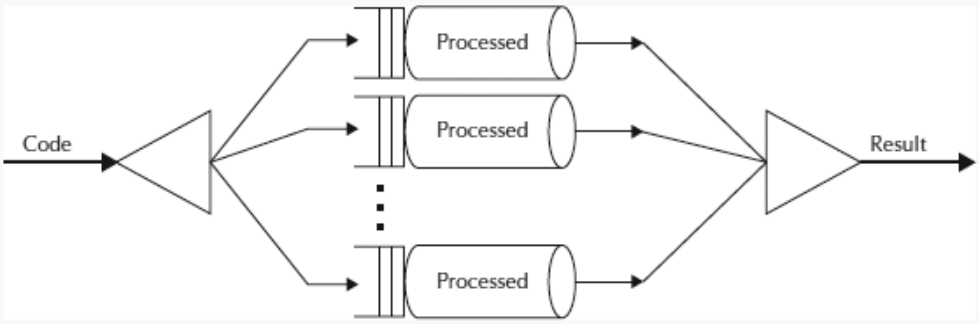
ForkJoinPools

ForkJoin được implementation từ ExecutorService interface giúp tận dụng được nhiều bộ xử lý.

Về cơ bản, ForkJoin chia nhỏ nhiệm vụ trong tay thành các nhiệm vụ cho đến khi nhiệm vụ đủ đơn giản để có thể giải quyết việc mà không cần phải chia them. Giống như việc chia để trị. Còn 1 vấn đề nữa là không có luông nào nghỉ. Nó sẽ thực hiện stealing lấy nhưng việc mà các luồng bận đang làm cho luồng đang nhàn dỗi.



Fork Join Framework

Mã giả :

```
Result solve(Problem problem) {
      if (problem is small)
            directly solve problem
      else {
            split problem into independent parts
            fork new subtasks to solve each part
            join all subtasks
            compose result from subresults
      }
}
```

Điểm khác biệt giữa ForkJoin và executorService

Là ForkJoin sử dụng thuật toán WorkStealing.

Ví dụ Hiệu quả của forkJoinPool

```java
package itlap.ex1;

import java.util.concurrent.ForkJoinTask;
import java.util.concurrent.RecursiveAction;

public class ForkJoinFibonacci extends RecursiveAction {

    private static final long threshold = 10;
    private volatile long number;

    public ForkJoinFibonacci(long number) {
        this.number = number;
    }

    public long getNumber() {
        return number;
    }

    @Override
    protected void compute() {
        long n = number;
        if (n <= threshold) {
            number = fib(n);
        } else {
            ForkJoinFibonacci f1 = new ForkJoinFibonacci(n - 1);
            ForkJoinFibonacci f2 = new ForkJoinFibonacci(n - 2);
            ForkJoinTask.invokeAll(f1, f2);
            number = f1.number + f2.number;
        }
    }

    private static long fib(long n) {
        if (n <= 1) return n;
        else return fib(n - 1) + fib(n - 2);
    }

}
```

```java
package itlap.ex1;

import java.util.concurrent.ForkJoinPool;

public class ForkJoinUser {

    public static void main(String[] args) {
        long start = System.nanoTime();
        ForkJoinFibonacci task = new ForkJoinFibonacci(50);
        new ForkJoinPool().invoke(task);

        System.out.println(task.getNumber());
        long end = System.nanoTime();
        System.out.println("Time  : " + (end - start)/1000000000 + "  seconds");

    }
}
```

kết quả chạy tìm số 50 của dạy finonacci : là 76s

```java
package itlap.ex1;

public class NoUserForkJoin {
    public static void main(String[] args) {
        long start = System.nanoTime();
        System.out.print(fibonacci(50));
        long end = System.nanoTime();
        System.out.println("Time  : " + (end - start)/1000000000 + "  seconds");
    }

    public static int fibonacci(int n) {
        if (n < 0) {
            return -1;
        } else if (n == 0 || n == 1) {
            return n;
        } else {
            return fibonacci(n - 1) + fibonacci(n - 2);
        }
    }

}
```

kết quả chạy là 191s.