DeepVM 内存管理方案

内存池分配器设计

滕泓宇 2020/12/20

大纲

- 设计目标
- 数据结构
- 加速机制
- 全局元信息
- Q&A

设计目标

核心目标：确保硬件可靠情况下内存池长期可用

- 设计目标
  - 支持不同大小的内存池，初始化时设定
  - 支持快速分配释放内存
  - 内存池大小不可变
- 指标权衡
  - **限定单线程**
  - 内存大小有限：单池上限 4GB，寻址使用 32 bit
  - 读写权限管理交由上级模块完成

数据结构

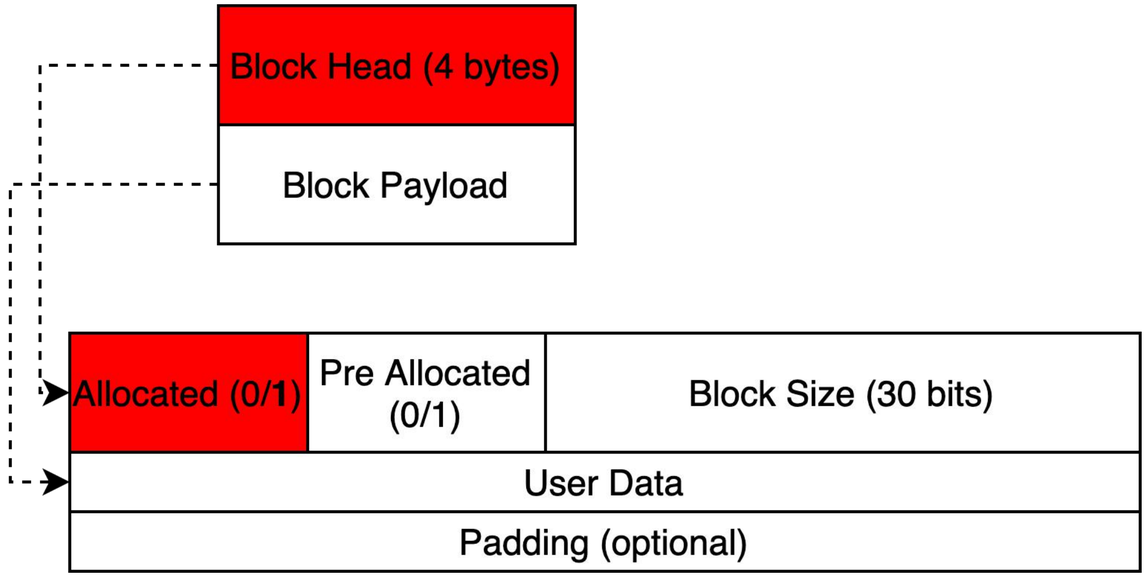
基于块（block）的内存管理

- 将连续的内存池划分为一些连续的块进行管理
- 块有两种状态，分别是空闲状态（Free Block）和已分配状态（Allocated Block）
  - 空闲块：管理的主体，需要保证在分配时可以高效地找到
  - 已分配块：保证对上级应用可用
    - 上传对应地址
    - 保证在上级应用返还前不受干扰

数据结构

内存块结构

- 对外不可见的块级元信息 Head
  - 已分配标识 Allocated Flag
  - 前驱块分配标识 Pre Allocated Flag
  - 块大小 Size / 8 保存
    - 地址八字节对齐，节约 3bit
- 对外可见的内存块数据段 Payload

