Bucket Cache:A solution about CMS,Heap Fragment and Big Cache on HBASE

HBase@Ali

zjusch@163.com

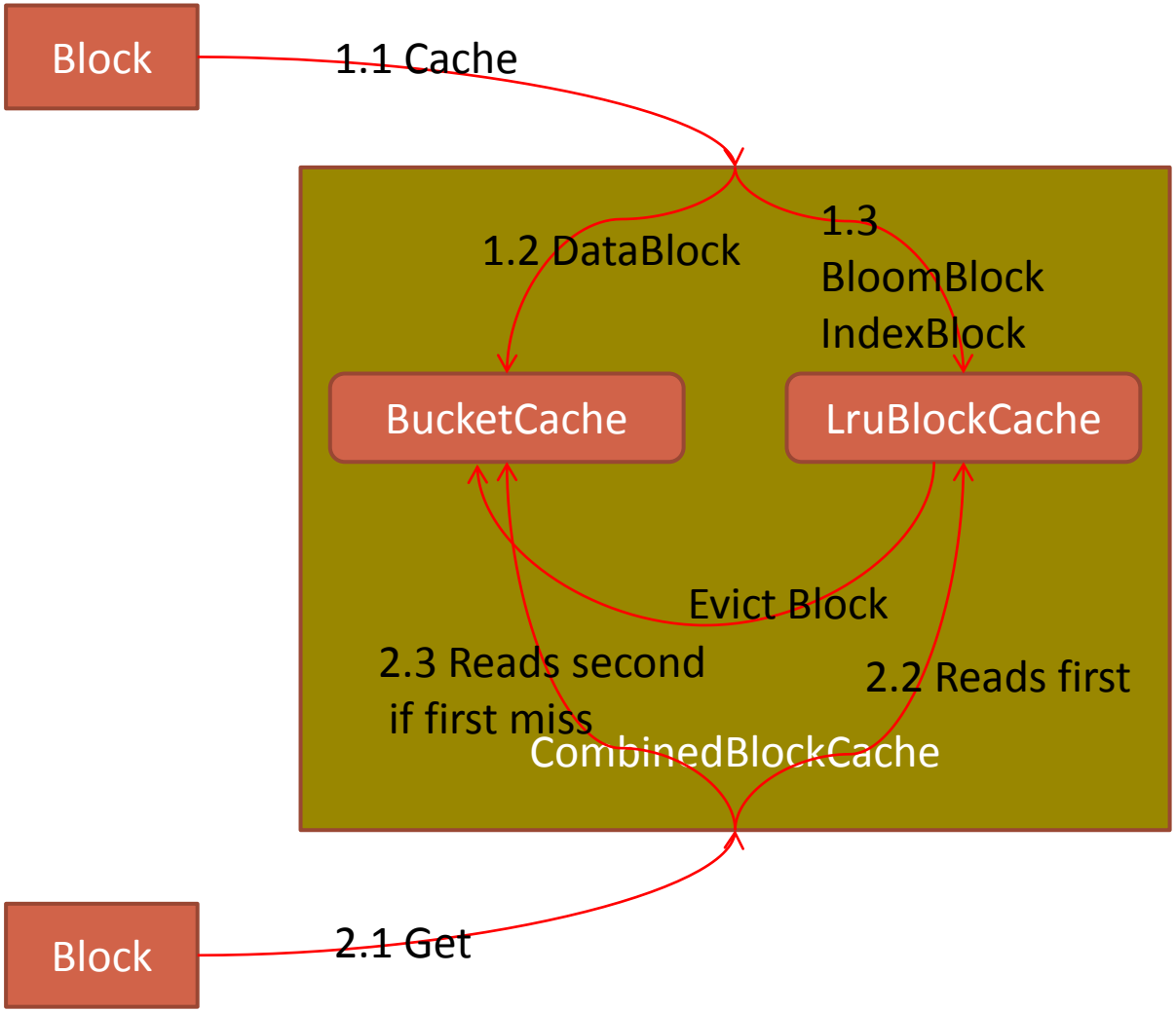
What's Bucket Cache

- An implementation of block cache like LruBlockCache
- Self manage blocks' storage position through Bucket Allocator
- The cached blocks could be stored in the memory or file system
- Bucket Cache could be used as a mainly block cache(see CombinedBlockCache), combined with LruBlockCache to greatly decrease CMS and heap fragment by GC.
- BucketCache also could be used as a secondary cache(e.g. using Fusionio to store block) to enlarge cache space

