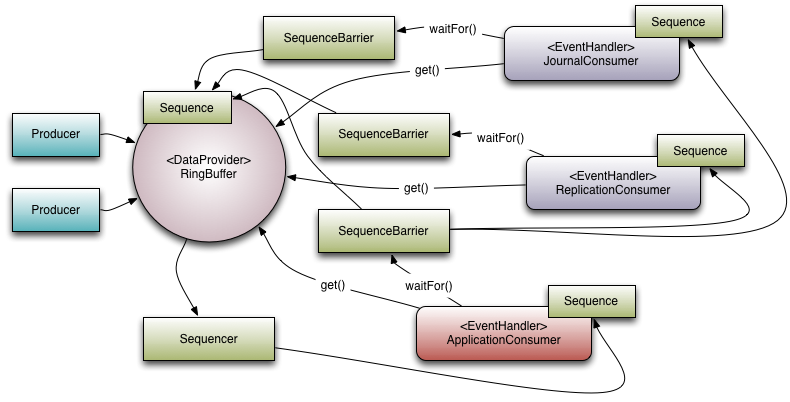
### 高性能无锁框架

* 环形队列：
  + Disruptor鼻祖：RingBuffer
  + Netty：环形队列：HashedWheelTimer 时间轮：定时任务
  + Netflix： hystrix断路器里面的核心实现：RollingNumber， 并发QPS统计
  + JUCTools: JUC的一个扩展
  + Canal：环形队列，用于做循环push binlog
  + Sentinel哨兵：LeapArray 滑动窗口实现
  + Skywalking：APM指标的统计
* Disruptor上手：
  + RingBuffer：固定长度，2的N次方；收尾相接的环形数组；
  + 扔芝麻和捡芝麻的小故事：
    - 消费者捡的比生产者扔的快，消费者就要停下来，等着生产者扔了新的芝麻过来，消费者继续消费；
    - RingBuffer(数组)的长度是有限的，当生产者到末尾的时候，会再次从数组的头部开始位置继续生产消息；这个时候可能会追上消费者，消费者如果还没有从那个地方捡走芝麻，这个时候生产者就要等待消费者捡走了芝麻以后，才能往里面扔芝麻；
* Disruptor架构图：
* 
* Disruptor核心的领域模型讲解：
  + RingBuffer：基于数组实现环形队列，也是创建sequencer与定义waitStrategy的入口；
  + Sequencer：单生产者和多生产者类型都需要创建的，根据ProducerType取值；作用就是用来协调生产者获取下一个可用序号、发布消息（发布下一个可用序号），包括唤醒wait状态的；
    - SingleProducerSequencer：处理单个生产者
    - MultiProducerSequencer：处理多个生产者
  + Sequence：
    - 做个填充缓存行（64bit）消除伪共享
      * @Contended：在JAVA8中有个@Contended的注解，可以自动填充缓存行。执行时，必须加上虚拟机参数-XX:-RestrictContended，@Contended注释才会生效。
      * 可用参考缓存行的文章：http://ifeve.com/disruptor-cacheline-padding/
    - 通过顺序递增的序号，来管理进行交换的数据（发布事件和消费数据）
  + Sequence Barrier：
    - 用于保持对RingBuffer的平衡，保持生产者和消费者之间的平衡关系；
  + waitStrategy：等待策略
    - 决定一个消费者如何等待生产者的关键等待策略配置
    - blocking、yielding、busySpin
  + Processor：
    - BatchEventProcessor：处理单生产者的核心工作处理线程
    - WorkProcessor：多生生产者的核心工作处理线程