HBase近期的发展与实践

沈春辉

2013.4

Confidential & Proprietary

大纲

- HBase简介及发展
- 实践与改进
- 我们目前的工作

Confidential & Proprietary

HBase 简介及发展

- HBase：高可靠性、高性能、面向列、可伸缩、支持实时读写的分布式存储系统
  - 海量数据（TB 以上）
  - 很高的随机写能力
  - 在海量数据中实现高效的读取
  - 很好的伸缩能力
  - 强一致性
  - 能够同时处理结构化和非结构化的数据
  - 动态列
  - 不需要复杂的查询需求: SQL、事务、Join、多维索引等（Many things doing on this）

Confidential & Proprietary

HBase简介及发展

- HBase 的版本发展
  - 2007.4　第一个版本 (HBASE-287 Mike Cafarella)
  - 2010.10 0.89　(Facebook 的生产版本 based on)
  - 2011.4　0.90.2 (阿里HBase的第一个版本 based on)
  - 2012.1　0.92　(Adds 安全, 协处理器, HFile V2, 分布式log-splitting)
  - 2012.5　0.94　(Performance Release)
  - 2013.4　0.95　(0.96的预发版, Adds protobuf，Table Snapshot，PrefixTreeCompression 等等)
  - Coming 0.96

Confidential & Proprietary

HBase简介及发展

- HBase 在阿里的发展
  - 2011.3月开始研究
  - 2011.5月上线第一个应用
  - 截止目前：
    - 以基于0.94的阿里HBase版本为主
    - 稳定性达到在线应用的生产标准之上
    - 近百个业务，特点多样化

Confidential & Proprietary

HBase简介及发展

- 国内外应用发展HBase的公司
  - Cloudera、Hortonworks、Salesforce等
  - Facebook、Yahoo、eBay、Twitter、Pinterest、 Line 等
  - Intel 、IBM等
  - 阿里巴巴、华为、小米、360等