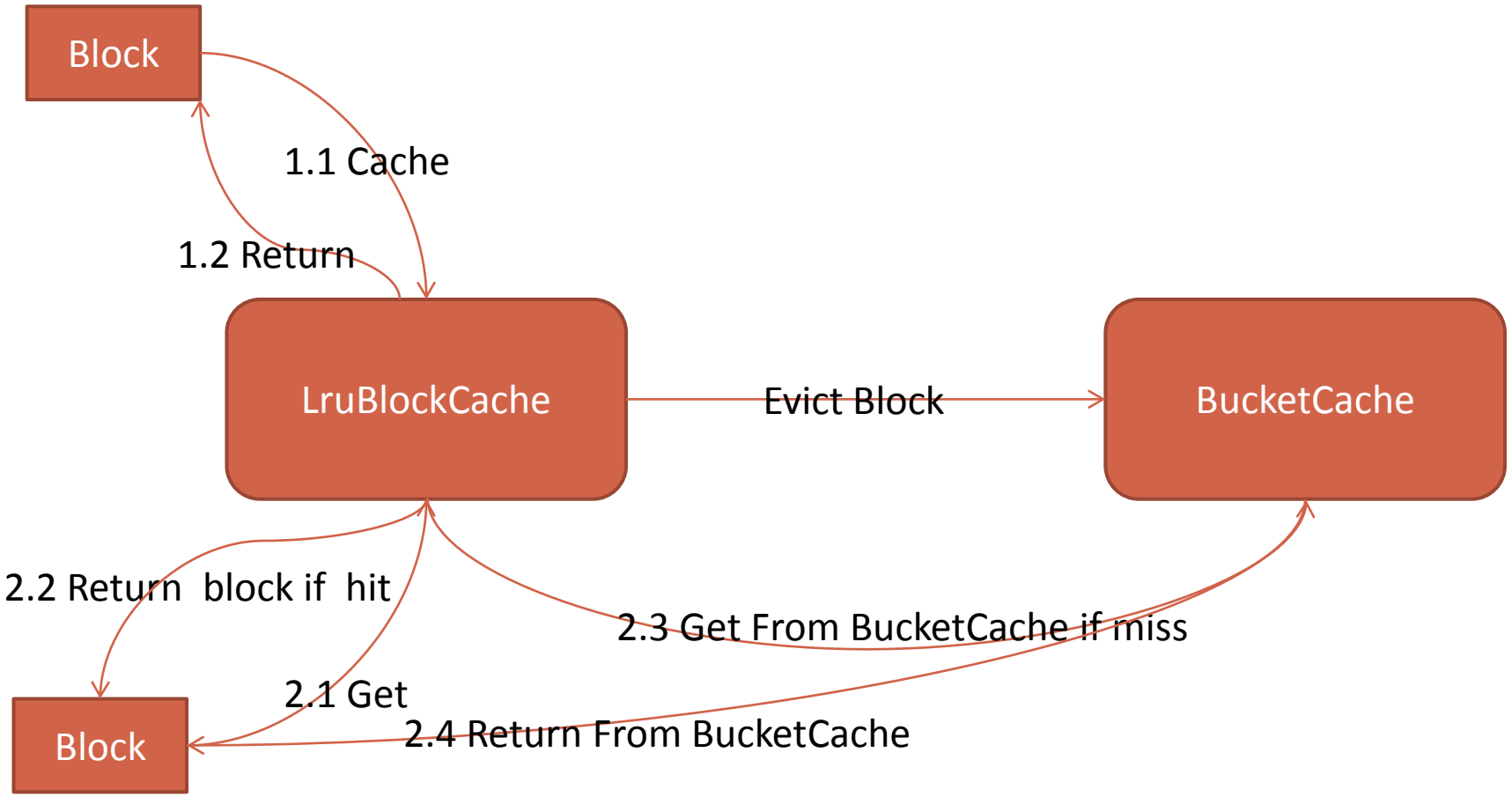
How to use

- Use it as a mainly cache , combined with LruBlockCache
  - “hbase.bucketcache.ioengine” “heap”
  - “hbase.bucketcache.size” 0.4(size for bucket cache, 0.4 is a percentage of max heap)
  - Optional configurations
    - “hbase.bucketcache.combinedcache.percentage” 0.9f as default(Percentage of BucketCache in the CombinedCache)
- Use it as a secondary cache
  - “hbase.bucketcache.ioengine” “file:/disk1/hbase/cache.data”(The file path where to store the block data)
  - “hbase.bucketcache.size” 10*1024 (size for bucket cache, unit is MB, so 10*1024 means 10GB)
  - “hbase.bucketcache.combinedcache “ false
  - Optional configurations
    - “hbase.bucketcache.persistent.path” “file:/disk1/hbase/cache.meta”(The file path where to store the meta data of bucket cache, use for restoring cache when restart)