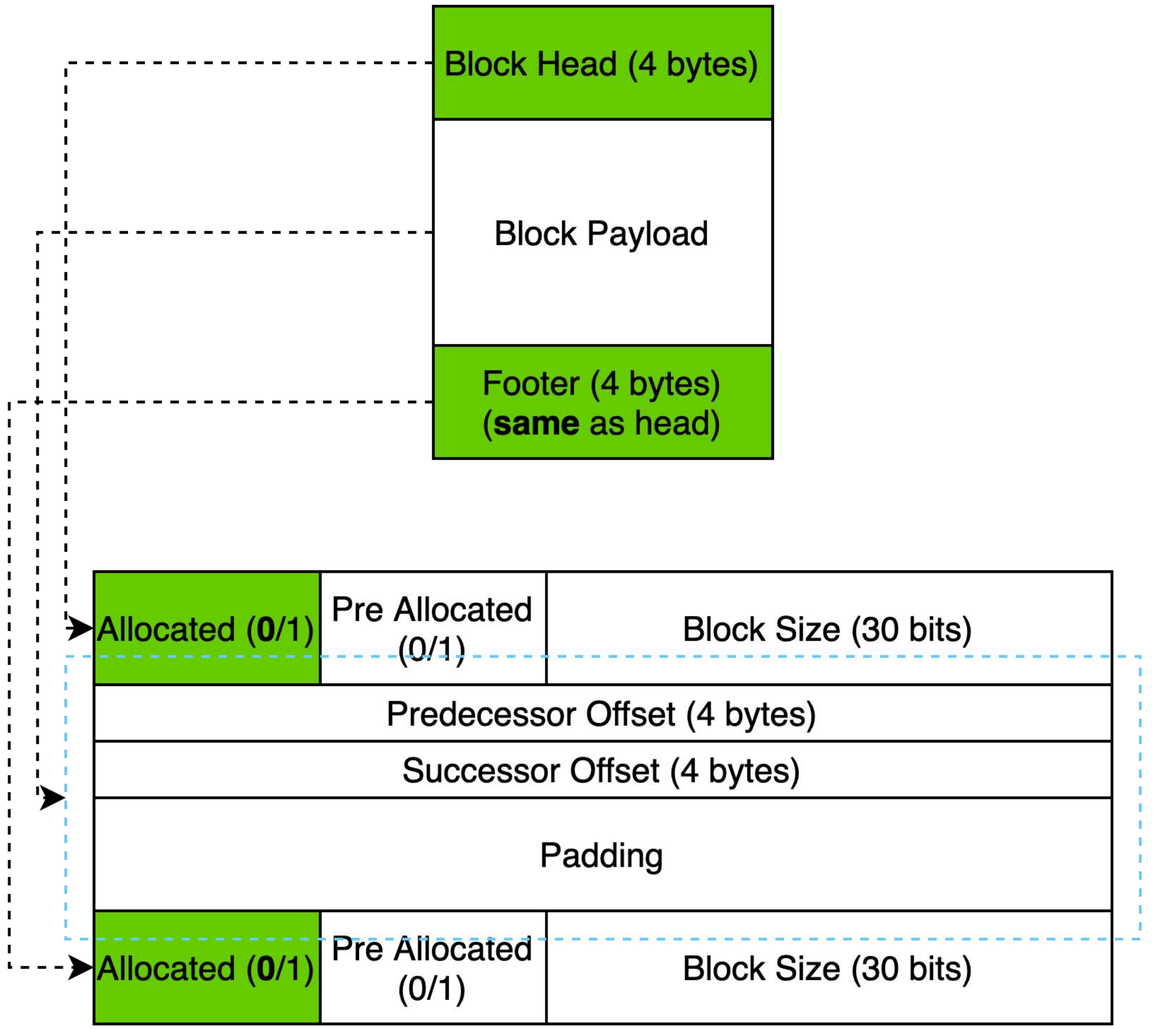


已分配块的结构。
注意所有已分配块结构都是一样的，特殊结构只存在于空闲块中。

数据结构

空闲块的特殊结构

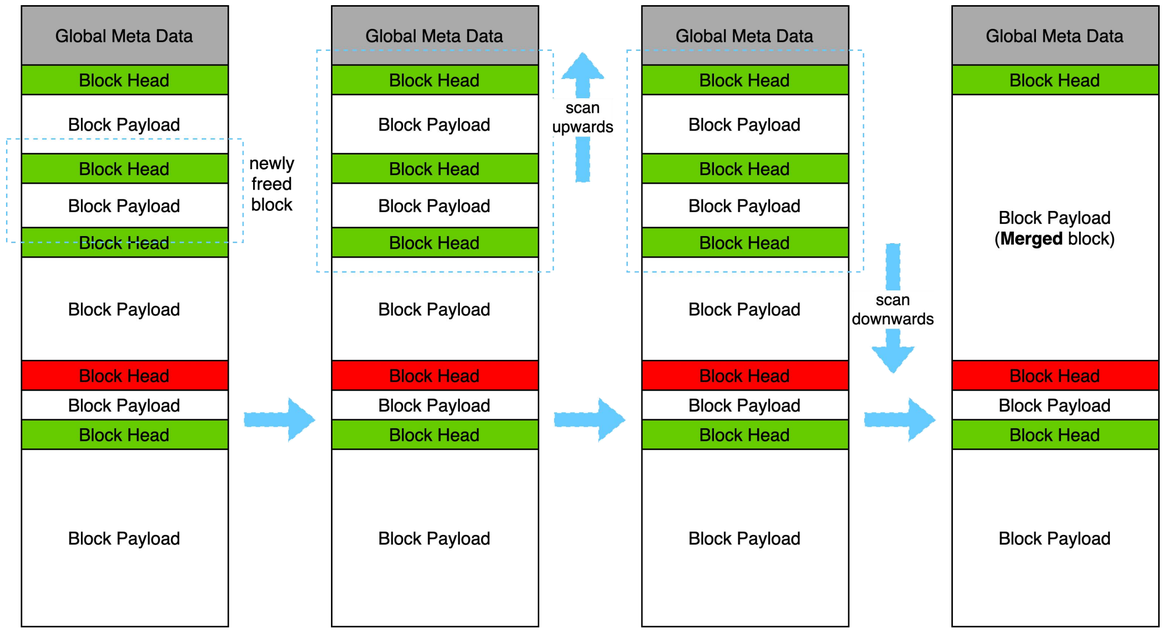
- 块级元信息 Head
- 管理用信息
  - 前驱块地址偏移 Predecessor
  - 后继块地址偏移 Successor（可选）
- 块级元信息 Footer，与 Head 相同
  - 为空闲块合并提供必要信息（可选）



数据结构

空闲块的合并

- 释放（Free）部分种类已占用块时进行
- 向上检查上一块是否为空
  - 检查自己的 前驱块分配标识 Pre Allocated Flag