Confidential & Proprietary

实践与改进

- 海量数据中的高效读
- 优化Java的GC麻烦
- 更高的服务可用性
- 其他

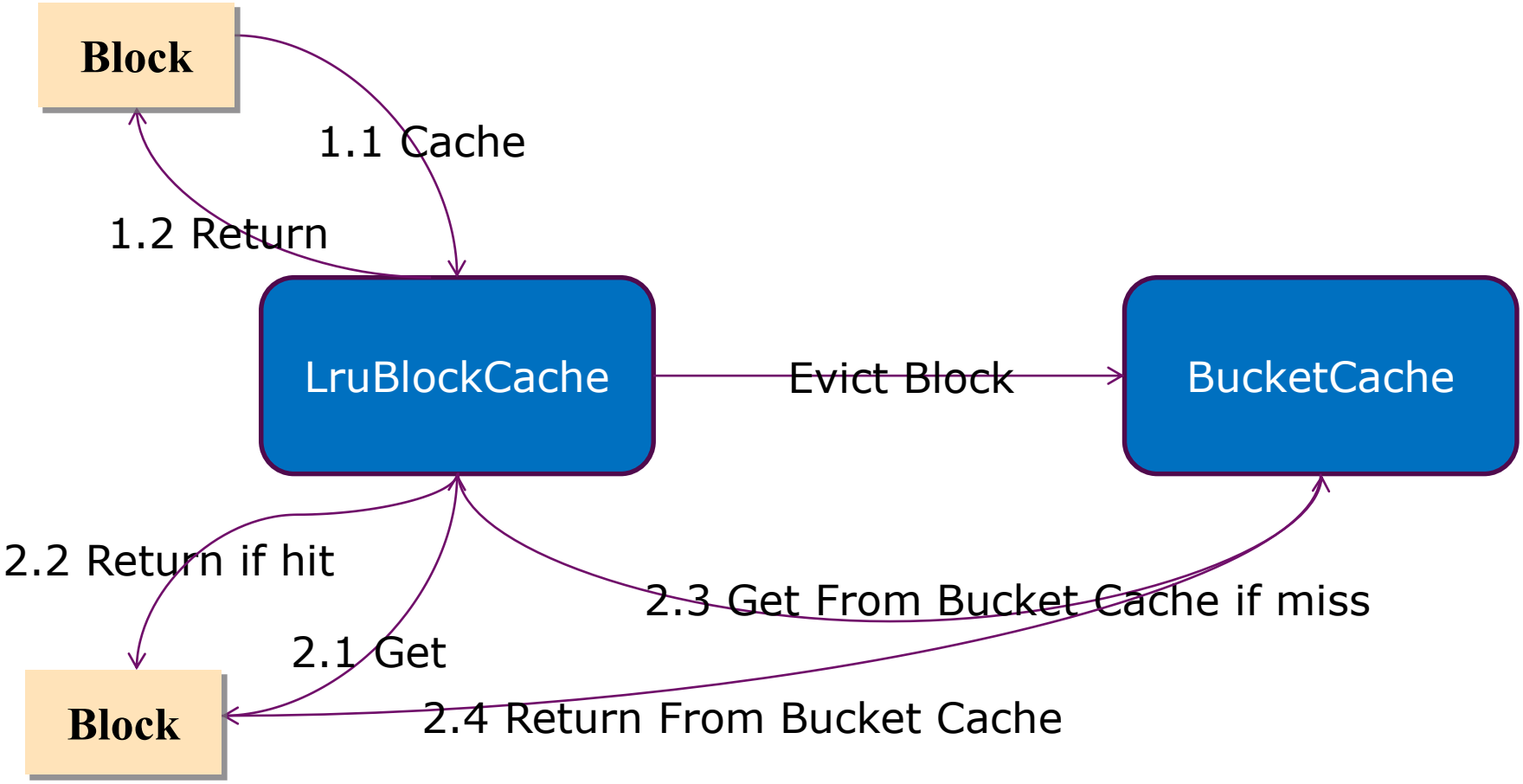
Confidential & Proprietary

实践与改进-Bucket Cache（L2 Cache on HBase ）

- 海量数据+高效的随机读性能
  - 96G、128G内存？
  - 直接采用SSD、Fusion-IO做底层存储介质？
  - L2 Cache？ -> Bucket Cache