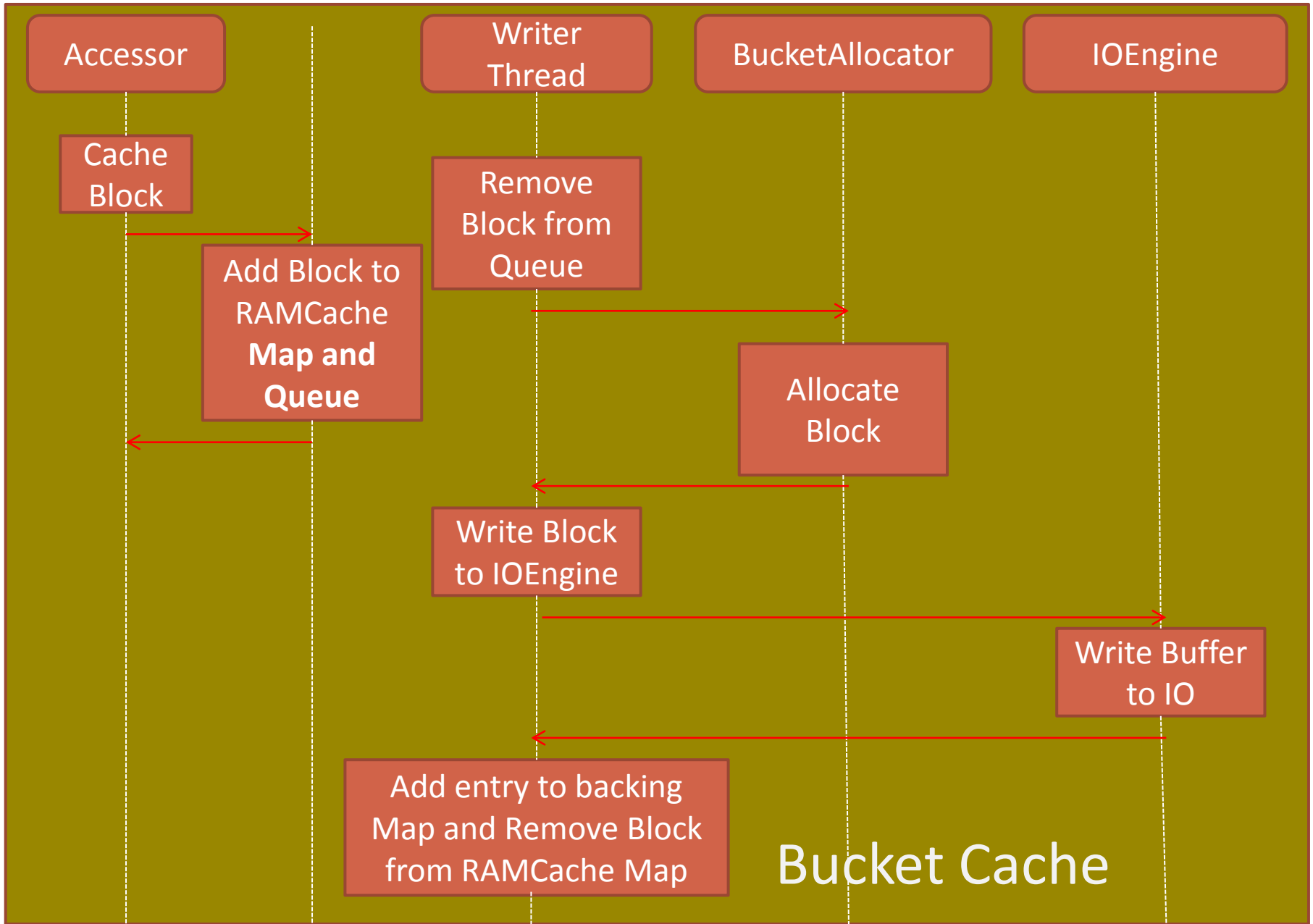
Process of First Usage



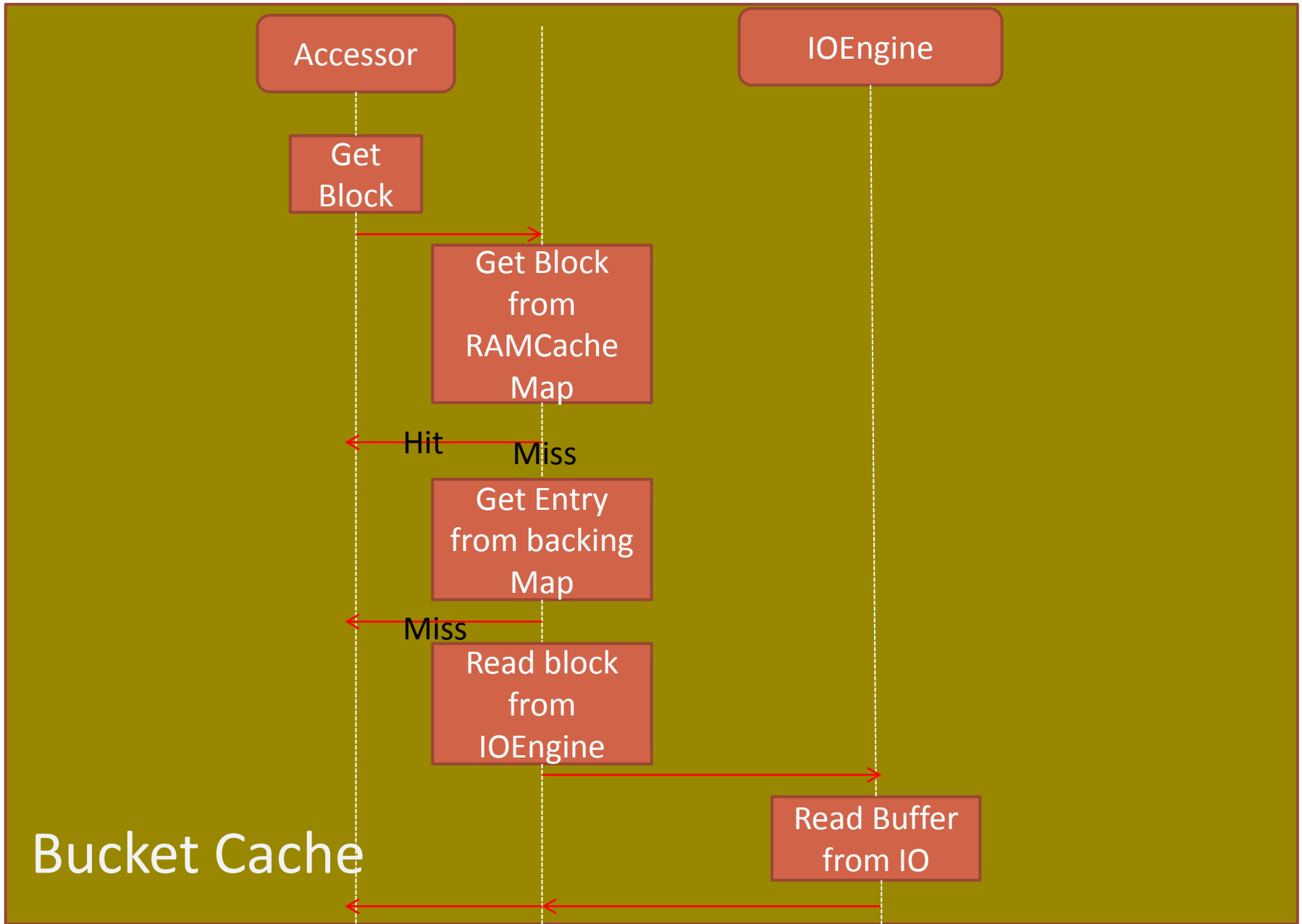
Process of Second Usage



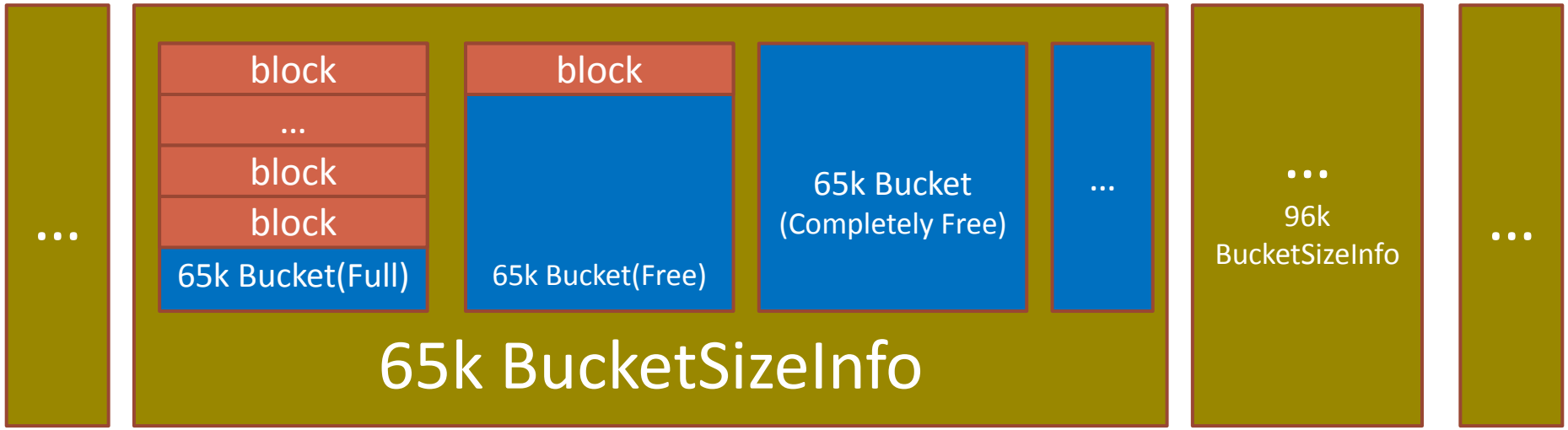
Cache block in Bucket Cache



Get block in Bucket Cache

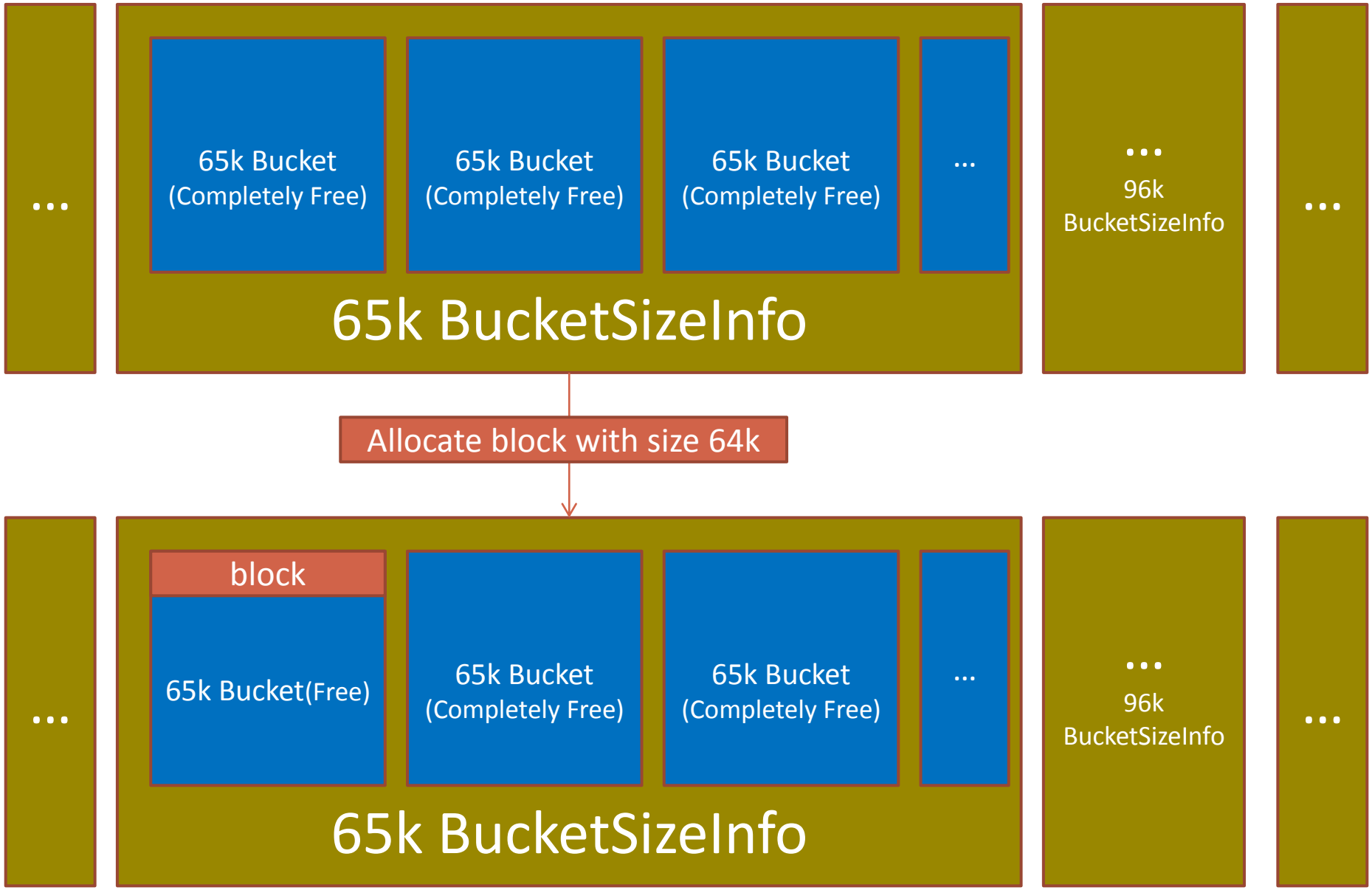


Bucket Organization in Bucket Allocator

