview

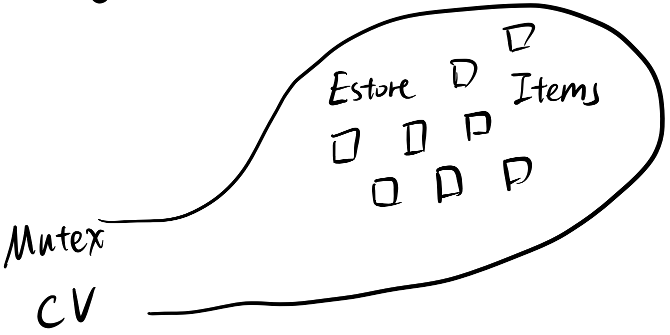
5.1 The general Producer - Consumer Architecture



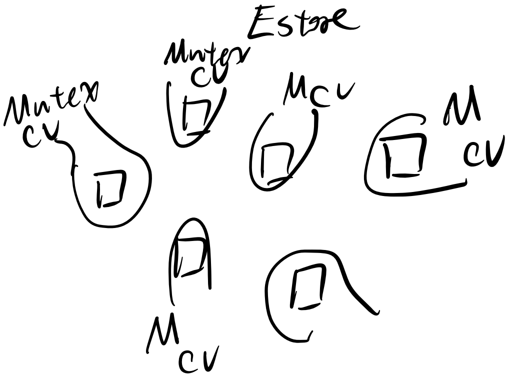
5.2 Lab 3 Architecture



Coarse grained:



Fine grained:



6.3 Notable Files

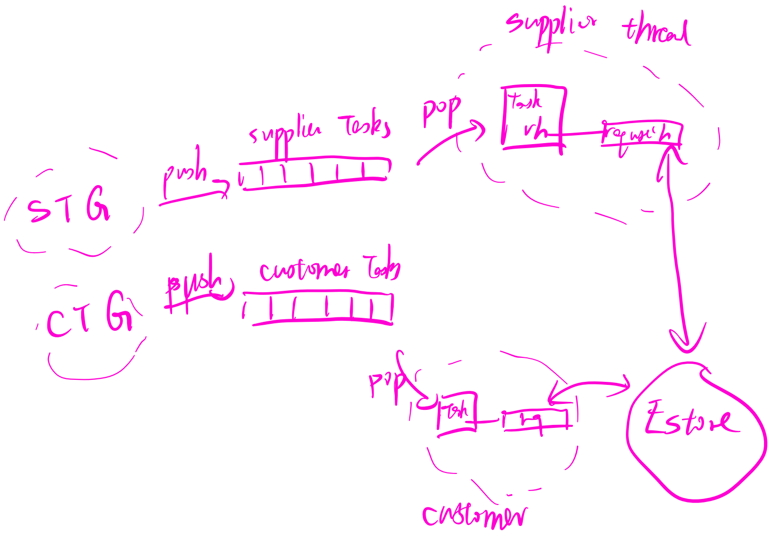
sthread.cpp

TaskQueue.cpp

estoresim.cpp

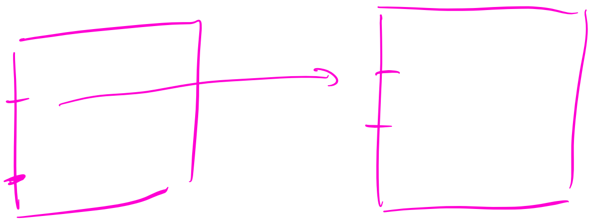
EStore.cpp

RequestHandler.cpp

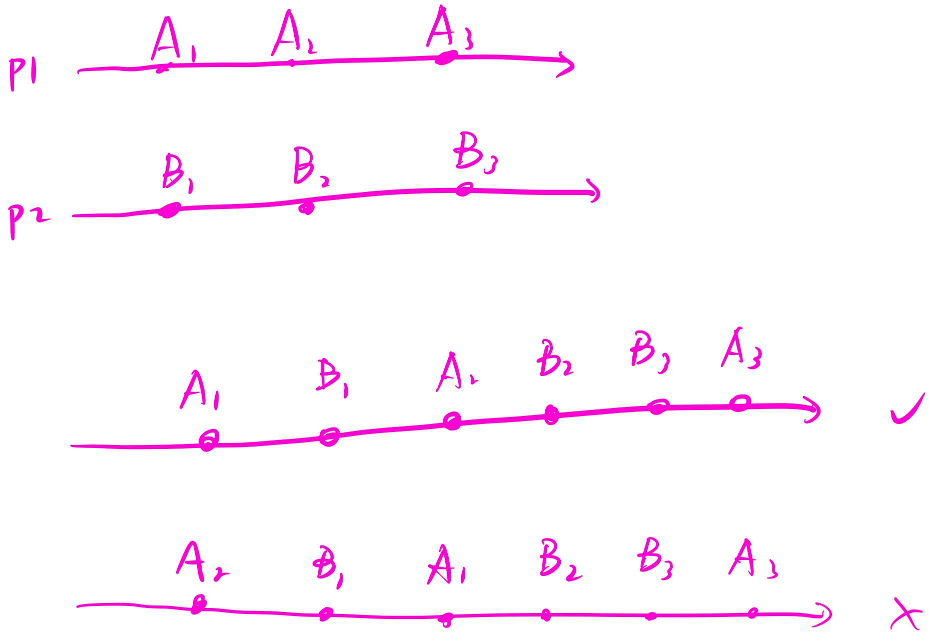


6.4 Common Pitfalls

- Forgot to free/destroy resources
- Forgot to release the mutex
- Forgot to provide destruction
- Acquire locks in inconsistent order



7. Sequential Consistency


P1
A1 A2 A3
P2
B1 B2 B3
A1 B1 A2 B2 B3 A3
✓
A2 B1 A1 B2 B3 A3
✗