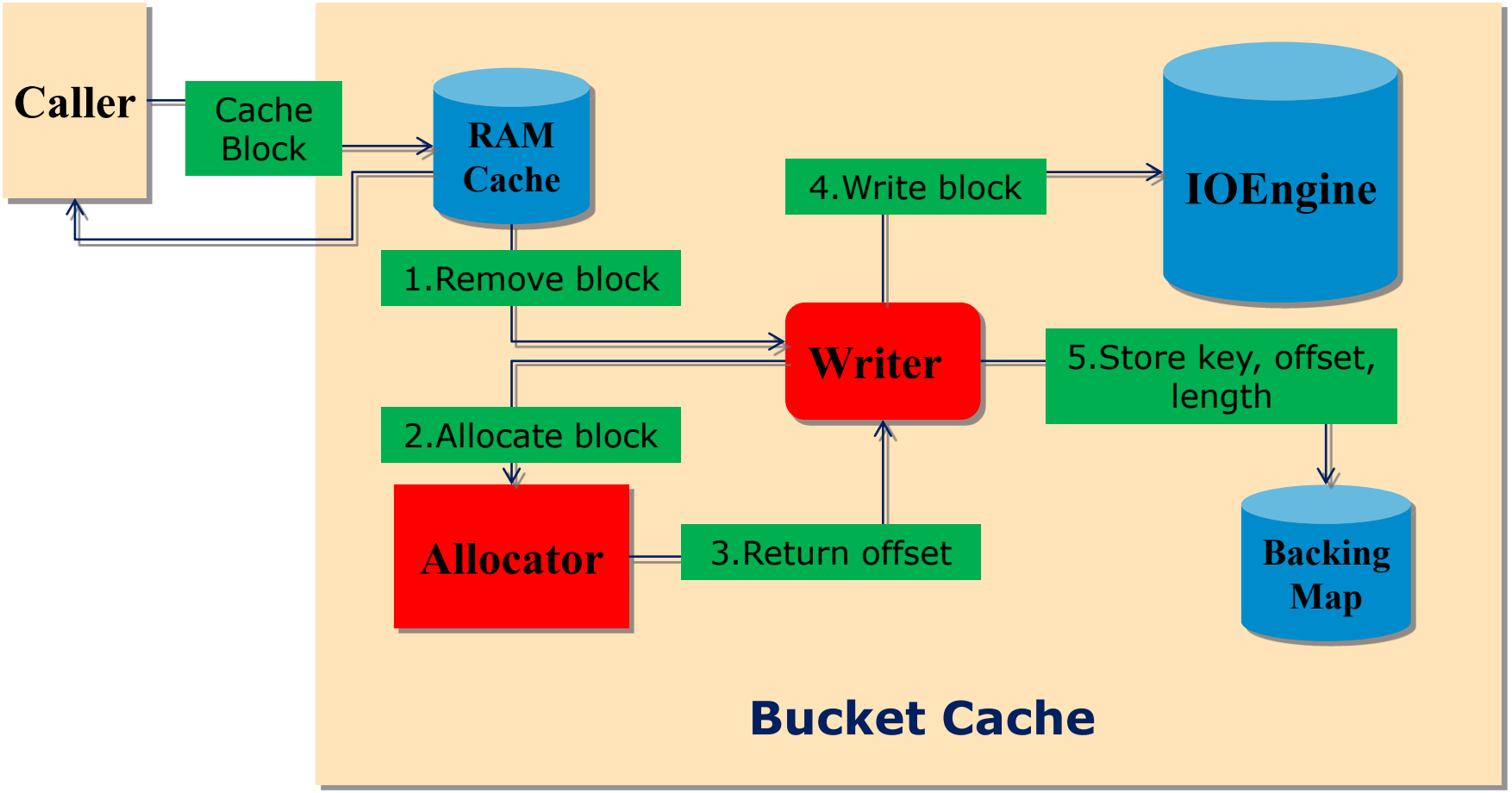
Confidential & Proprietary

实践与改进-Bucket Cache

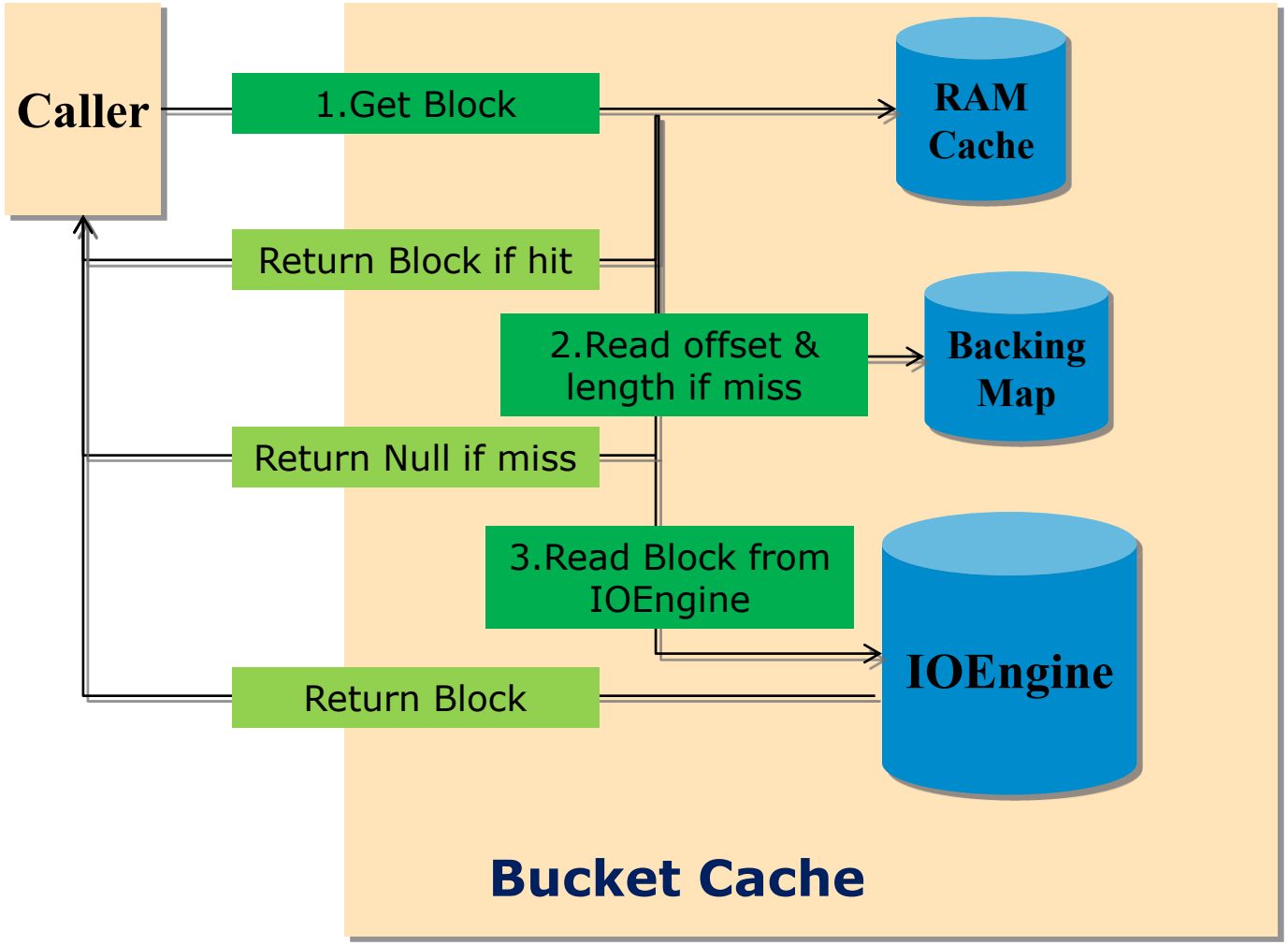