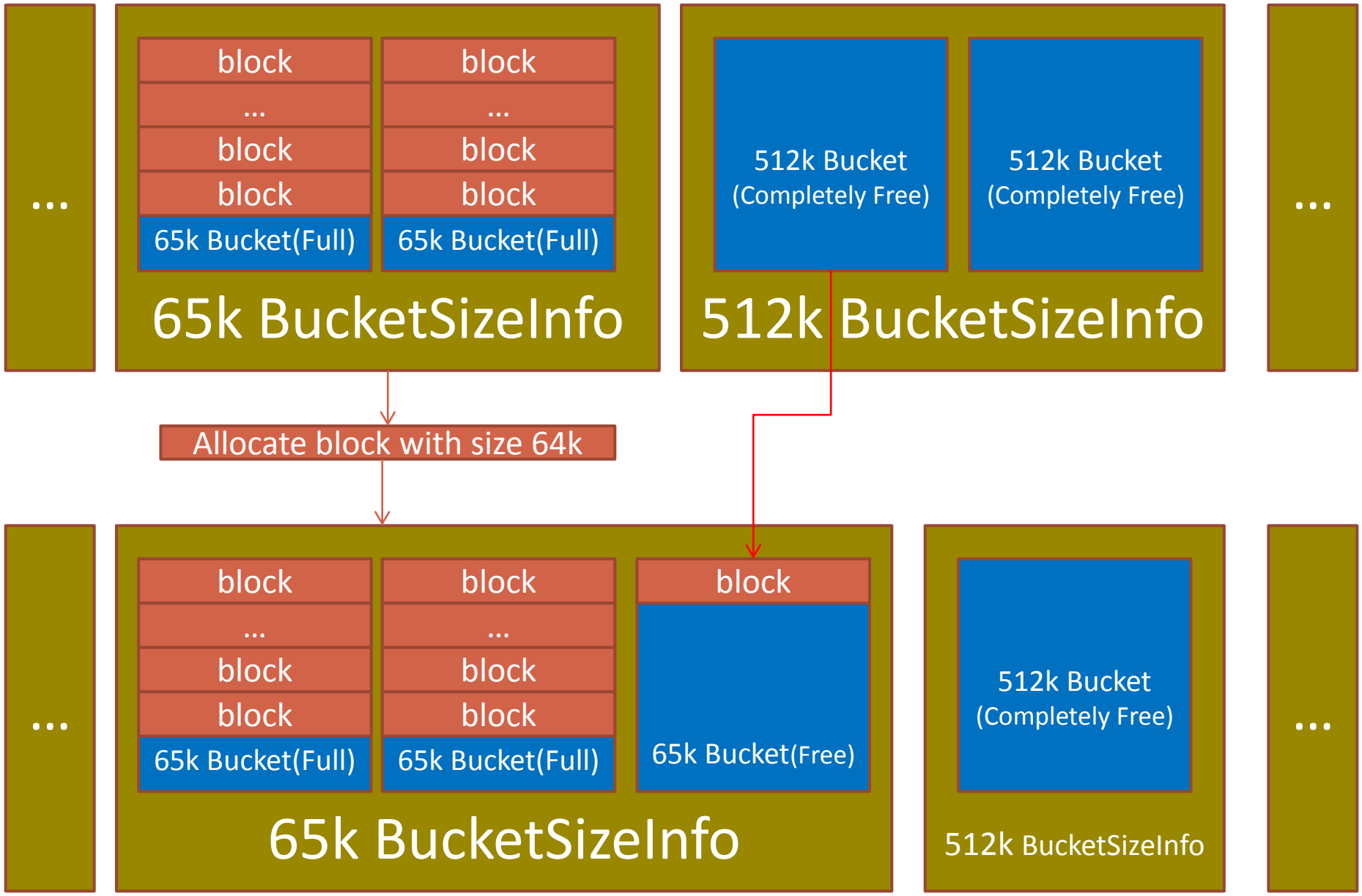


1.Each bucket has a fixed capacity, 2MB as default;
2.Each bucket is specified a size and caches blocks up to this size
3.For completely free bucket, its size could be re-specified
4. Bucket allocator just allocate/free blocks logically, physical block data is stored on the IO engine

Allocate block in Bucket Allocator



Allocate block in Bucket Allocator



Free block in Bucket Allocator

Test Results of First Usage

- Test Environment
  - HBase-0.94,5 client, 5 regionserver, 50 threads per client, DataBlock hit ratio about 75%
- Case 1(Read Only):

	QPS	RT	Load	YGC	YGCT	CMS	CMST
Before	11328	4.41ms	9.92	4870	237.9s	268	7.9s
After	13688	3.65ms	11.59	7131	147.6s	0	0

- Test Case 2(Read + Write)

	QPS	TPS	Load	YGC	YGCT	CMS	CMST
Before	11333	3133	10.28	6609	355.4s	437	15.8s
After	12185	3813	10.58	7162	257.5s	10	0.7

Test Results of Second Usage

- Test Environment
  - Using fusionIO to store cached data

- Case (Read Only):

	QPS	RT	Data Block Hit Ratio	Bucket Cache Hit Ratio	IOPS of Read Bucket Cache (Fusion-IO)	RT of Read Bucket Cache (Fusion-IO)	RT of Read Datanode
Before	1934	42.21ms	11.12%	0	0	0	44.98ms
After	14420	2.2ms	22.5%	87.3%	11598	1.5ms	2.58ms