Lifetime-Leveling LSM-Tree Compaction for ZNS SSD

汇报人：马绍博　　汇报日期：2022-12-8

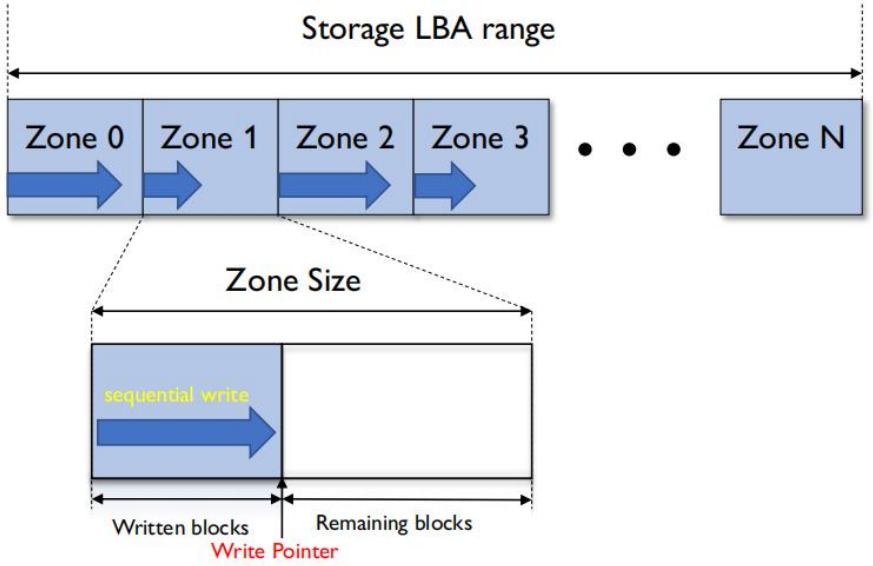
背景知识

Zoned Namespace（ZNS）

·逻辑地址空间被分成固定大小的区域

·每块区域必须被顺序写，且重用前需重置

·无需SSD内部的垃圾回收机制



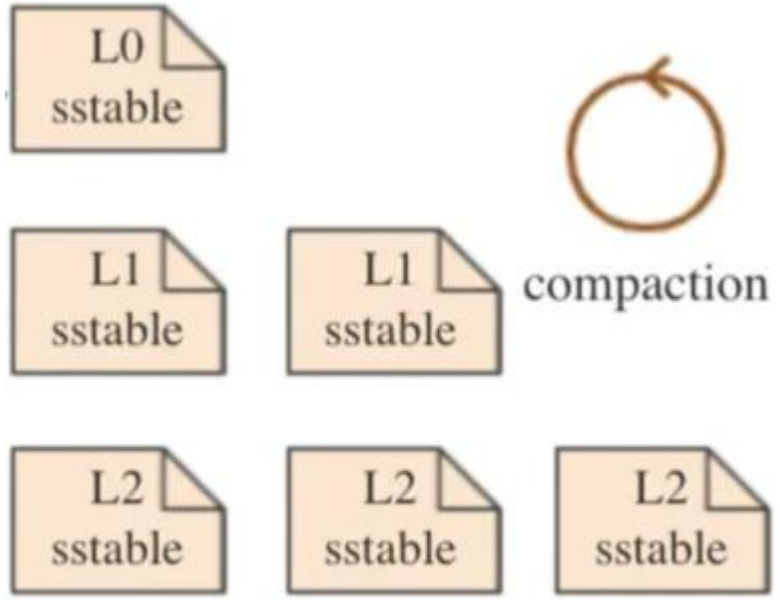
