- 缓冲优先的线程扩展策略
  - 背景
    - 传统线程池难点
      - 传统线程池的线程扩展速率无法根据业务需求进行灵活调整。
        - JDK线程池
          - 以 Java 线程池为例，]ava线程池(JDKTP)在核心线程已满后，只有当任务队列达到上限后才会尝试创建新线程。尽管此策略有助于节省资源，但由于线程扩展速率较低，会导致线程池在执行 I/0 密集型任务时存在并发度低的问题。
        - Tomcat线程池
          - Tomcat 线程池(TTP)通过取消任务缓冲机制以提高线程扩展速率，进而提升对 J/0 密集性任务的响应能力，但线程频繁地创建与销毁会造成资源浪费。
    - 设计思路
      - 缓冲扩展策略提供了一种灵活控制线程扩展速率的机制。该机制允许用户根据业务需要自主调整线程扩展速率，以在提升处理能力的同时避免资源浪费。
  - 引入
    - 缓冲因子BF
      - BF=1-TIO/Ttotal
        - TIO：任务中IO执行时间
        - Ttotal：任务执行总时间
          - Ttotal = Tcpu + Tio + Tblocked(阻塞时间)
            - CPU密集型：T_IO ≈ 0，T_blocked ≈ 0
            - I/O 密集型：T_blocked ≈ T_IO
        - 阻塞度
          - TIO/Ttotal
            - 接近1，IO密集型
              - 尽可能多创建线程
            - 4-6区间，混合密集型
            - 接近0，CPU密集型
              - 尽可能少创建线程
    - 缓冲阈值BT
      - BT=Qsize×BF
        - Qsize：队列长度
        - BF小，缓冲效果减弱，激进扩展--Tomcat
BF大，缓冲效果增强，保守扩展--JDKTP
  - 任务入队划分
    - 核心线程扩展阶段
    - 最大线程饱和阶段
      - 控制非核心线程的创建速率
      - 核心线程已满且任务队列中的任务≤BT，新任务直接入队
      - 任务数量 > BT且线程数小于最大线程数，创建新线程
  - 流程图实现思路
    - 假设任务队列为100，阈值为0.8。核心线程数被创建完，任务队列也达到80，那么就可以直接创建非核心线程数。如果达到最大最大线程数，此时任务队列还剩下20，继续往里面加，直到达到100，才触发拒绝策略。