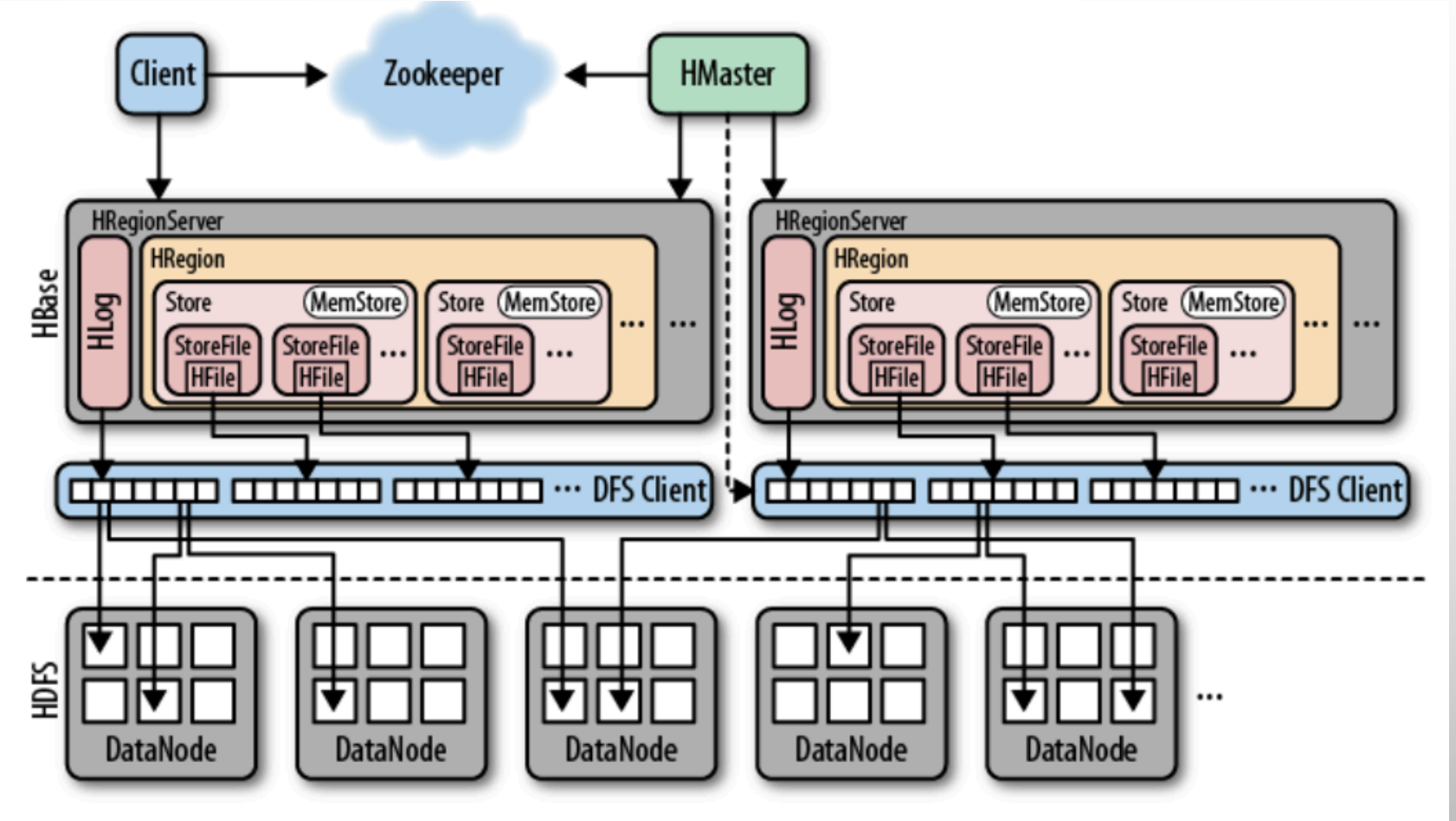
# HBase回顾

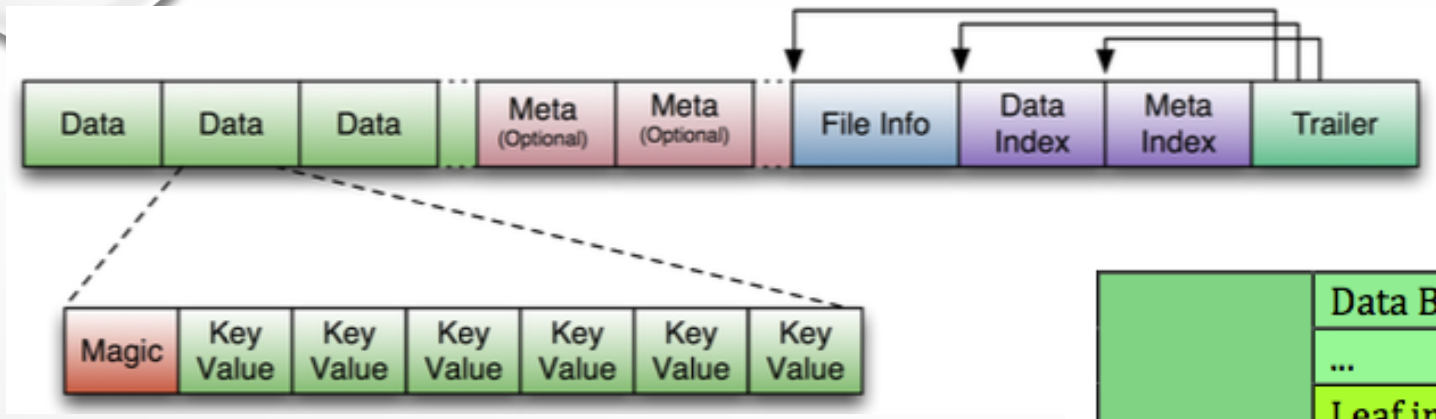
邓志华

# HBase

- TreeMap？ Get/put/iterate/remove/contains…



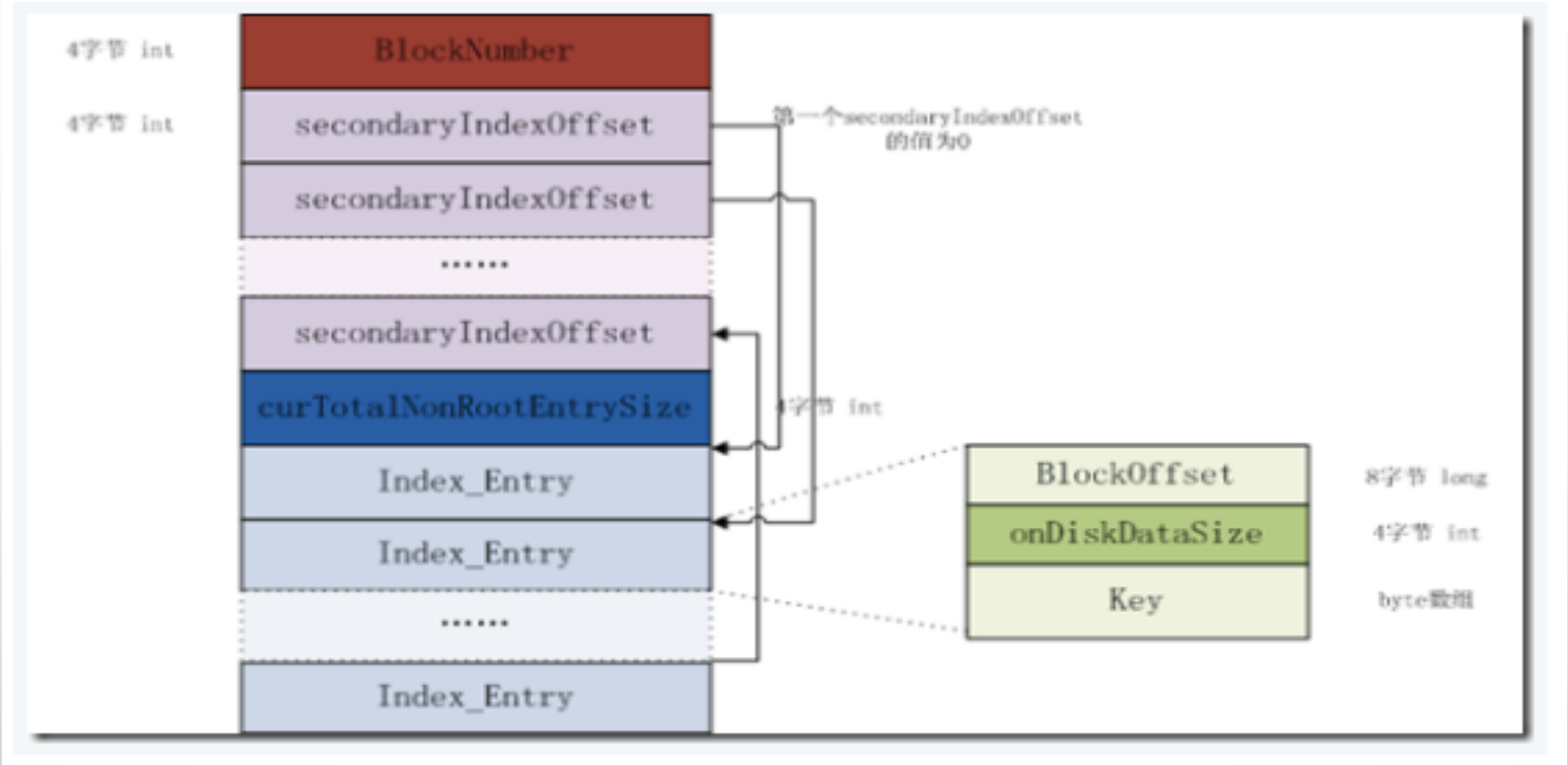
# HFile




How to position an inputKey rapidly?

| | | | |
|---|---|---|---|
| "Scanned block" section | Data Block | | |
| | ... | | |
| | Leaf index block / Bloom block | | |
| | ... | | |
| | Data Block | | |
| | ... | | |
| | Leaf index block / Bloom block | | |
| | ... | | |
| | Data Block | | |
| "Non-scanned block" section | Meta block | ... | Meta block |
| | Intermediate Level Data Index Blocks (optional) | | |
| "Load-on-open" section | Root Data Index | | Fields for midkey |
| | Meta Index | | |