背景知识

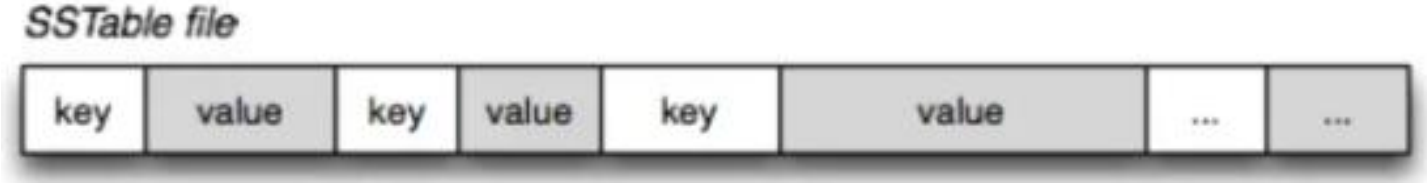
LSM-tree

·LSM是一种分层数据结构，由SST组成

·除L0外，剩下层由上层SST压缩产生

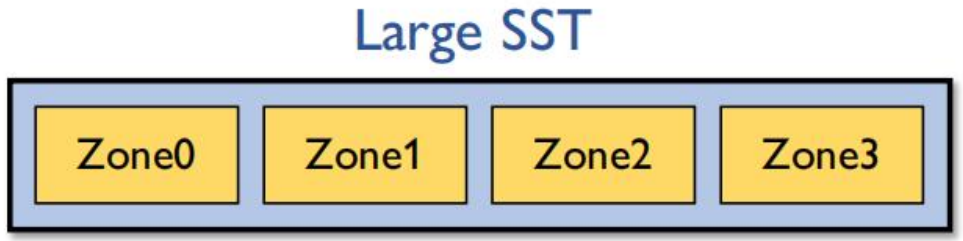
·每层均有阈值限制SST的数量和容量

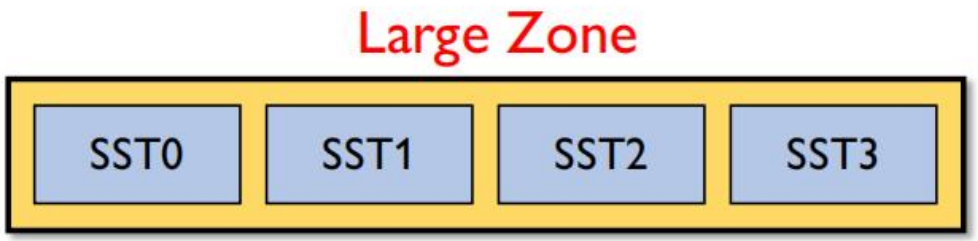




背景知识