实践与改进-Bucket Cache (Cache Block)

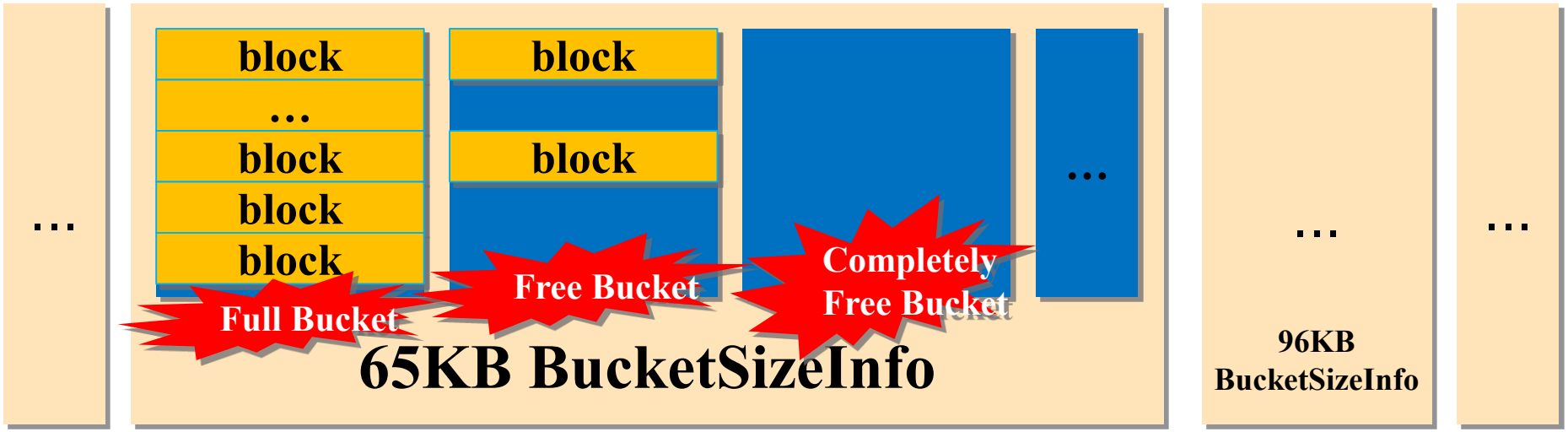
DTCC2013



实践与改进-Bucket Cache (Get Block)



