CS 181AI
Lecture 9

Locking

Arthi Padmanabhan

Feb 15, 2023

Last Time

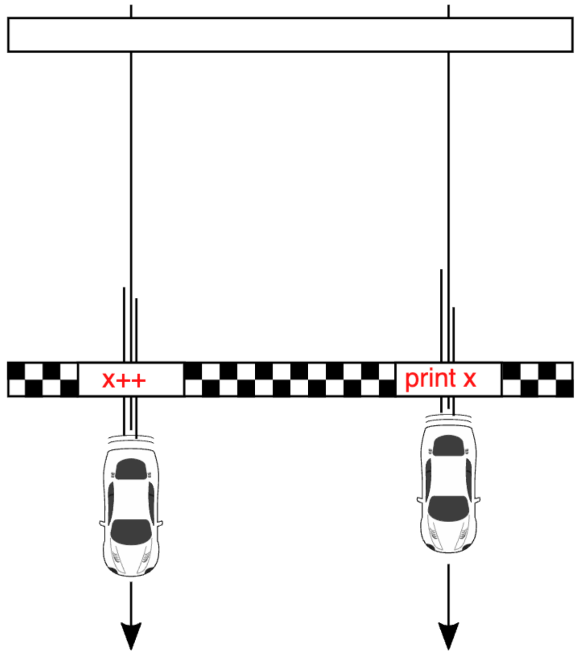
- Metrics for performance and common ways to improve them
- Moore’s Law: can’t simply wait for better CPUs anymore; current trend: more parallelism
- Intro to parallelizing code

Today: Parallelizing Code

- Races
- Deadlocks

Races

- A **race** occurs when two or more processes try to access the same data and at least one of those accesses is a write
- Avoid races using locks, synchronization



What are the possible values of the bank account?

```
Class Account
        amount = 20

        withdraw:
                amount -= 5
```

Order				Result
R1	W1	R2	W2	
R1	R2	W1	W2	
R1	R2	W2	W1	
R2	W2	R1	W1	
R2	R1	W2	W1	
R2	R1	W1	W2	

If withdraw is executed in parallel by two threads, what are the possible values of amount?

Locks

- Simplest lock is “mutual exclusion” lock, or mutex
- Associated with a single variable
- When someone has the lock, nobody else can read or write the variable

Mutex Lock

```
Class Account
    amount = 20
    lock = Lock()


    withdraw:
        acquire lock
        amount -= 5
        release lock
```

Mutex Lock

```
Class Account
        amount = 20
        lock = Lock()

        withdraw:
                acquire lock
                amount -= 5
                release lock
```

Important: “amount -=5” is both a read and a write. Why is it important that the lock is around both the read and the write?

Critical Section

- Everything between the acquire and release is the critical section

```
// code
// acquire lock
// critical section
// release lock
// code
```

- Critical sections should have minimal overhead: efficient, fair, and simple

Thread

- We’ll get to these when we talk about GPUs!
- For now, take thread to mean something that runs processes in parallel

Synchronization

- Mutex is great for resource control, but it doesn’t specify the *order* in which things should happen

Order of prints

- How do we make this print “I’m Thread 1” before “I’m Thread 2”?

```
def p1:
        print(“I’m Thread 1”)

def p2:
        print(“I’m Thread 2”)

Create thread that calls p1
Create thread that call p2
Initialize the two threads // initializes them in unknown order
```

Semaphore

- An unsigned integer: you set the value at the beginning and can never directly access the value again after that
- There are functions to increment and decrement by 1
- Decrement is potentially a blocking function – if decrementing would result in a negative number, the function waits until some other thread increments it
- When incrementing, if there is at least one thread waiting on the semaphore, exactly one thread will become unblocked

Order of prints

- How do we make this print “I’m Thread 1” before “I’m Thread 2”?

```
def p1:
        print(“I’m Thread 1”)

def p2:
        print(“I’m Thread 2”)

Create thread that calls p1
Create thread that call p2
Initialize the two threads // initializes them in unknown order
```

Order of prints

- How do we make this print “I’m Thread 1” before “I’m Thread 2”?

```
def p1:
        print(“I’m Thread 1”)
        sem.increment()

def p2:
        sem.decrement()
        print(“I’m Thread 2”)

Initialize semaphore sem with value 0
Create thread that calls p1
Create thread that call p2
Initialize the two threads // initializes them in unknown order
```

