Asynchronous programming & Crypto

COMPSCI210 Recitation
25th Mar 2013
Vamsi Thummala
Reminder on Java synchronized

• Combines: a lock and a CV

• In your Elevator, if you implement EventBarrier correctly, you only need locking, but not a CV
  – Java does not provide a way to do that directly
  – Locks are in turn implemented using “synchronized” as a library
  – java.util.concurrent.locks
  – Restricted for Elevator lab
java.util.concurrent

• Lock
• Thread safe collections
  – HashMap, Queue, and ..
• Semaphore
• CyclicBarrier
• ExecutorService
  – Thread pool
• FutureTask
Thread pooling

```java
public class SumFirstN implements Runnable {
    private final int _N;

    SumFirstN(int N) {
        _N = N;
    }

    @Override
    public void run() {
        long sum = 0;
        for (int i = 1; i < _N; i++) {
            sum += i;
        }
        System.out.println(sum);
    }
}

import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class Main {
    private static final int NTHREDS = 10;

    public static void main(String[] args) {
        ExecutorService executor = Executors.newFixedThreadPool(NTHREDS);
        for (int i = 0; i < 500; i++) {
            Runnable worker = new SumFirstN(i);
            executor.execute(worker);
        }
        executor.shutdown(); // Do not accept any more threads
        while (!executor.isTerminated()) {
        }
    }
}
```
What if each task is an IO or a network call?

- May take arbitrary amount of time to complete
- Each thread submit a task and just waits!
- Waste of resources
Asynchronous call

• Similar interface as Runnable

```java
public class CallBackTask implements Callable {
    public void call() {
    }
}
```
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class Main {
    private static final int NTHREADS = 10;

    public static void main(String[] args) {
        ExecutorService executor = Executors.newFixedThreadPool(NTHREADS);
        for (int i = 0; i < 500; i++) {
            Callable worker = new SumFirstN(i);
            executor.execute(worker);
        }
        executor.shutdown(); // Do not accept any more threads
        // Wait until all threads are finish
        while (!executor.isTerminated()) {
        }
    }
}

public class SumFirstN implements Callable {
    private final int _N;

    SumFirstN(int N) {
        _N = N;
    }

    @Override
    public void call() {
        long sum = 0;
        for (int i = 1; i < _N; i++) {
            sum += i;
        }
        System.out.println(sum);
    }
}
What if we expect a result from a callback?

- Typically, a read on disk
- A `Future` can capture the result of an asynchronous computation

```java
Future<Long> sum = executor.submit(new Callable<Integer>()
```
Using Callable with Future

public class SumFirstN implements Callable {
    private final int _N;

    SumFirstN(int N) {
        _N = N;
    }

    @Override
    public Long call() {
        long sum = 0;
        for (int i = 1; i < _N; i++) {
            sum += i;
        }
        return sum;
    }
}

import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class Main {
    private static final int NTHREDS = 10;
    List<Future<Long>> list = new ArrayList<Future<Long>>();
    public static void main(String[] args) {
        ExecutorService executor = Executors.newFixedThreadPool(NTHREDS);
        for (int i = 0; i < 500; i++) {
            Callable worker = new SumFirstN(i);
            Future<Long> sW = executor.execute(worker);
            list.add(sW);
        }
        // Now retrieve the result
        for (Future<Long> future : list) {
            long sum = future.get(); // ignored the try/catch block
        }
        executor.shutdown(); // Do not accept any more threads
        // Wait until all threads are finish
        while (!executor.isTerminated()) {
        }
    }
}
Asynchronous programming

• Event driven
  – Awaiting for IO
  – Awaiting for input for network
  – Awaiting for input from user
    • GUI, Mobile device (Android)

• Java Future library
  – Primitive but powerful stuff
  – More native support in other languages

• You will be doing callbacks in Lab4
Crypto: Concept checkers

• What is the basic assumption that cryptography relies on?
• What is a hash/fingerprint/digest?
• What is a digital signature?
• Symmetric vs Asymmetric crypto
• What is a nonce?
• What is a security/treat model?
• Type of attacks and defenses
“Cryptographic hash functions (also called secure hashing or SHA) are useful even if the result digest (also called a hash or fingerprint) is not encrypted, as it is with digital signatures. For example, if Alice knows a secret, and passes Bob a digest of the secret, then Bob can determine if another party also knows the secret, even without knowing the secret himself.”
Symmetric and Asymmetric Crypto: Better Together

- Use asymmetric crypto to “handshake” and establish a secret session key (slow, but allows for key distribution).
- Then use the key to talk with symmetric crypto (fast and cheap)
- **Example:** Secure Sockets Layer (SSL) or Transport-Layer Security (TLS), used in HTTPS (Secure HTTP), SSH, SCP, etc.

```
Client

“SYN, etc.”

“My public key is K.”

“Let’s establish a session key:

{S}K .”

{M}S

[encrypted data or content]

…”

Server

[Image of Client and Server with handshake diagram]
```