LSM树应用于ZNS



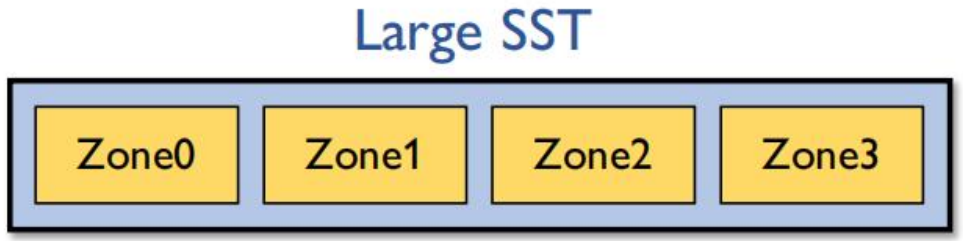


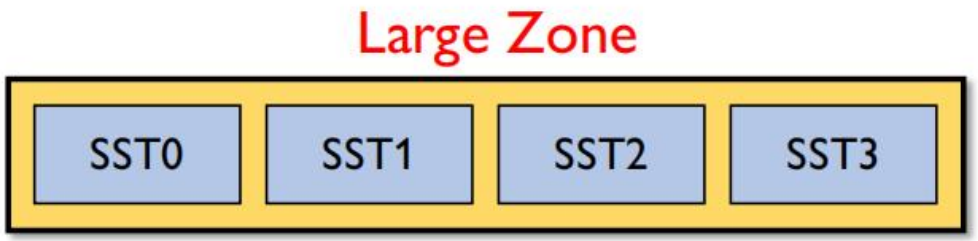
- 高压缩开销
- 高内存压力

- 低压缩开销
- 空间放大

背景知识

LSM树应用于ZNS





- 高压缩开销
- 高内存压力

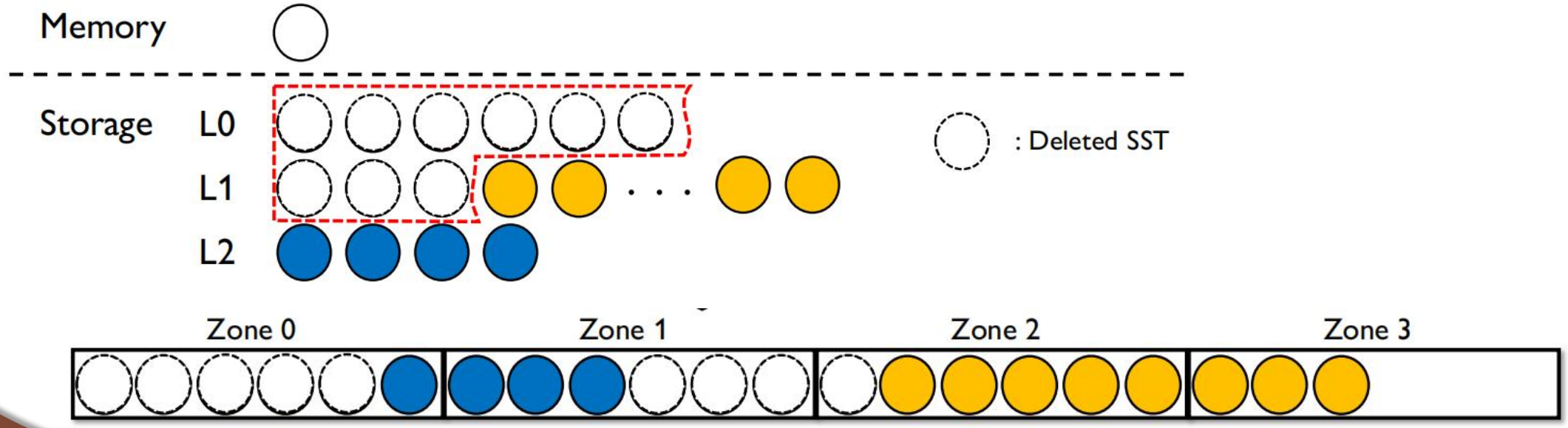
- 低压缩开销
- 空间放大