Barriers

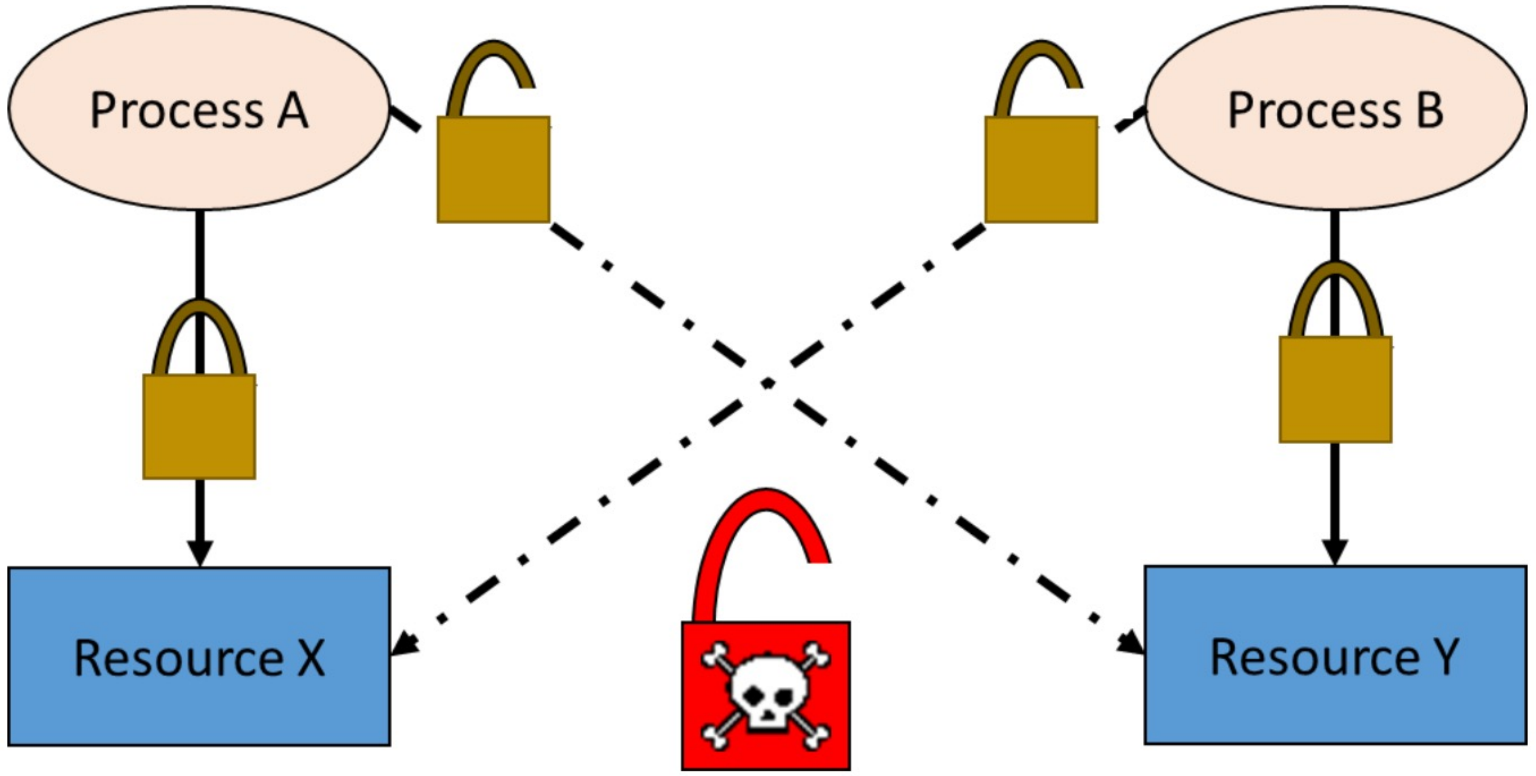
- Point in the program where threads stop and wait for all threads to reach that point before proceeding
- Describe a situation where this might be useful (hint: think about operations over arrays or grids)

Monitors & Condition Variables

- Monitor – mutual exclusion
- Conditional variables – synchronization
- Conditional variables – wait and notify
    - Wait: block myself and give up control of the lock (a queue is formed on this variable)
    - Notify: causes next thread in that queue to be released so it can re-acquire the lock and keep running

Deadlocks

- A deadlock occurs when none of the processes can make progress because there is a cycle in the resource requests



Bank Account Example

- User A wants to transfer money from Account 1 to Account 2

Bank Account Example

- User A wants to transfer money from Account 1 to Account 2
- User B wants to transfer money from Account 2 to Account 1

Next Time: GPUs and Parallelization

- All about GPUs!