Java線程(十一)：Fork/Join-Java並行計算框架

blog.csdn.net/ghsau/article/details/46287769

標籤： 並行計算Fork-Join
2015-07-22 21:12 2275人閱讀 收藏 舉報
分類：

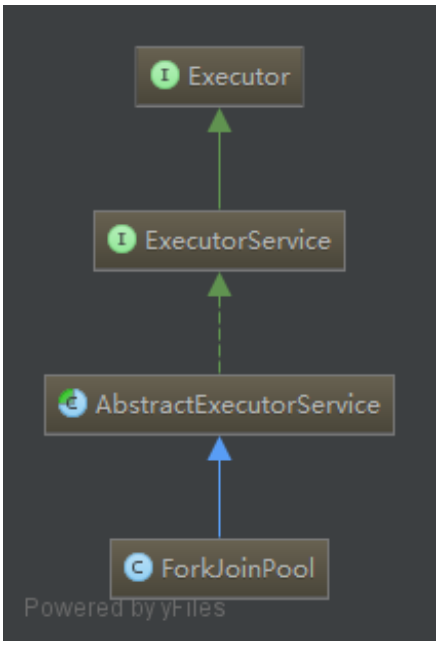
並行計算在處處都有大數據的今天已經不是一個新鮮的詞彙了，現在已經有單機多核甚至多機集群並行計算，注意，這裡說的是並行，而不是並發。嚴格的將，**並行是指系統內有多個任務同時執行**，而**並發是指系統內有多個任務同時存在**，不同的任務按時間分片的方式切換執行，由於切換的時間很短，給人的感覺好像是在同時執行。
Java在JDK7之後加入了並行計算的框架Fork/Join，可以解決我們系統中大數據計算的性能問題。Fork/Join採用的是分治法，Fork是將一個大任務拆分成若干個子任務，子任務分別去計算，而Join是獲取到子任務的計算結果，然後合併，這個是遞歸的過程。子任務被分配到不同的核上執行時，效率最高。偽代碼如下：

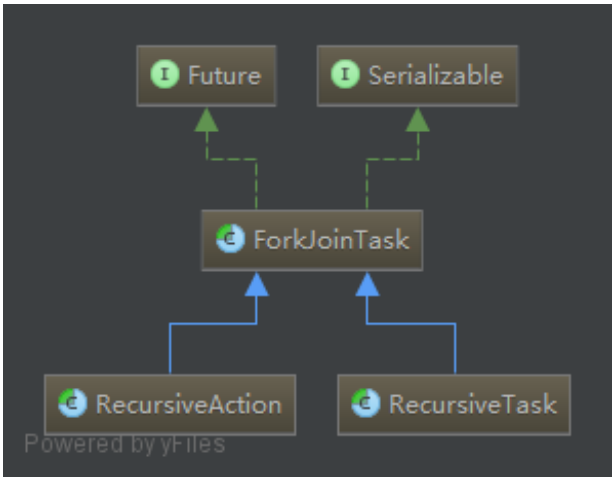
```
Result solve(Problem problem) {
    if (problem is small)
        directly solve problem
    else {
        split problem into independent parts
        fork new subtasks to solve each part
        join all subtasks
        compose result from subresults
    }
}
```

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10

Fork/Join框架的核心類是ForkJoinPool，它能夠接收一個ForkJoinTask，並得到計算結果。ForkJoinTask有兩個子類，RecursiveTask（有返回值）和RecursiveAction（無返回結果），我們自己定義任務時，只需選擇這兩個類繼承即可。類圖如下：

下面來看一個實例：計算一個超大數組所有元素的和。代碼如下：






```
import java.util.Arrays;
import java.util.Random;
import java.util.concurrent.ExecutionException;
import java.util.concurrent.ForkJoinPool;
import java.util.concurrent.RecursiveTask;


public class SumTask extends RecursiveTask<Integer> {

    private static final long serialVersionUID =
-6196480027075657316L;

    private static final int THRESHOLD = 500000;

    private long[] array;

    private int low;

    private int high;

    public SumTask(long[] array, int low, int
high) {
        this.array = array;
        this.low = low;
        this.high = high;
    }

    @Override
    protected Integer compute() {
        int sum = 0;
        if (high - low <= THRESHOLD) {

            for (int i = low; i < high; i++) {
                sum += array[i];
            }
        } else {

            int mid = (low + high) >>> 1;
            SumTask left = new SumTask(array, low, mid);
            SumTask right = new SumTask(array, mid + 1, high);


            left.fork();
            right.fork();


            sum = left.join() + right.join();
```

```
        }
        return sum;
    }

    public static void main(String[] args) throws ExecutionException,
InterruptedException {
        long[] array = genArray(1000000);

        System.out.println(Arrays.toString(array));


        SumTask sumTask = new SumTask(array, 0, array.length - 1);

        long begin = System.currentTimeMillis();


        ForkJoinPool forkJoinPool = new ForkJoinPool();


        forkJoinPool.submit(sumTask);


        Integer result = sumTask.get();

        long end = System.currentTimeMillis();

        System.out.println(String.format("結果 %s 耗時 %sms", result, end - begin));
    }

    private static long[] genArray(int size) {
        long[] array = new long[size];
        for (int i = 0; i < size; i++) {
            array[i] = new Random().nextLong();
        }
        return array;
    }
}
```

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9

- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25
- 26
- 27
- 28
- 29
- 30
- 31
- 32
- 33
- 34
- 35
- 36
- 37
- 38
- 39
- 40
- 41
- 42
- 43
- 44
- 45
- 46

- 47
- 48
- 49
- 50
- 51
- 52
- 53
- 54
- 55
- 56
- 57
- 58
- 59
- 60
- 61
- 62
- 63
- 64
- 65
- 66
- 67
- 68
- 69
- 70
- 71
- 72
- 73
- 74
- 75
- 76
- 77
- 78
- 79
- 80
- 81
- 82
- 83

- 84

我們通過調整閾值（THRESHOLD），可以發現耗時是不一樣的。實際應用中，如果需要分割的任務大小是固定的，可以經過測試，得到最佳閾值；如果大小不是固定的，就需要設計一個可伸縮的算法，來動態計算出閾值。如果子任務很多，效率並不一定會高。
未完待續。。。

參考資料

http://gee.cs.oswego.edu/dl/papers/fj.pdf
https://docs.oracle.com/javase/tutorial/essential/concurrency/forkjoin.html
https://www.ibm.com/developerworks/cn/java/j-lo-forkjoin/
http://www.ibm.com/developerworks/cn/java/j-jtp11137.html

- 上一篇MySQL從5.5升級到5.6，TIMESTAMP的變化
- 下一篇CiscoVPNClient不支持WIN10

我的同類文章

- •2014-11-22
- •2013-12-27
- •2013-11-13
- •2012-04-20
- •2016-02-05
- •2014-08-18
- •2013-12-26
- •2012-10-25
- •2012-04-14
- •2012-04-09

更多文章