Mục lục

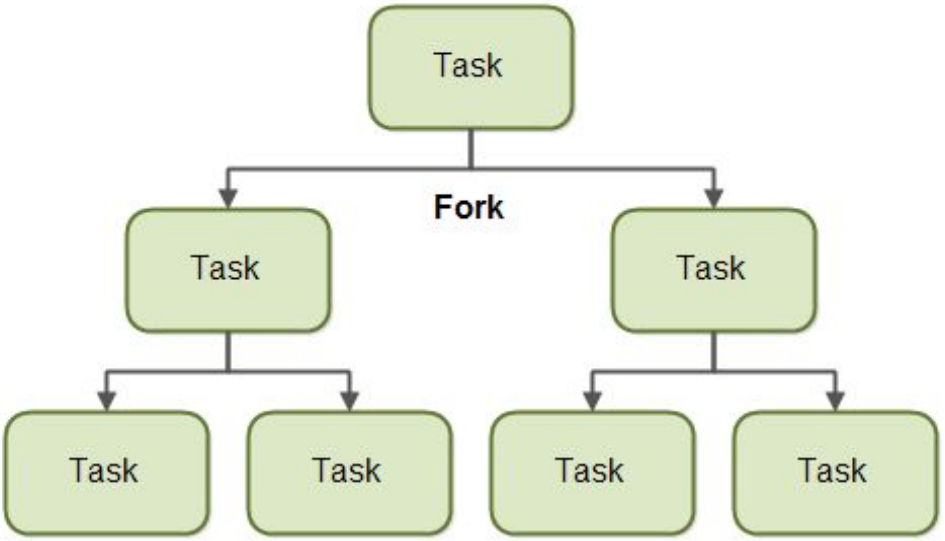
1. Fork?

a. Khái niệm.

- Là chia nhỏ task, đệ quy chia nhỏ nhiệm vụ thành các nhiệm vụ phụ nhỏ hơn cho đến khi chúng đơn giản đủ để được thực hiện xử lý không đồng bộ.

b. Nguyên tắc hoạt động.

- Một nhiệm vụ (parent task) sử dụng nguyên tắc fork để chia tách chính nó thành các nhiệm vụ con (sub task) nhỏ hơn để có thể được thực hiện đồng thời. Điều này được minh họa trong sơ đồ dưới đây:



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.

Một nhiệm vụ chỉ phân chia thành các nhiệm vụ phụ nếu công việc mà nhiệm vụ được đưa ra là đủ lớn để điều này có ý nghĩa. Có một chi phí để chia tách một nhiệm vụ thành các nhiệm vụ phụ, vì vậy với số lượng nhỏ công việc trên không thể lớn hơn tốc độ đạt được bằng cách thực hiện các công việc phụ đồng thời.

2. Join?

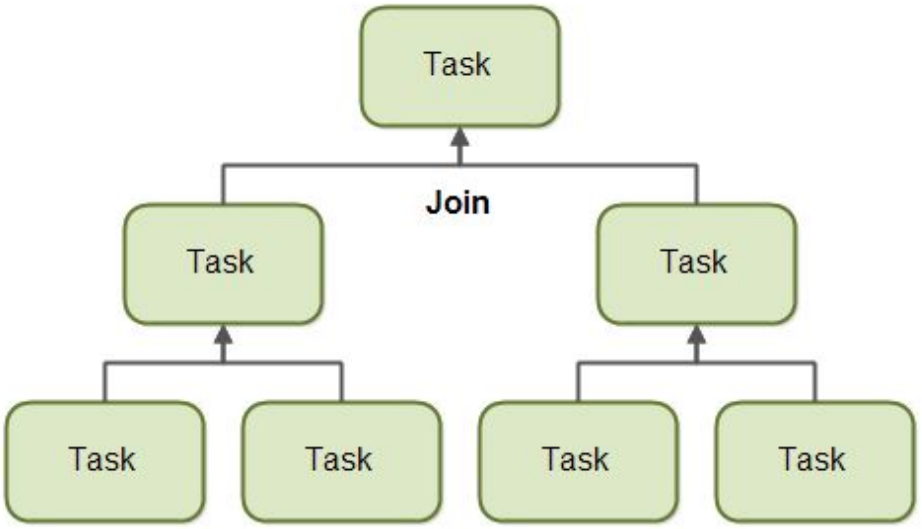
a. Khái niệm.

- Phần gộp kết quả, trong đó các kết quả của tất cả các nhiệm vụ phụ được đệ quy một cách đệ quy vào một kết quả, hoặc trong trường hợp một nhiệm vụ trả về void, chương trình chỉ cần đợi cho đến khi mỗi nhiệm vụ phụ được thực hiện.

b. Nguyên tắc hoạt động.

- 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. Điều này được minh họa trong sơ đồ dưới đây:



Tất nhiên, không phải tất cả các loại nhiệm vụ có thể trả về một kết quả. Nếu các tác vụ không trả lại kết quả, thì một nhiệm vụ chỉ cần đợi cho các công việc phụ của nó hoàn thành, không có sự kết hợp kết quả nào xảy ra sau đó.

3. Thế nào là ForkJoinPool?

a. Khái niệm.

- ForkJoinPool là một Thread Pool đặc biệt được thiết kế để làm việc tốt với chia tách công việc fork/ join.
- Để cung cấp thực hiện xử lý song song hiệu quả, các fork / join Framework sử dụng hồ chứa (pool) các Thread được gọi là ForkJoinPool.

b. Nguyên tắc hoạt động.

– ForkJoinPool tương tự như Java ExecutorService nhưng với một sự khác biệt. ForkJoinPool phân chia các tác vụ cho các luồng thực thi trong Thread Pool. Framework Fork/ Join sử dụng thuật toán 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.
- Work stealing là gì?
    + 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.

4. Tại sao cần dùng nó?

- Cung cấp các công cụ giúp tăng tốc xử lý song song bằng cách cố gắng sử dụng tất cả các lõi bộ xử lý có sẵn, được thực hiện thông qua cách tiếp cận phân chia (fork) và gộp (join) task. Mục đích là để sử dụng tất cả các khả năng xử lý để nâng cao hiệu suất cho các ứng dụng.