Concurrency

Tam-CO

Angenda

1. Process and thread
2. Race condition
3. Synchronization
4. Java concurrency API
5. Advanced topics (not cover in this presentation)

1. Process & Thread

- Process
  - An instance of program running on machine
  - Each process has its own memory space
  - Cannot access other’s memory space
  - Scheduled by operating system
  - One of the most important concepts in computer science

1. Process & Thread (cnt)

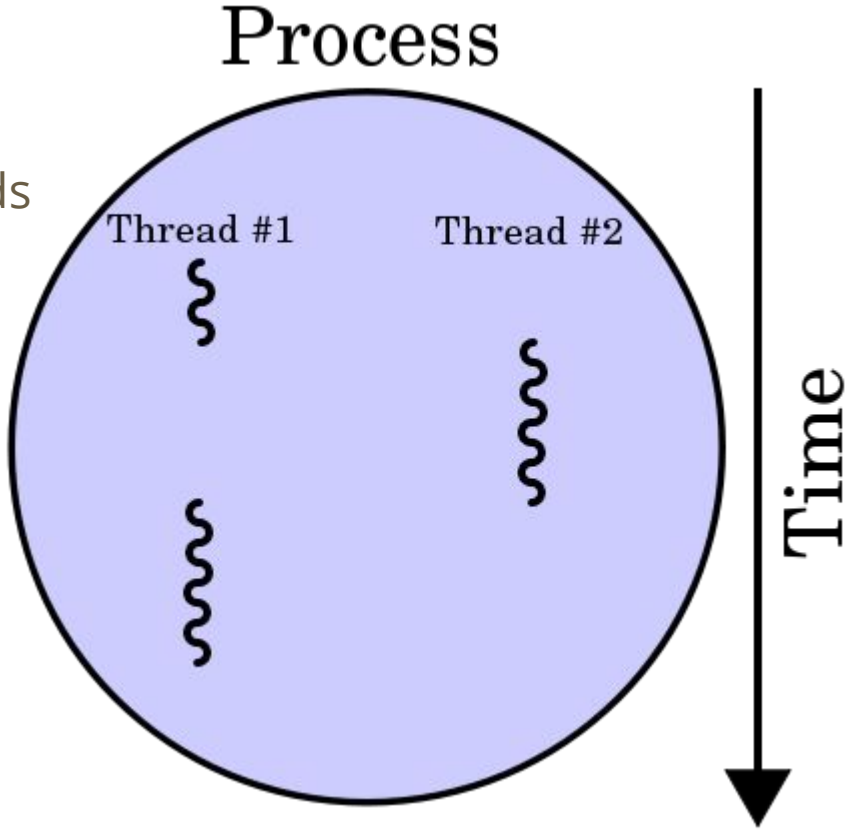
- Thread
  - Lightweight process, exists within process
  - Sequence of machine instructions
  - Share process’s resources including memory
  - Efficient but potential problemetic
  - Scheduled by process

1. Process & Thread (cnt)

- An image of process running on machine

1. Process & Thread (cnt)

- An virtual image of process with threads



1. Process & Thread (cnt)

- Why multi-processes and multi-threads
  - Take advantage of CPU speed
  - Take advantage of multi CPUs

2. Race condition

- Definition
  - Many threads access and modify the shared resource (memory) concurrently and cause unexpected result

2. Race condition (cnt)

- Example
  - Thread A execute increase
  - Thread B execute decrease

```
public class Main {

    // Sharing resource
    private static int balance = 100;

    public static void main(String[] args) {

    }

    private static void increase() {
        balance++;
    }

    private static void decrease() {
        balance--;
    }
}
```

2. Race condition (cnt)

- Example (cnt)
  - *balance*++ and *balance*-- very simple operations but not atomic
    - 1. Retrieve the current value of balance
    - 2. Increment the retrieved value by 1
    - 3. Store the incremented value back in balance

2. Race condition (cnt)

- Example (cnt)
  - A possible scenario if Thread A and Thread B execute concurrently
    - 1. Thread A: Retrieve balance
    - 2. Thread B: Retrieve balance
    - 3. Thread A: Increment retrieved value; result is 101
    - 4. Thread B: Decrement retrieved value; result is 99
    - 5. Thread A: Store result in balance; balance is now 101
    - 6. Thread B: Store result in balance; balance is now 99
  - This causes an unexpected result and very hard to debug

2. Race condition (cnt)

DEMO

3. Synchronization

- https://en.wikipedia.org/wiki/Synchronization_(computer_science)
- Meaning make *increase()* and *decrease()* operations atomic
    - 1. Thread A: Retrieve balance
    - 2. Thread A: Increment retrieved value; result is 101
    - 3. Thread A: Store result in balance; balance is now 101
    - 4. Thread B: Retrieve balance
    - 5. Thread B: Decrement retrieved value; result is 100
    - 6. Thread B: Store result in balance; balance is now 100

3. Synchronization (cnt)

- Very fortunately, hardware supports us for synchronization
- The key ability we require to implement synchronization in a multiprocessor is a set of hardware primitives with the ability to atomically read and modify a memory location
- Many modern hardware provides special atomic hardware instructions by either test-and-set the memory word or compare-and-swap contents of two memory words

3. Synchronization (cnt)

- Hardware supports
  - Test and set
    - Many processors have an atomic test-and-set machine language instruction - Intel, IBM
    - ```
      function TestAndSet(boolean_ref lock) {
          boolean initial = lock;
          lock = true;
          return initial;
      }
      ```
  - Compare and swap
      ```

3. Synchronization (cnt)

Explain testAndSet in code

4. Java concurrency API

- Java concurrency API to do synchronization
  - Lock interface
    - Aqquire lock when enter critical section
    - Release lock when exit critical section

4. Java concurrency API (cnt)

DEMO Lock

4. Java concurrency API (cnt)

- Java concurrency API to do synchronization
  - Synchronized method
    - Add keyword *synchronized* to method declaration
    - Use the intrinsic lock of current object
  - Synchronized statement
    - Must specify the object that provides the intrinsic lock

4. Java concurrency API (cnt)

DEMO Synchronized Method

4. Java concurrency API (cnt)

DEMO Synchronized Statement

5. Advanced topics (not cover in this presentation)

- Semaphore
- Deadlock
- Starvation and livelock

References

- https://en.wikipedia.org/wiki/Synchronization_(computer_science)
- https://en.wikipedia.org/wiki/Test-and-set
- JavaSE concurrency tutorial
- Operating system concepts (dinosaur book) Chapter 5: Synchronization

Thank you!

Time for discussion