- 向下检查下一块是否为空
  - 使用自己的体积 Size 和地址找到下一块头部/元信息地址，读取其已分配标识
- 清空新块数据段 Payload
- 更新新块的头部/元信息，完成合并



加速机制

针对两种使用场景使用不同设计

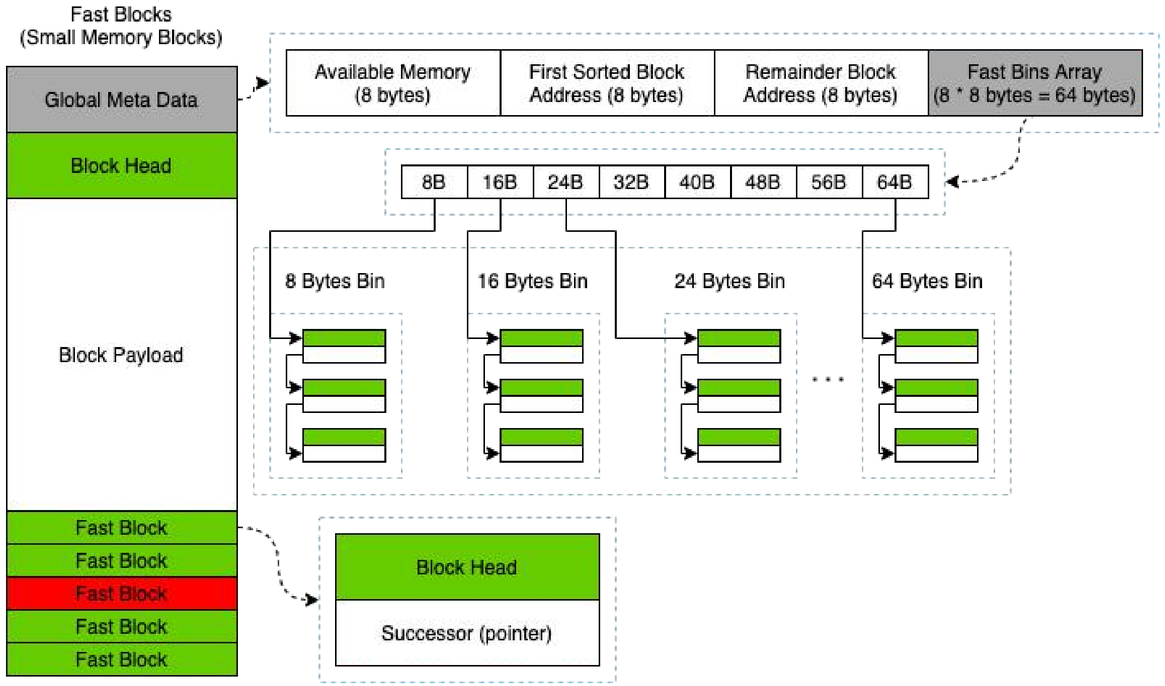
- 小内存块（≤ 64 bytes）
  - 快速分配快速回收
  - 体积较为固定
  - 使用散列表，按体积索引
    - 闭地址法 / 外接链表
- 大内存块（≥ 72 bytes）
  - 可以忍受一定效率损失
  - 体积变化较大
  - 使用跳表 Skiplist 索引块体积
    - 实现较为简单
    - 外接链表连接空闲块

