Lab 17
Fork-Join Model

TA：Bo-Hua Xu

Professor：Hsung-Pin Chang

Operating System Lab

Outline

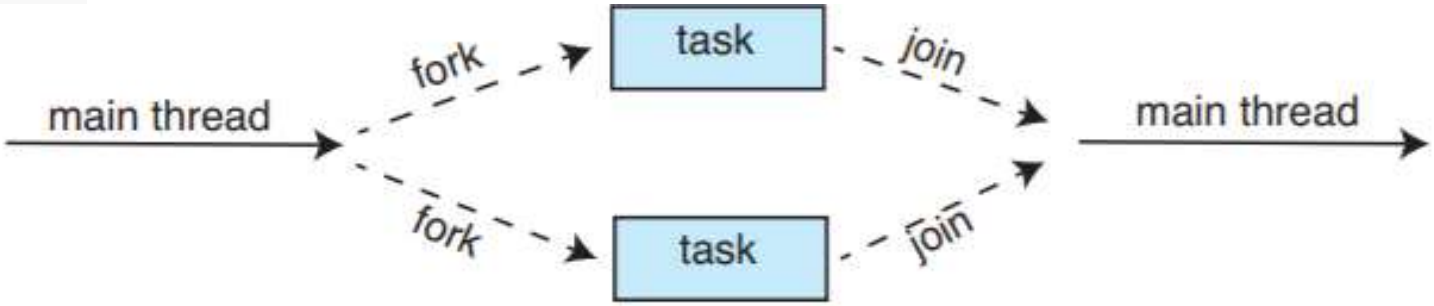
- **Fork-Join Model**
- Java Fork-Join Framework
- Fork-Join Example
- Exercise

Fork-Join Model

- Fork-Join Model is a popular synchronous model.
- The main thread which has a big task to solve creates (forks) one or more child threads and each thread is assigned a small task divided by the big task.
- After that, the main thread waits for the children to terminate and join with it. Then, it can retrieve and combine their results.

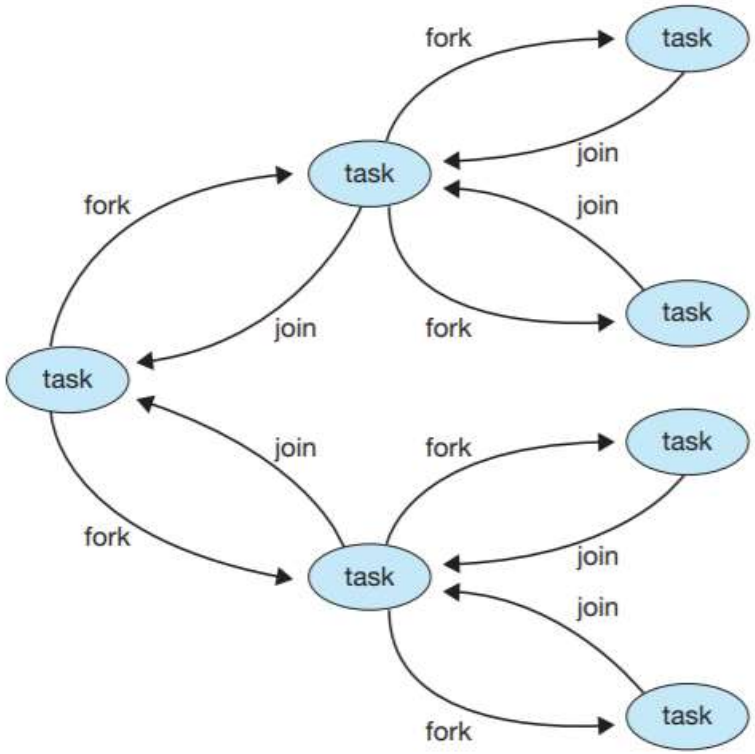
Fork-Join Model

- This is a basic Fork-Join Model. We can see a main thread fork two child tasks to handle with it big task, after finishing tasks they will join the main thread.



Fork-Join Model

- Beside a main thread fork many child tasks, we also can make a multiple level Fork-Join Model.



Outline

- **Fork-Join Model**
- **Java Fork-Join Framework**
- Fork-Join Example
- Exercise

Java Fork-Join Framework

- The fork/join framework is an implementation of the ExecutorService interface that helps you take advantage of multiple processors.
- It is designed for work that can be broken into smaller pieces recursively.
- The goal is to use all the available processing power to enhance the performance of your application.

