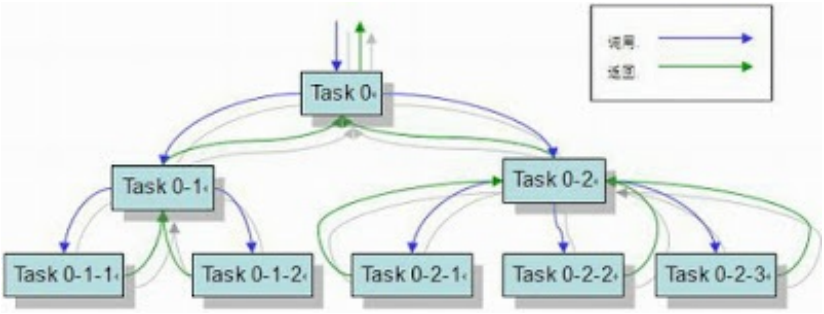
Java線程池相關-ForkJoinPool

slamke.blogspot.tw/2015/07/java-forkjoinpool.html

當一個問題被拆分成兩個或者多個子問題的時候，需要啟動多個子線程去執行，在必要的情況下會迭代的依次啟動下去。這裡就產生了一些線程之間的依賴，這個大的問題需要等待它的子問題線程返回，因此需要某些機制來保證他們的同步。ThreadPoolExecutor默認使用的線程池是期望他們所有執行的任務都是不相關的，可以儘可能的並行執行。

ForkJoinPool有一個特點是work stealing。每個工作線程都有自己的工作隊列，這是使用deque來實現的。當一個任務劃分一個新線程時，它將自己推到 deque 的頭部。當一個任務執行與另一個未完成任務的合併操作時，它會將另一個任務推到隊列頭部並執行，而不會休眠以等待另一任務完成（像 Thread.join() 的操作一樣）。當線程的任務隊列為空，它將嘗試從另一個線程的 deque 的尾部 竊取另一個任務。如果我們用傳統的ThreadPoolExecutor則比較難用上work stealing的技術。
Fork/Join 模式有自己的適用範圍。如果一個應用能被分解成多個子任務，並且組合多個子任務的結果就能夠獲得最終的答案，那麼這個應用就適合用 Fork/Join 模式來解決。

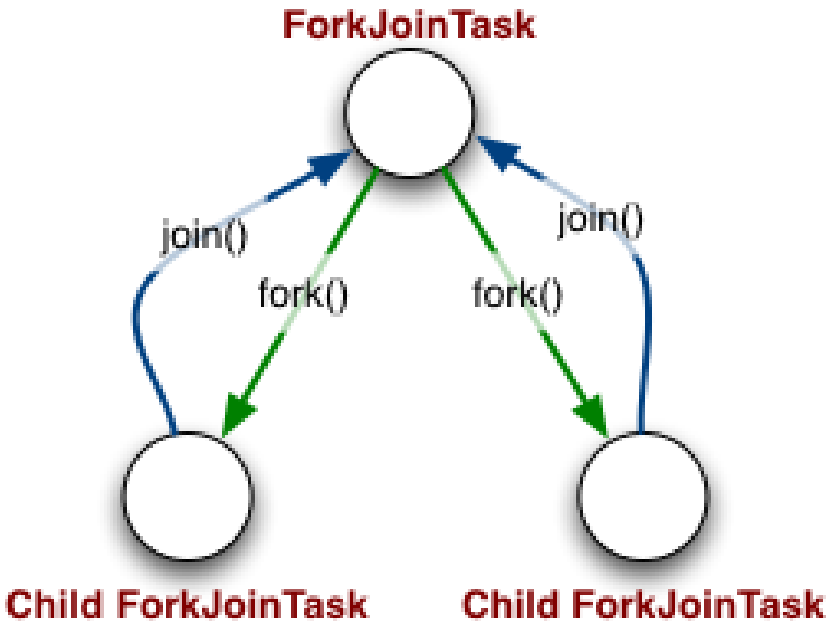


ForkJoinPool核心的添加是新的ForkJoinPool執行者，專門執行實現了ForkJoinTask接口的實例。ForkJoinTask對象支持創建子任務來等待子任務完成。有了這些清晰的語義，當一個任務正在等待另一個任務完成並且有待執行的任務時，executor就能夠通過」偷取」任務，在內部的線程池裡分發任務。

ForkJoinTask對象主要有兩個重要的方法：

- fork()方法允許ForkJoinTask任務異步執行，也允許一個新的ForkJoinTask從存在的ForkJoinTask中被啟動。
- 反過來, join()方法允許一個ForkJoinTask等待另一個ForkJoinTask執行完成。

如圖所示，通過fork()和join()實現任務間的相互合作。注意fork()和join()方法名稱不應該與POSIX中的進程能夠複製自己的過程相混淆。fork()只會讓ForkJoinPool調度一個新的任務，而不會創建子虛擬機。



有兩種類型的ForkJoinTask的定義：

- RecursiveAction的實例代表執行沒有返回結果。
- 相反，RecursiveTask會有返回值。

通常，RecursiveTask是首選的，因為大部分分而治之的算法會在數據集上計算後返回結果。對於任務的執行，不同的同步和異步選項是可選的，這樣就可以實現複雜的模式。
注意點：ForkJoinTask對應的fork/join任務應該是純內存算法，而沒有I/O操作。此外，應該儘可能避免通過共享狀態來進行任務間的通信，因為這通常意味著加鎖會被執行。理想情況下，僅當一個任務fork另一個任務或一個任務join另一個任務時才進行任務通信。

ForkJoinPool系統介紹：http://ifeve.com/fork-and-join-java/
Java線程池概述：http://blog.csdn.net/dm_vincent/article/details/39505977
詳解介紹了線程池的線程數目如何設置以及普通線程池和ForkJoinPool的區別的使用場景。

ForkJoinPool簡單源碼解析：http://blog.csdn.net/aesop_wubo/article/details/10300273

ForkJoinPool介紹：
http://www.ibm.com/developerworks/cn/java/j-jtp11137.html
http://www.ibm.com/developerworks/cn/java/j-lo-forkjoin/index.html