背景知识

1）读放大:读取数据时实际读取的数据量大于真正的数据量。

2）写放大:写入数据时实际写入的数据量大于真正的数据量。

3）空间放大:数据实际占用的磁盘空间比数据的真正大小更多。



ZNS-LSM算法

1.决定压缩层数，初始化一组SSTs作为合并集

2.将压缩指针CP指向第一个SST，设置压缩窗口为其首尾键值

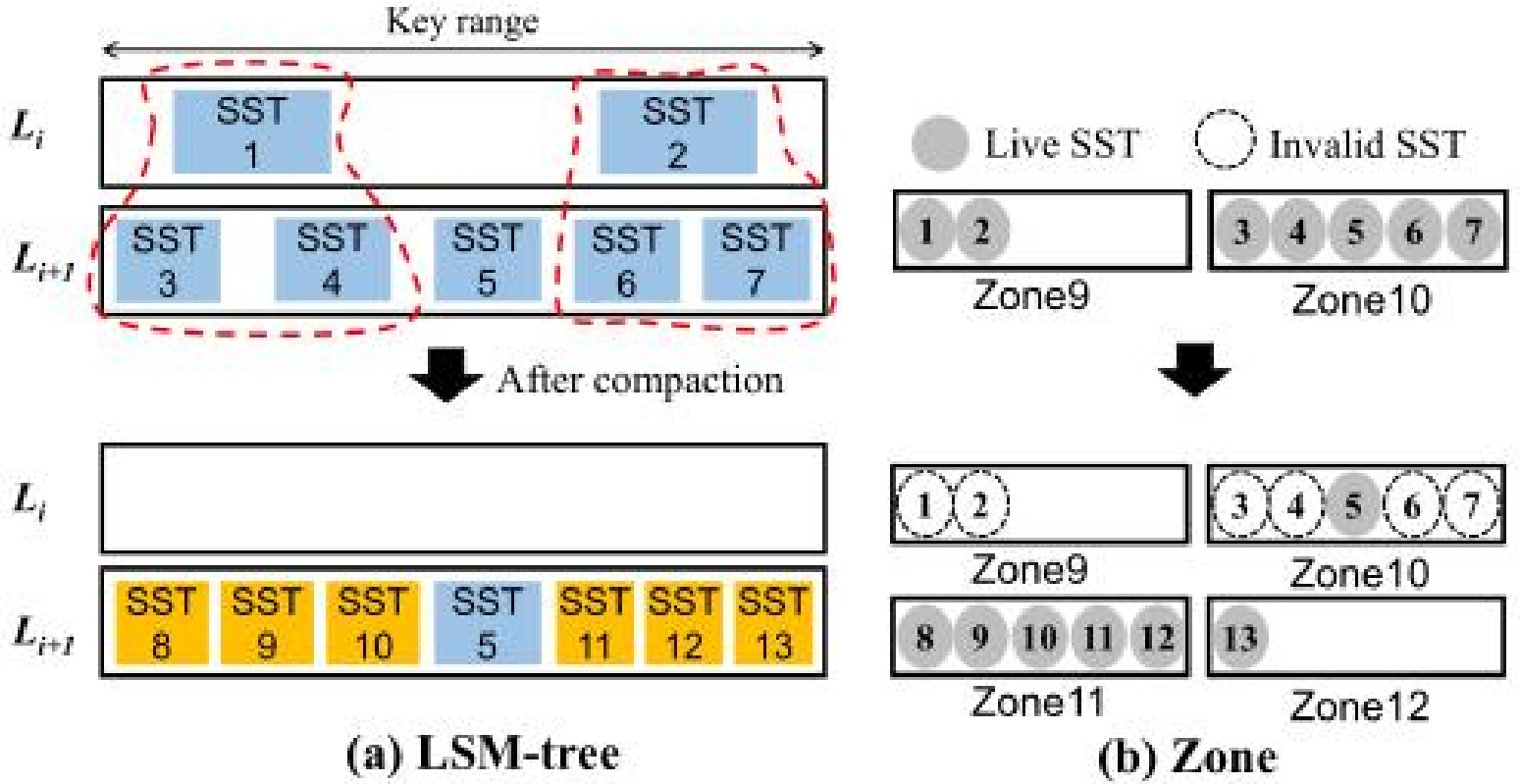
3.对于下层在键值范围内的SST，将其插入合并集M，并更新压缩窗口