中间跳过的 8 bytes 是因为内存地址按八字节对齐，不存在体积处于 64 - 72 bytes 之间的块。
小内存块上限 64 bytes 是大内存块的设计需求，可以增加不能减少。

加速机制

小内存块 Fast Blocks

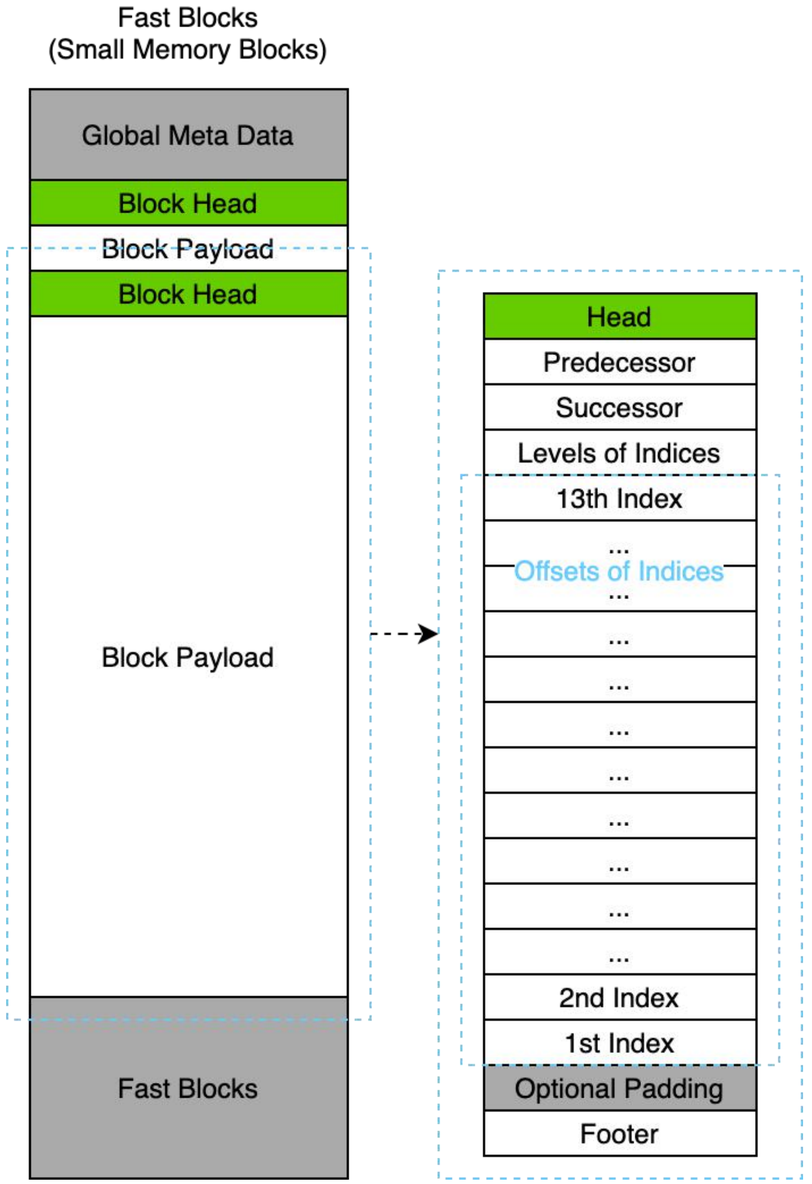
- 将同样大小的空闲内存块链接在同一条单向链表上
  - 在每个块的数据段里直接保存下一个块的地址
- 只在表头做插入删除操作
- 使用一个数组来索引所有的链表头
- 不进行合并，不保留 Footer



