ForkJoinPool được giới thiệu trong Java 7

Nó tương tự như **khung Executor** nhưng có một điểm khác biệt. ForkJoinPool hoạt động đệ quy, không giống như các luồng Executor, phân tách tác vụ và gửi các phần nhỏ hơn cho các luồng. ForkJoinPool nhận một nhiệm vụ lớn, chia nó thành các nhiệm vụ nhỏ hơn và những nhiệm vụ nhỏ hơn đó lại tự phân chia thành các nhiệm vụ cho đến khi mỗi nhiệm vụ con là nguyên tử hoặc không chia hết. Vì vậy, nó hoạt động đệ quy. ForkJoinPool phân chia các tác vụ cho các luồng thực thi trong Thread Pool. Framework Fork/ Join sử dụng thuật toán [work-stealing](https://en.wikipedia.org/wiki/Work_stealing). Các luồng sẽ thực thi công việc của mình trên một bộ xử lý riêng biệt (thread/ processor), khi làm hết việc của mình, nó lấy bớt (steal) các tác vụ từ các luồng khác đang bận rộn.

Thuật toán [Work stealing](https://en.wikipedia.org/wiki/Work_stealing) :

[Work stealing](https://en.wikipedia.org/wiki/Work_stealing) là cơ chế giúp scheduler (có thể là trên ngôn ngữ, hoặc OS) có thể thực hiện việc tạo thên M thread mới hoạt động mượt mà trên N core, với M có thể lớn hơn N rất nhiều.

Idea của work-stealing scheduler là mỗi một core sẽ có một queue những task phải làm. Mỗi task đó bao gồm một list các instructions phải thực hiện một cách tuần tự. Khi một processor làm hết việc của mình, nó sẽ nhìn ngó sang các processor xung quanh, xem có gì cần làm không và “steal” công việc từ đó.

Một mô hình khác với work stealing là work sharing, tức là mỗi task sẽ quyết fix là sẽ được thực hiện trên processor nào.

Các ForkJoinPool là một [Thread Pool](https://gpcoder.com/3548-huong-dan-tao-va-su-dung-threadpool-trong-java/) đặc biệt được thiết kế để làm việc tốt với chia tách công việc fork/ join. ForkJoinPool nằm trong gói java.util.concurrent, vì vậy tên lớp đầy đủ là **java.util.concurrent.ForkJoinPool**. Một số lớp tiêu biểu của Fork/ Join Framework:

**ForkJoinTask<V>**: một abstract class định nghĩa task sẽ được thực thi trong một ForkJoinPool.

**ForkJoinPool**: là một thread pool quản lý việc thực thi các ForkJoinTasks.

**RecursiveAction**: là một lớp con của ForkJoinTask, nó thực thi tác vụ mà không trả lại bất kỳ kết quả nào (action).

**RecursiveTask<V>**: là một lớp con của ForkJoinTask, nó thực thi tác vụ mà có trả lại kết quả (task).

Cú pháp tạo ForkJoinPool:

|  |  |
| --- | --- |
|  | ForkJoinPool forkJoinPool = new ForkJoinPool(numOfProcessor); |

Tạo một ForkJoinPool với tham số là số lượng luồng hoặc các CPU bạn muốn làm việc đồng thời trên các nhiệm vụ được truyền vào ForkJoinPool. Nếu bạn không xác định numOfProcessor, nó sẽ lấy số bộ vi xử lý có sẵn cho máy ảo Java để thực thi.

Trong đó:

Nguyên tắc Fork/ Join gồm 2 bước được thực hiện đệ quy. Hai bước này là: bước chia tách (fork/ split) và bước gộp (join/ merge).

Với Fork: Chúng chia nhỏ công việc cần làm thành các công việc con . Bằng cách chia nhỏ thành các nhiệm vụ con, mỗi nhiệm vụ con có thể được thực hiện song song bởi các CPU khác nhau, hoặc các luồng khác nhau trên cùng một CPU.

Với Join: Khi một nhiệm vụ (parent task) đã tự tách mình thành các nhiệm vụ con (sub task), nhiệm vụ cha sẽ đợi cho đến khi các nhiệm vụ con hoàn thành.

Khi nhiệm vụ con đã hoàn thành, nhiệm vụ cha có thể kết hợp (join/ merge) tất cả các kết quả con vào một kết quả cuối cùng