4.将被压缩窗口完全覆盖的上层SST全部放入M中

5.合并M并将其放入下层，删除被合并的SSTs

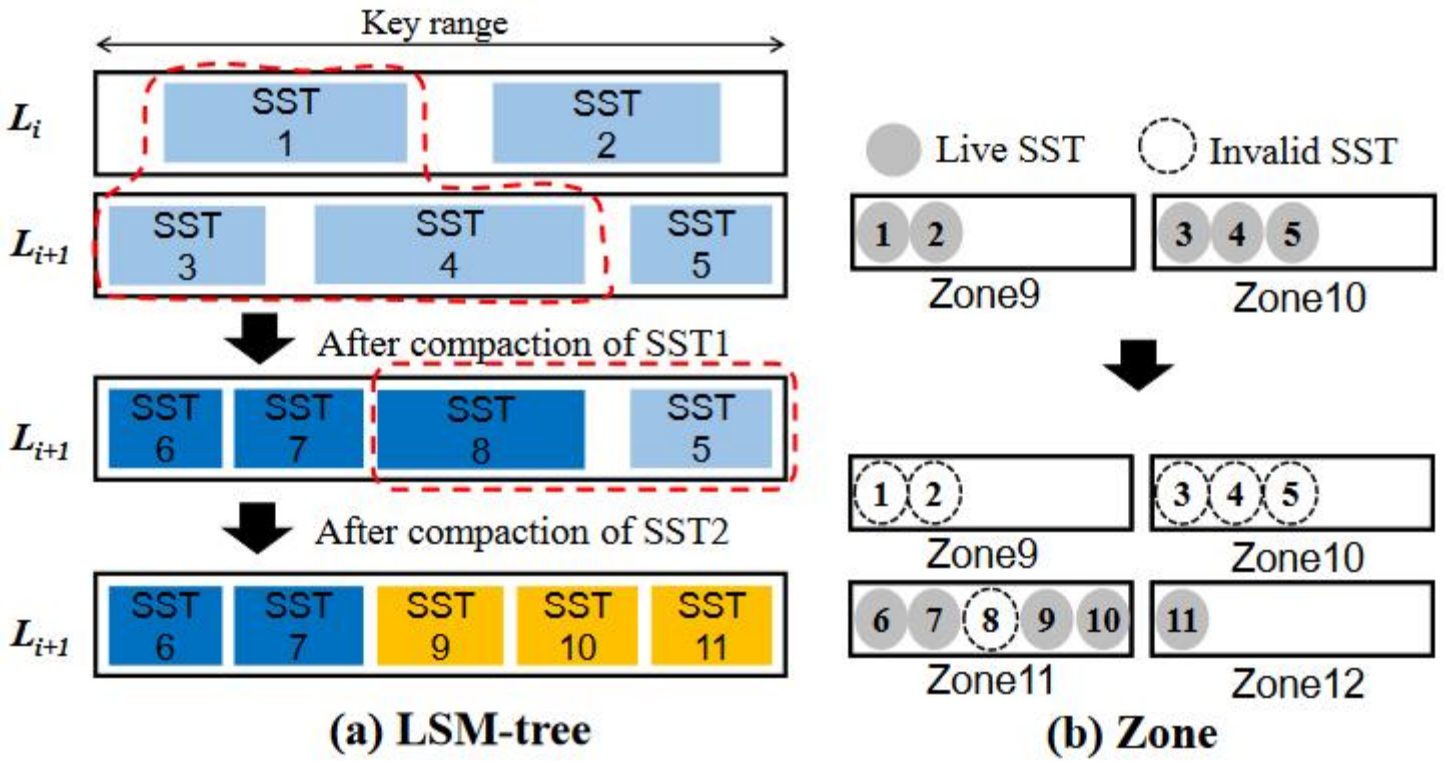
6.将压缩指针置于下一个SST对象

ZNS-LSM算法



Long-lived SST

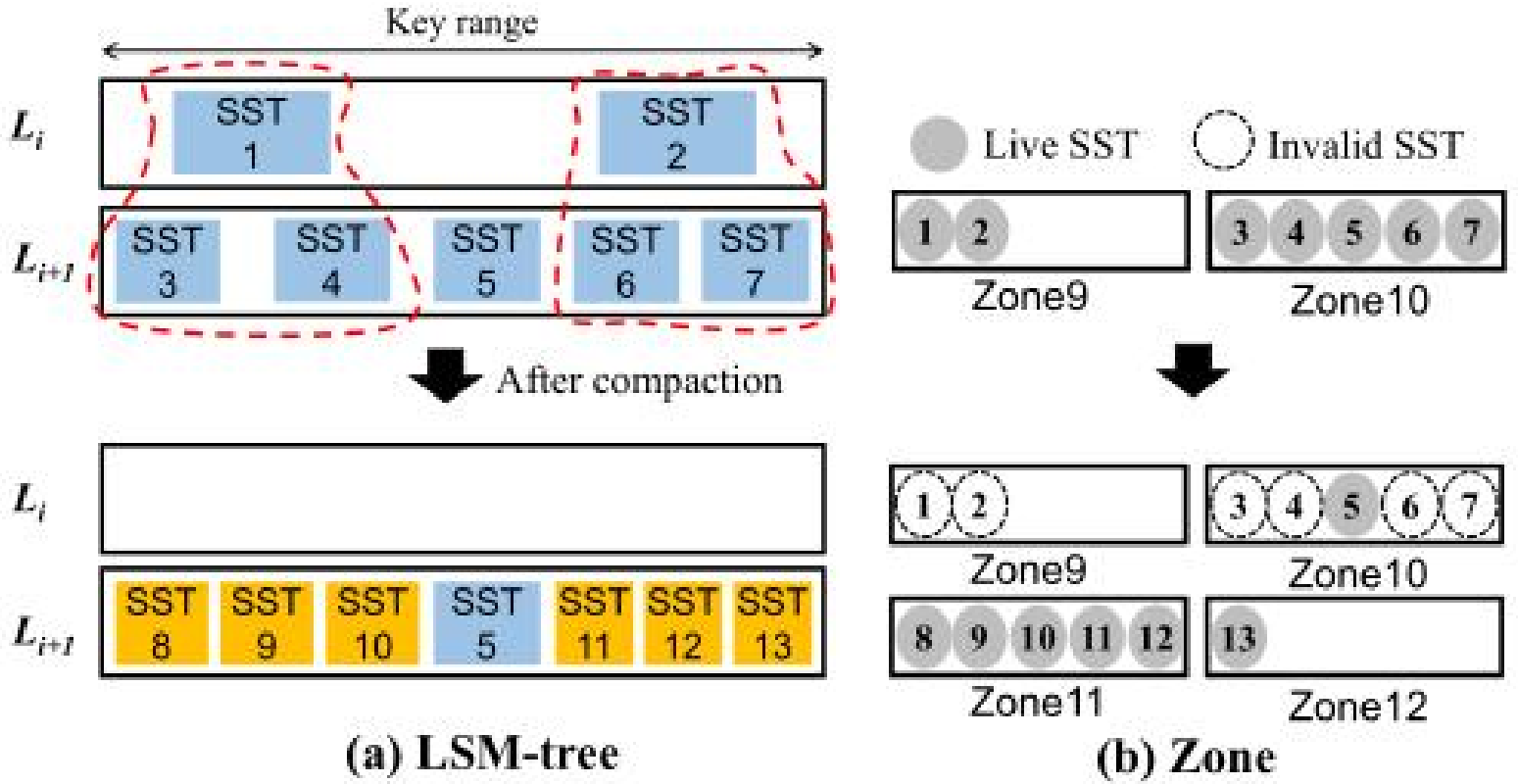
ZNS-LSM算法



Short-lived SST

