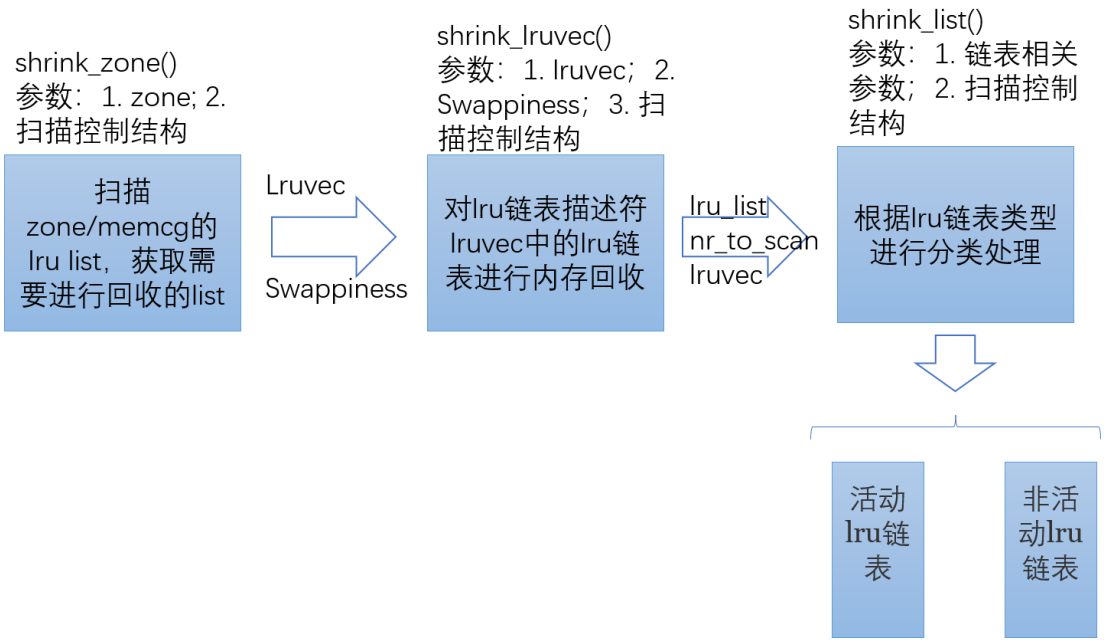
linux 内存回收流程

之前看到cgroup能够针对单个cgroup进行回收，以为是会有单独的处理函数。这周进行调研，读了一下内存回收的代码，发现以cgroup进行回收是以cgroup为单位进行lru链表的收集，后续回收过程则是统一接口。

目前感觉在这上面进行源码修改比较困难，结构比较复杂。

主要参考这篇博客linux内存源码分析 - 内存回收(整体流程) - tolimit - 博客园 (cnblogs.com)，介绍得很详细。虽然逻辑是以zone为单位，但是源码是完整的，里面有cgroup回收代码的注释，比较详细。

代码调用过程如下，由于代码比较复杂，目前还无法判断能否通过修改这部分代码实现并行后端。但**扫描控制结构scan_control是传到底层回收函数的**，感觉能够从这儿入手进行控制后端选择。



shrink_zone()

zone回收&memcg回收，最终只有一个函数入口shrink_zone()

该函数主要流程：

1. 从root_memcg开始遍历memcg
    1. 获取memcg的lru链表描述符lruvec
    2. 获取memcg的swapiness
    3. 调用shrink_lruvec()对此memcg的lru链表进行处理

2. 遍历完所有memcg后，检查是否还要对此zone再次进行内存回收。

shrink_lruvec()

核心函数即shrink_lruvec()，对对lru链表描述符lruvec中的lru链表进行内存回收，此lruvec有可能属于一个memcg，也可能是属于一个zone

- swappiness: 扫描匿名页的亲和力，其值越低，就扫描越少的匿名页，当为0时，基本不会扫描匿名页lru链表，除非针对整个zone进行内存回收时，此zone的所有文件页都释放了都不能达到高阀值，那就只对匿名页进行扫描
- sc: 扫描控制结构
- lruvec: lru链表描述符，里面有5个lru链表，活动/非活动匿名页lru链表，活动/非活动文件页lru链表，禁止换出页链表

该函数主要流程：

1. 调用get_scan_count()计算每个lru链表需要扫描的页框数量，保存到nr数组中；
2. 循环判断nr数组中是否还有lru链表没有扫描完成
    - 以活动匿名页lru链表、非活动匿名页lru链表、活动文件页lru链表、非活动文件页lru链表的顺序作为一轮扫描，每次每个lru链表扫描32个页框，并且在nr数组中减去lru链表对应扫描的数量；
    - 一轮扫描结束后判断是否回收到了足够页框，没有回收到足够页框则跳到 2 继续循环判断nr数组；
    - 已经回收到了足够页框，当nr数组有剩余时，判断是否要对lru链表继续扫描，如果要继续扫描，则跳到 2
3. 如果非活动匿名页lru链表中页数量太少，则对活动匿名页进行一个32个页框的扫描；
4. 如果太多脏页正在进行回写，则睡眠100ms

后续代码的阅读还没有完成