加速机制

大内存块 Sorted Blocks

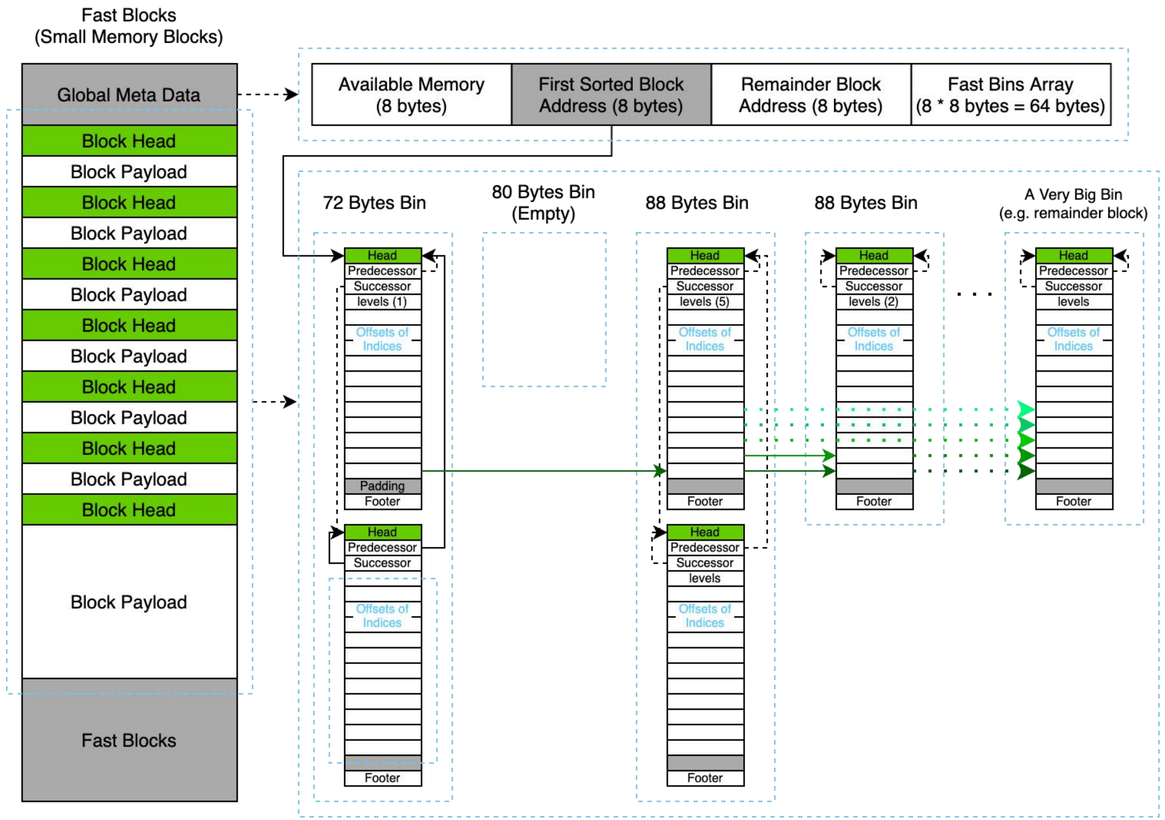
- 将同样大小的空闲内存块链接在同一条双向链表上
  - 在每个块的数据段里保存下一个块的地址偏移量
- 只在表头/表头后插入删除
- 使用一个跳表来索引所有的链表头
  - 块内内嵌跳表所需所有索引值
  - 依体积升序排列



