System.Collections.Concurrent命名空间下包含了一些数据结构，这些数据结构具备可伸缩性，尽可能地避免锁，同时还能提供线程安全的访问。该集合使用原子比较和交换（Compare and Swap，简称CAS）操作。

Ps：什么是CAS操作：原子的比较和交换（Atomic Compare and Exchange）是一种用于实现线程安全的操作的机制。它通常用于解决多线程环境下的并发问题，确保对共享数据的操作是原子性的，避免竞态条件和数据不一致的问题。

下面列举这些数据结构：

ConcurrentQueue：先进先出，同队列。

图形用户界面, 应用程序

描述已自动生成

ConcurrentStack：先进后出，同栈。

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

表格

描述已自动生成

CurrentBag：一个支持重复元素的无序集合。

Ps：ConcurrentBag不提供索引访问或按照特定条件搜索元素的功能。

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

ConcurrentDictionary:线程安全的字典集合，对于读操作不使用锁，对写操作使用锁。该并发字典使用多个锁，在字典桶上实现了一个细粒度锁模型，使用参数concurrencyLevel可以在构造函数中定义锁的数量，这意味着预估的线程数量将并发地更新该字典。

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

图形用户界面, 应用程序, 表格

描述已自动生成

BlockingCollection：对IProducerConsumerCollection<T>实现的一个高级封装，提供阻塞和限制功能

表格

描述已自动生成

图形用户界面, 应用程序

描述已自动生成

图形用户界面, 应用程序

中度可信度描述已自动生成