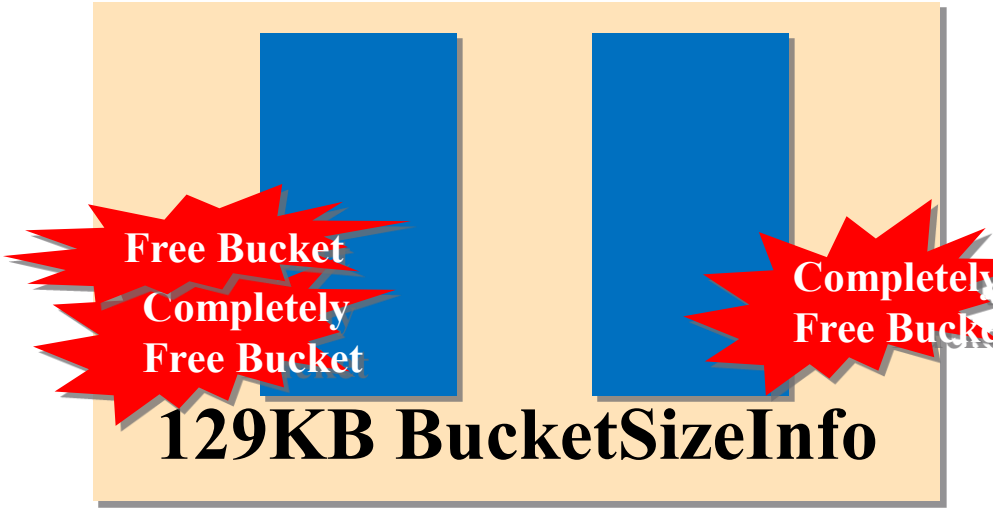
实践与改进-Bucket Cache (Allocate Block)



1.将整个逻辑上的存储块切割成一个个的Bucket
2.每一个Bucket都有相同的固定的容量，e.g. 2MB as default;
3.每一个Bucket都拥有一个size tag，cache这个size以内的Block
4.对于完全空闲的Bucket，它的size tag可以被重新指定
5.每一个Bucket与物理存储(IOEngine)的位置有固定的映射关系，根据Block在Bucket中的偏移，则就可以计算出其在物理存储中的偏移

实践与改进-Bucket Cache (Allocate Block)

59KB Block
62KB Block
63KB Block
66KB Block
64KB Block



实践与改进-Bucket Cache

- Block无法分配或者空间使用达到设置阀值，触发异步地淘汰机制
- HBASE-7404
- Origin code comes from FIO
- Performance Test
  - 0.94 + L2 Cache采用fusion-IO作为存储
  - 50 并发线程

	QPS	RT	L1 Hit Ratio	L2 Hit Ratio	IOPS On FIO	RT On FIO	RT On Datanode
Before	1934	42.21ms	11.12%	N/A	N/A	N/A	44.98ms
After	14420	2.2ms	22.5%	87.3%	11598	1.5ms	2.58ms

Confidential & Proprietary

实践与改进

- 海量数据中的高效读
- 优化Java的GC麻烦
- 更高的服务可用性
- 其他

Confidential & Proprietary

实践与改进-优化JAVA的GC麻烦

- Concurrent mode failure
  - 原因：old区要分配内存了，但是old区空间不够，而此时CMS正在进行中
  - 解决：降低YGC频率，降低CMS GC触发时机(降低CMSInitiatingOccupancyFraction的值)
- Promotion Failed（发生很多）
  - 原因：old区要分配内存了,但是找不到空间分配，却还没达到CMS的触发值。
  - heap碎片+YGC晋升对象（Block，MemStore Chunk）过大