加速机制

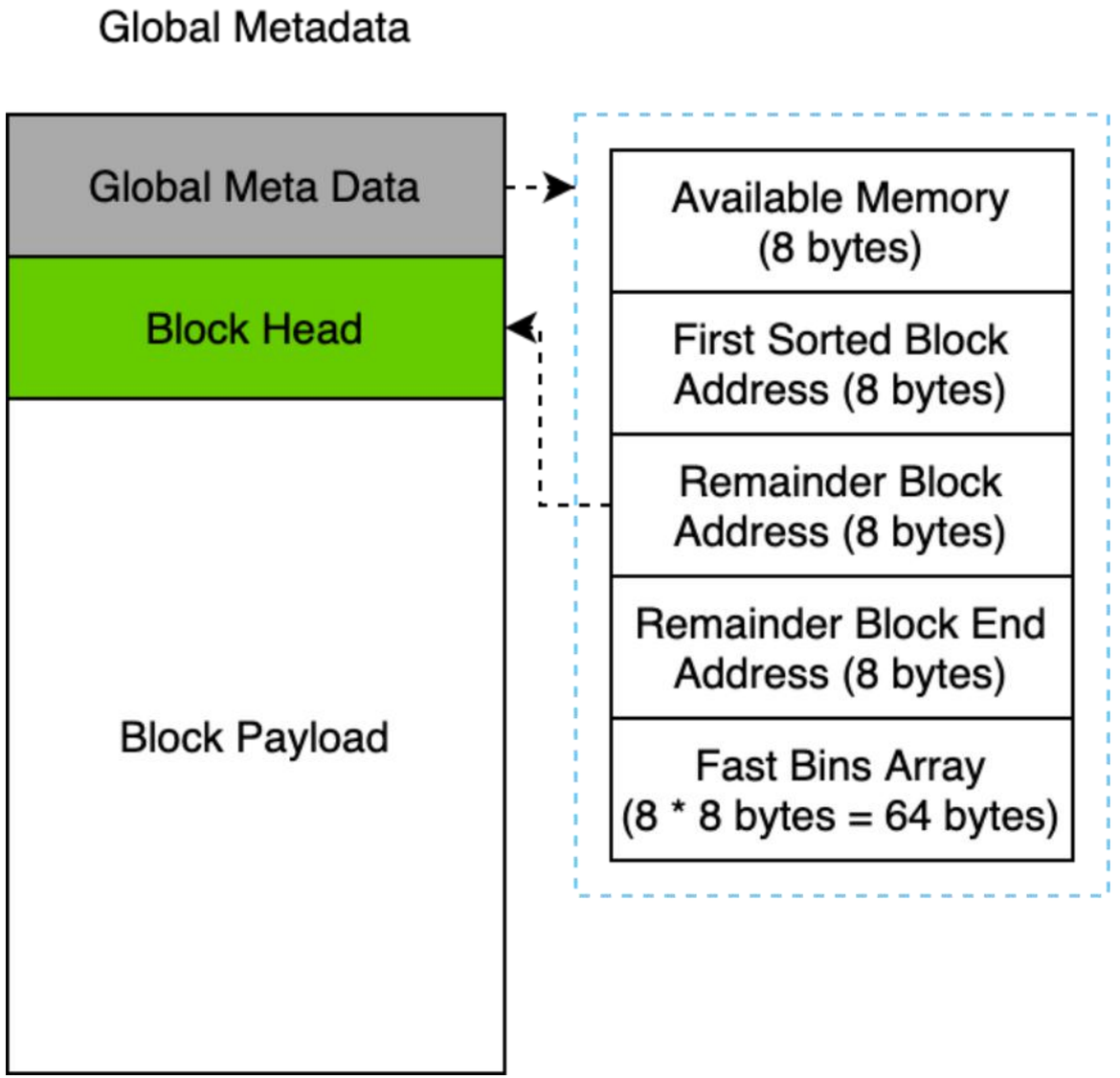
跳表（Skip List）结构

- 跳表在每个节点建立至多13级索引，每一级索引都保存同级索引中下一个节点的偏移量
- 最低级索引即依序连接所有节点
- 每个节点建立的索引层数是随机的
- 在 4GB 内存条件下时间复杂度大约是，常数 ⩽ 5
  - 平均情况下单次检索至多 60 个节点



全局元信息

- 全局空闲内存
  - 不包括任何已填写元信息的块头
- 第一个大内存块地址
  - 初始化内存池时在元信息块后直接分配，大小为最小值，永远为跳表头
- 剩余块头部地址和尾部后的地址
  - 即 End address 为不可访问地址
- 小内存块索引数组



展望

局限与未来工作目标

- 缺少测试
- 以 4GB 32位寻址为底本编写，需要适应 64位
- 对零散内存的整合能力偏弱
- 未实现内存池相关信息的查询功能

参考资料

glibc 堆内存管理 & 跳表

- Linux堆内存管理深入分析（上）(https://introspelliam.github.io/2017/09/10/Linux堆内存管理深入分析（上）/)
- Linux堆内存管理深入分析下(https://introspelliam.github.io/2017/09/15/pwn/Linux堆内存管理深入分析下/)
- Skip List--跳表（全网最详细的跳表文章没有之一）(https://www.jianshu.com/p/9d8296562806)
- Redis/.../t_zset (Implementation of Skip-List)(https://github.com/antirez/redis/blob/unstable/src/t_zset.c)
- 随机数生成器 xoroshiro128+ (http://prng.di.unimi.it)