LL Compaction

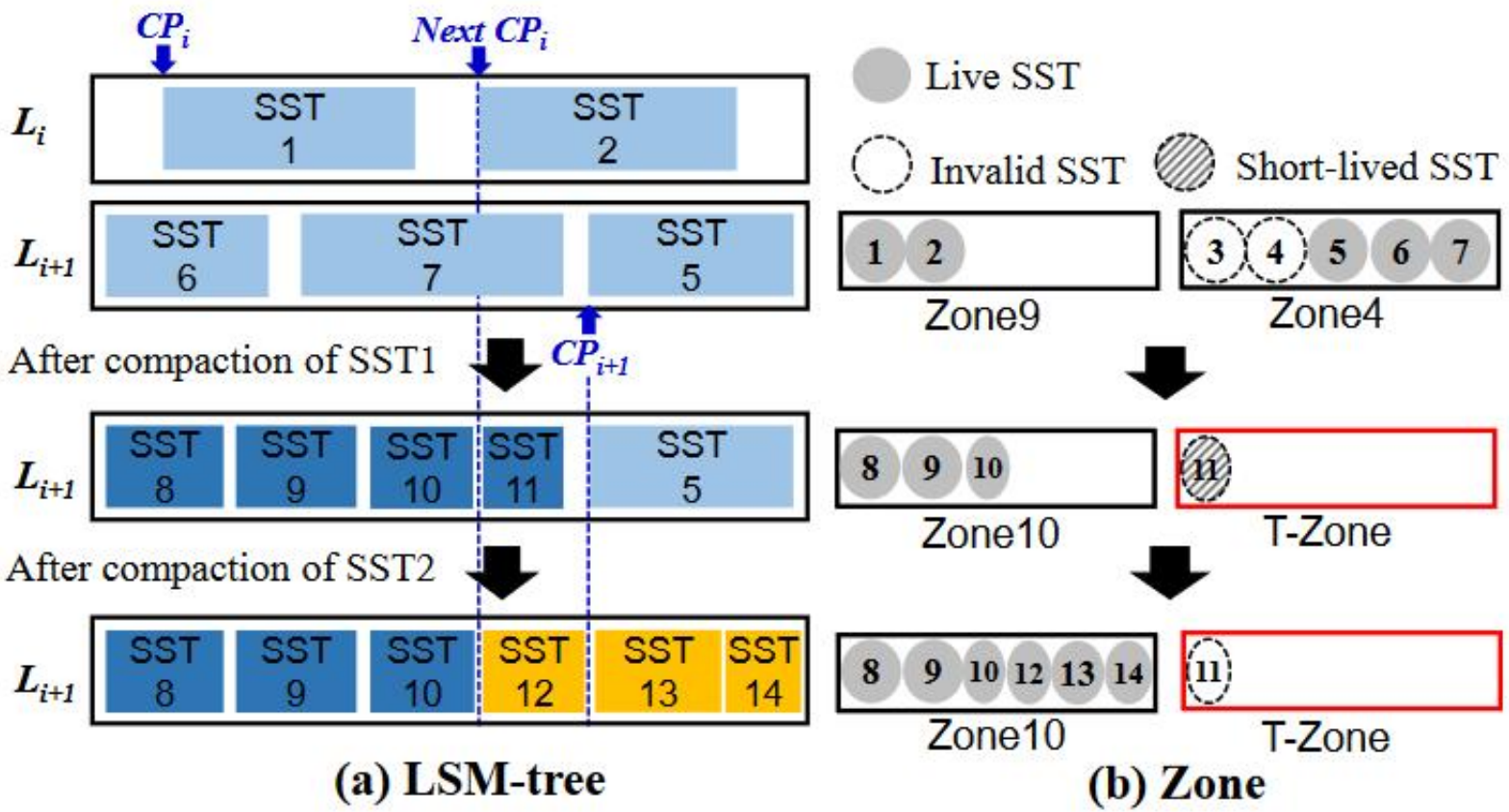
核心思想：使同区域SST生命周期尽可能一致



增加压缩成本

LL Compaction

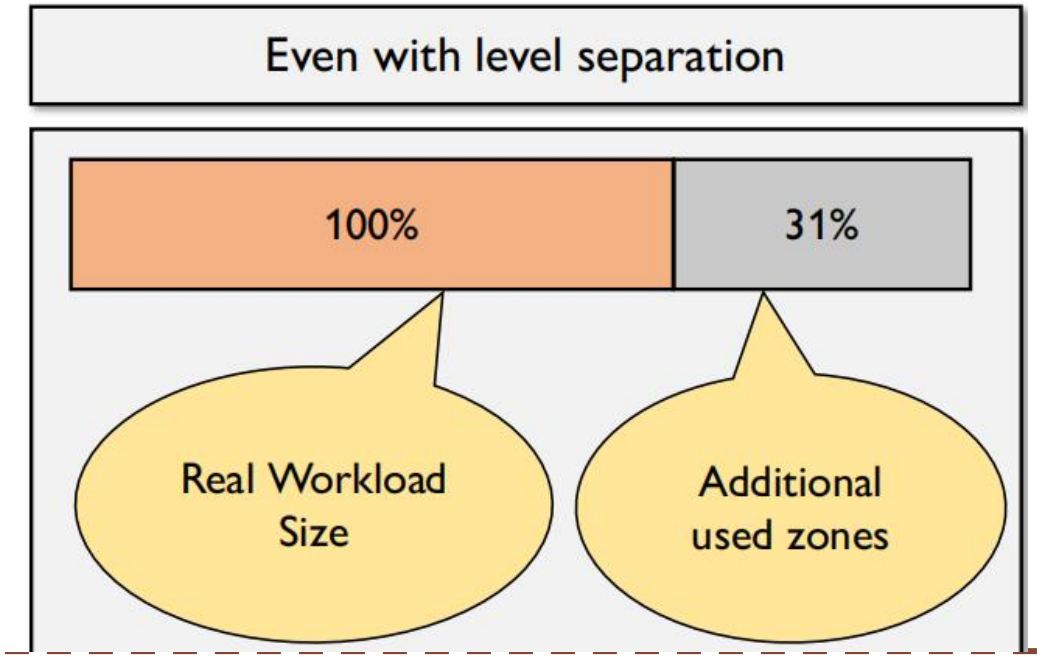
核心思想：使同区域SST生命周期尽可能一致



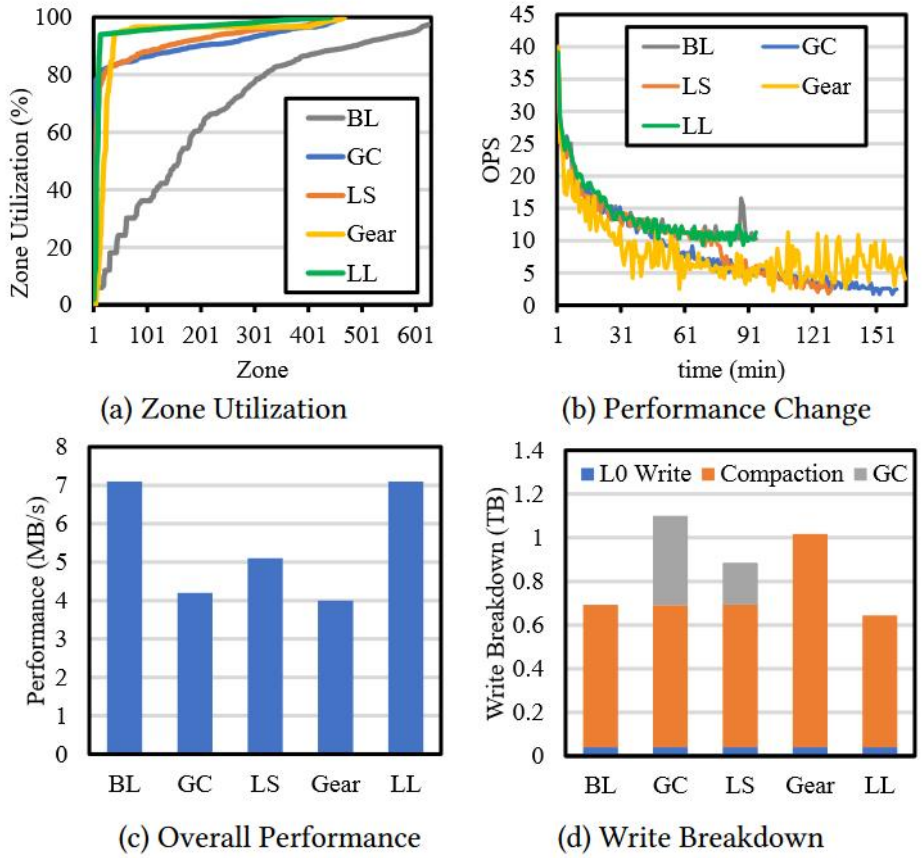
减少压缩成本

LL Compaction

