<http://www.infoq.com/cn/news/2014/04/java-se2-tool-improve>

Java 8在Lambda表达式、接口默认方式、新的日期API等方面引入的新特性广受关注，同时在并发编程方面也做出了大量改进。以往的几个Java版本都对java.util.concurrent做了不同程度的增强，比如Java 7的Fork/Join框架，而Java 8则进一步在java.util.concurrent下增加了新的接口、类与方法。目前[java.util.concurrent的官方文档](http://docs.oracle.com/javase/8/docs/api/index.html?java/util/concurrent/package-summary.html)已经更新，变更部分总结如下：

**新的类以及接口**

[java.util.concurrent](http://docs.oracle.com/javase/8/docs/api/java/util/concurrent/package-summary.html) 中增加了两个接口四个类：

1. [CompletableFuture.AsynchronousCompletionTask](http://docs.oracle.com/javase/8/docs/api/java/util/concurrent/CompletableFuture.AsynchronousCompletionTask.html)接口：标识在async方法中执行的异步任务。
2. [CompletionStage<T>](http://docs.oracle.com/javase/8/docs/api/java/util/concurrent/CompletionStage.html)接口：异步计算中可能出现的一个阶段，也就是说当一个CompletionStage 完成时执行的动作或计算。
3. [CompletableFuture<T>](http://docs.oracle.com/javase/8/docs/api/java/util/concurrent/CompletableFuture.html)类：一个可以确定完成状态的Future。有关CompletableFuture的详细用法可参考NoBlogDefFound上的《[Java 8: CompletableFuture in action](http://www.nurkiewicz.com/2013/05/java-8-completablefuture-in-action.html)》一文。
4. [ConcurrentHashMap.KeySetView<K,V>](http://docs.oracle.com/javase/8/docs/api/java/util/concurrent/ConcurrentHashMap.KeySetView.html)类：ConcurrentHashMap 的键的集合视图。
5. [CountedCompleter<T>](http://docs.oracle.com/javase/8/docs/api/java/util/concurrent/CountedCompleter.html)类：一个在没有其他action等待的情况下，会执行一个完成action的 ForkJoinTask 。
6. [CompletionException](http://docs.oracle.com/javase/8/docs/api/java/util/concurrent/CompletionException.html)类：异常类。

**ConcurrentHashMap增加新方法**

在Java 8中，集合框架基于streams和Lambda表达式做了全新调整：

[ConcurrentHashMap](http://docs.oracle.com/javase/8/docs/api/java/util/concurrent/ConcurrentHashMap.html)增加了30多个方法，包括foreach系列（forEach,forEachKey, forEachValue, forEachEntry）、search系列（search, searchKeys, searchValues, searchEntries）、reduce系列（reduce, reduceToDouble, reduceToLong）以及mappingCount 、newKeySet等方法， 增强后的[ConcurrentHashMap](http://docs.oracle.com/javase/8/docs/api/java/util/concurrent/ConcurrentHashMap.html)更适合做缓存了， 读者可以看看这篇用ConcurrentHashMap类和lambda表达式实[现本地缓存](http://ifeve.com/java8-local-caching/)的文章。

**java.util.concurrent.atomic包的改进**

java.util.concurrent.atomic包中增加了四个新的类：DoubleAccumulator、DoubleAdder、LongAccumulator、 LongAdder，这四个类的作者是并发大师Doug lea。

原有的Atomic系列类通过CAS来保证并发时操作的原子性，但是高并发也就意味着CAS的失败次数会增多，失败次数的增多会引起更多线程的重试，最后导致AtomicLong的效率降低。新的四个类通过减少并发，将单一value的更新压力分担到多个value中去，降低单个value的“热度”以提高高并发情况下的吞吐量，京东的刘锟洋[详细分析了LongAdder的源代码](http://www.liuinsect.com/2014/04/15/%E6%AF%94atomiclong%E8%BF%98%E9%AB%98%E6%95%88%E7%9A%84longadder-%E6%BA%90%E7%A0%81%E8%A7%A3%E6%9E%90/)，另外minddotout的博主在早些时候对LongAdder、AtomicLong进行了[性能测试](http://minddotout.wordpress.com/2013/05/11/java-8-concurrency-longadder/)，我们从结果中也可以直观的看到 LongAdder的强大。

**ForkJoinPool中增加新方法**

Java 8除了对Fork-Join框架做了优化外，也为ForkJoinPool增加了两个静态方法：getCommonPoolParallelism() 、commonPool()，Oleg Shelajev[在这篇博文中对Fork/Join做了详细的介绍](http://zeroturnaround.com/rebellabs/is-java-8-the-fastest-jvm-ever-performance-benchmarking-of-fork-join/)并做了测试以比较其在JDK7、JDK8中的性能。

**增加StampedLock类**

StampedLock是一种新型锁的实现，很可能在大多数场景都可以替代[ReentrantReadWriteLock](http://javadocs.techempower.com/jdk18/api/java/util/concurrent/locks/ReentrantReadWriteLock.html)。它为读写操作提供了三种模式：Writing、ReadingOptimistic、 Reading。

有关Java 8并发编程方面特性的详细介绍可从[官方文档](http://docs.oracle.com/javase/8/docs/api/index.html?java/util/concurrent/package-summary.html)获取。