Confidential & Proprietary

实践与改进-优化JAVA的GC麻烦

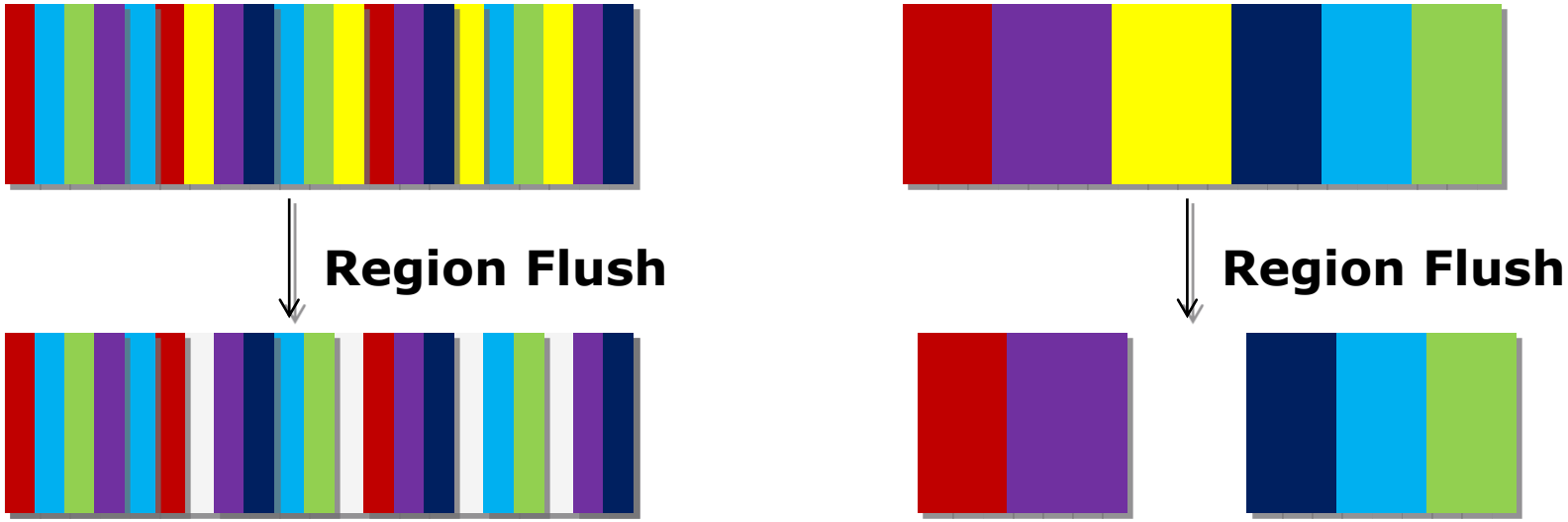
- Promotion Failed（发生很多）

  解决：

  – 聚合小数据：开启mslab （注意region数目）

  – HBase自己管理（重用）内存对象

    - Bucket Cache（HBASE-7404）（使用Byte Buffer作为IOEngine的存储介质）
    - Chunk Pool（HBASE-8163）



实践与改进-优化JAVA的GC麻烦

- YGC停顿过长
  - 原因：young区活对象数目过多、old区脏页数目过多
  - 原因：MemStore中的KeyValueSet，开启WAL压缩后的字典map
  - 解决：
    - 降低young区大小
    - 减小-XX:MaxTenuringThreshold
    - 开启BucketCache/ChunkPool，减少晋升对象，减少拷贝时间

Confidential & Proprietary

实践与改进

- 海量数据中的高效读
- 优化Java的GC麻烦
- 更高的服务可用性
- 其他

Confidential & Proprietary

实践与改进-更高的服务可用性

- 服务单点
  - Regionserver 宕机后，部分Region将处于不在线状态
- 缩减Regionserver宕机后的恢复时间
  - 通过脚本去除3分钟的ZK感知（HBASE-5844）
  - 改进Split-Worker (HBASE-6134)、修改NFS中的tcp_nodelay为true
  - 加大Meta Scan的caching（HBASE-5913)
  - SSH采用Bulk Assign（HBASE-5914）
  - Master加快处理Region Opened事件 (HBASE-5970)
- 2012双11前的宕机演习：宕机恢复时间 -> 1分钟