工作负载总共写入16.5GB的数据，因此如果没有发生空间放大，则使用264个分区。然而，实际分配的区域数量为348个。在我们的分析中，其中81个区域(31%的已分配区域)由short-live sst占用，3个区域由long-live sst占用。



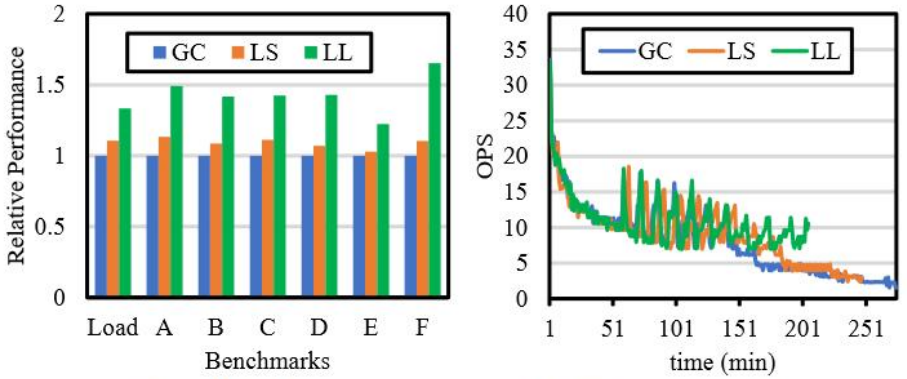
性能评估



(a) Zone Utilization (b) Performance Change

(c) Overall Performance (d) Write Breakdown

Figure 4: Fill-random workload



(a) Overall Performance (b) Performance Change

Figure 5: YCSB workload

总结

·现有基于LSM的ZNS算法会造成空间放大

·LL压缩算法在不引入垃圾回收机制的前提下缓解空间放大

·LL压缩算法无法使用优先级驱动的压缩算法

谢谢大家