Java Fork-Join Framework

Write the ForkJoinTask class

1.With return value use RecursiveTask

2.Without return value use RecursiveAction

Write the compute() function in ForkJoinTask class

Write the code to do what you want to do

Initialize the ForkJoinPool and invoke the ForkJoinTask

Run the task

Java Fork-Join Framework

- Firstly, we extends a ForkJoinTask class, there are two types of ForkJoinTask, and in it we can add code that performs a segment of the work.
  - 1.RecursiveAction：

    This won't return result when ForkJoinTask.join()

    Ex: public class Forktask extends RecursiveAction{}
  - 2. RecursiveTask：

    This will return result when ForkJoinTask.join()

    Ex: public class Forktask extends RecursiveAction{}

Java Fork-Join Framework

- Secondly, you implement the abstract compute() method in the ForkJoinTask class you just extends.
- In compute(),you add code that splits a work to small works until it is small enough to execute.

```
protected void compute() {
    if (work < Threshold) {
        computeDirectly();return;
    }
    else{
        int split=work/2;
```

Java Fork-Join Framework

//when you need to handle with return value

```
Forktask f1 = new Forktask(split);
Forktask f2 = new Forktask(split);
F1.fork();
return f2.compute() + f1.join();}}
```

//when you don’t need to handle with return value ,or don’t have return value.

```
invokeAll(new Forktask(split),
          new Forktask(split));
```

Java Fork-Join Framework

- fork()：

  Arranges to asynchronously execute this task in the pool the current task is running in, if applicable, or using the ForkJoinPool.commonPool() if not inForkJoinPool().

- join()

  Returns the result of the computation when it is done.

- invokeAll(Collection<T> tasks)：

  Forks all tasks in the specified collection, returning when isDone holds for each task or an (unchecked) exception is encountered, in which case the exception is rethrown.

Java Fork-Join Framework

- Finally, set up the task to run in a ForkJoinPool.
- ForkJoinPool：

  An ExecutorService for running ForkJoinTasks. A ForkJoinPool provides the entry point for submissions from non-ForkJoinTask clients, as well as management and monitoring operations.

1. Create a task that represents all of the work to be done.

   ```
   Forktask ft = new Forktask(work);
   ```

2. Create the ForkJoinPool that will run the task.

   ```
   ForkJoinPool pool = new ForkJoinPool();
   ```

3. Run the task.

   ```
   pool.invoke(ft);
   ```

Outline

- **Fork-Join Model**
- **Java Fork-Join Framework**
- **Fork-Join Example**
- Exercise

Fork-Join Example

- sumOfArray.java

```
import java.util.concurrent.ExecutionException;
import java.util.concurrent.ForkJoinPool;
import java.util.concurrent.RecursiveTask;
import java.util.stream.LongStream;

public interface sumOfArray{
        long sumUp(long[] numbers);
}
```

Fork-Join Example

- ForkJoinSum.java

```
import java.util.concurrent.ExecutionException;
import java.util.concurrent.ForkJoinPool;
import java.util.concurrent.RecursiveTask;
import java.util.stream.LongStream;

public class ForkJoinSum implements sumOfArray {
        //creat forkjoin pool
        private ForkJoinPool pool;
        //write the forkjointask with RecursiveTask
        private static class SumTask extends RecursiveTask<Long>{
                private long[] numbers;
                private int from;
                private int to;
        //initialize
                public SumTask(long[] numbers,int from,int to) {
                        this.numbers = numbers;
                        this.from = from;
                        this.to = to;
                }
                @Override
        //write the code to do sum
                protected Long compute(){
                        if(to-from < 6){
                                long total = 0;
                                for(int i = from;i <= to;i++){
                                        total += numbers[i];
```

Fork-Join Example

```
                    total += numbers[i];
                }
                return total;
            }
        //not small enough so we spilt size to half
            else{
                int split =(from+to)/2;
                SumTask f1 = new SumTask(numbers,from,split);
                SumTask f2 = new SumTask(numbers,split+1,to);
                f1.fork();
                return f2.compute()+f1.join();
            }
        }
    }
    //initialize
    public ForkJoinSum() {
        pool = new ForkJoinPool();
    }
    //invoke the task to pool
    public long sumUp(long[] numbers) {
        return pool.invoke(new SumTask(numbers, 0,numbers.length-1));
    }
}
```

Fork-Join Example

- example.java

```
import java.util.concurrent.ExecutionException;
import java.util.concurrent.ForkJoinPool;
import java.util.concurrent.RecursiveTask;
import java.util.stream.LongStream;


public class example {
        public static void main(String[] args){
                //creat a 1000 length array
                long[] numbers = LongStream.rangeClosed(1,1000).toArray();
                //creat fork join object
                sumOfArray sum = new ForkJoinSum();
                //print result
                System.out.println(sum.sumUp(numbers));
        }
}
```

```
oslab@oslab-VirtualBox:~/java_fork_join/example$ javac example.java ForkJoinSum
.java sumOfArray.java
oslab@oslab-VirtualBox:~/java_fork_join/example$ java example
500500
```

Outline

- Fork-Join Model
- Java Fork-Join Framework
- Fork-Join Example
- Exercise

Exercises

- **Write a merge sort program to merge an array with 100 elements array with fork/join Framework.**

Reference

- The Java Tutorial
- https://docs.oracle.com/javase/tutorial/essential/concurrency/forkjoin.html
- OS concepts 10th chapter 4 4.5.2