Confidential & Proprietary

实践与改进-更高的服务可用性

- 更多的优化
  - 优化AssignmentManager内部的同步及拷贝 (HBASE-6109)
  - 将用户HLog与META/ROOT HLog分离 (HBASE-7213)
  - Log-Split期间支持数据写入和时间范围内的读(HBASE-6752)
  - Master启动时，采用SSH方式恢复Dead Server(HBASE-7824)

Confidential & Proprietary

实践与改进

- 海量数据中的高效读
- 优化Java的GC麻烦
- 更高的服务可用性
- 其他

Confidential & Proprietary

实践与改进-其他

- 更好的运维性
  - 在线Region Merge
  - Compaction的动态控制
  - 慢响应请求的跟踪定位
  - RPC请求的控制

Confidential & Proprietary

实践与改进-其他

- 需要Merge的情形：
  - Region出现hole或者overlap
  - TTL+不合理的rowkey设计 => 空region
  - 创建表时，pre-split不合理，部分region没有或者很少写入数据
  - 低版本(e.g. 0.90)HBase时region数目比较多，升级到0.94+后需要Merge

- 在线Region Merge(HBASE-7403)
  - 依赖于META region中的多行事务（HBASE-7721）

Confidential & Proprietary

实践与改进-其他

- 更好的运维性
  - 在线Region Merge
  - Compaction的动态控制
  - 慢响应请求的跟踪定位
  - RPC请求的控制

Confidential & Proprietary

实践与改进-其他

- Compaction的影响：消耗大量网络以及io的带宽
- 优化：
  - 高低峰期使用不同的compaction
  - 允许高峰期禁止Major Compaction
  - 动态调整Compaction参数（线程数目、低峰期时间、高峰期enable/disable Major Compaction）
- More
  - 新的compaction策略—Stripe Compaction （HBASE-7667）

Confidential & Proprietary

实践与改进-其他

- 更好的运维性
  - 在线Region Merge
  - Compaction的动态控制
  - 慢响应请求的跟踪定位
  - RPC请求的控制

```
process_time=46 response_size=1777 call_size=181 get{pttest1020q,,1365406962137.1350d8ca4241b3098c4d23bb08fbffb9., {"timeRange":
[0,9223372036854775807],"totalColumns":1,"cacheBlocks":true,"families":{"c1":
["ALL"]},"maxVersions":1,"row":"3H4AC5WSSOHBNUEYCB5ADVXDSVHKM6M9Y7T3XDP6YQSKLVGJ6D"}), rpc version=1, client version=29, methodsFingerPrint=1289090786 from
10.[redacted]50:56128 ProfilingData:data_block_miss_cnt:1, index_block_miss_cnt:1, server_process_time.ms:46, response_size.byte:1777, server_queue_time.ms:1,
total_block_read_time.ms:44, HFILE_NAME:{a4f863e7118d444093279150ce0b6dc3[INDEX Block offset=1670507047/compressedSize=131167
/decompressedSize=131098],a4f863e7118d444093279150ce0b6dc3[DATA Block offset=1655862671/compressedSize=65667/decompressedSize=65614],}
```

实践与改进-其他

- 更好的运维性
  - 在线Region Merge
  - Compaction的动态控制
  - 慢响应请求的跟踪定位
  - RPC请求的控制
    - RPC的黑白名单
    - RPC的队列控制

Confidential & Proprietary

实践与改进-其他

- HDFS的改进
  - Namenode HA
  - HDFS Sync并行写三份
  - 新的块放置策略
  - 跳过checksum
- HBase Replication
- 自动化测试
- 可视化监控
- 自动化报警
- 热升级

Confidential & Proprietary

实践与改进-Next & Future

- Coming HBase-0.96
  - Table Snapshot
  - Protobuf
  - PREFIX_TREE data block encoding
  - Region Server Group
  - 模块化

- 展望HBase的未来 by Ted Yu
  - http://www.csdn.net/article/2013-04-08/2814798-the_future_of_HBase

Confidential & Proprietary

我们目前的工作

- SQL+事务+索引
  - WASP
- 多租户下的Big Cluster
- 认证与权限
- 机房容灾
  - 冗余容灾（replication、snapshot）
  - 单集群跨机房
- 支持JDK7

Confidential & Proprietary

Thank you!

Q & A

Confidential & Proprietary