| | File Info | | |
| | Bloom filter metadata (interpreted by StoreFile) | | |
| Trailer | Trailer fields | | Version |

# Index(Root + Intermediate)



# LSM tree vs B+ tree

## Seek vs. Transfer (Example)

- Comparison
  - 10 MB/s transfer bandwidth
  - 10 ms disk seek times
  - 100 bytes per entry (10 billion entries)
  - 10 KB per page (1 billion pages)

- When updating 1% of entries (100,000,000), it takes:
  - 1,000 days with random B–tree updates
  - 100 days with batched B–tree updates
  - 1 day with sort and merge (LSM)

- ➔ At scale, seek is inefficient compared to transfer!

# Why HBase

- 更容易管理大量的数据
- 灵活的schema定义，类型，大小等
- 方便的支持水平拓展及高可用
- 强一致性

# Hbase优化

- 1， HDFS垃圾日志刷屏

  HBASE-16212 (https://issues.apache.org/jira/browse/HBASE-16212)

- 2， Regionserver不稳定原因

➢ 2.1， GC (gc pool 'concurrentmarksweep' had collection(s): count=2 time=63229ms)

-XX:pretenuresizethreshold=2097088 -Xx:cmsinitiatingoccupancyfraction=70

-XX:+usecmsinitiatingoccupancyonly -XX:cmsmaxabortableprecleantime=500

-XX:maxtenuringthreshold=5 -XX:targetsurvivorratio=66

-XX:-usebiasedlocking -XX:+perfdisablesharedmem

OPENJDK8中 G1 垃圾回收要次于CMS，默认的垃圾回收机制为**PARALLELGC**(jinfo命令）

# Hbase优化

- 2， Regionserver不稳定原因

➢ 2.2， HDFS Pipeline IO压力造成中间DATANODE写入失败，恢复Pipeline时出现问题。

具体原因分析： https://housong.github.io/2016/streaming-pipeline/

➢ 2.3， 配置遗漏

修复更新用户组文件时不能及时更新进程内用户组映射关系的问题

修复Hbase中一些小BUG

- 3， 搭建监控报警体系

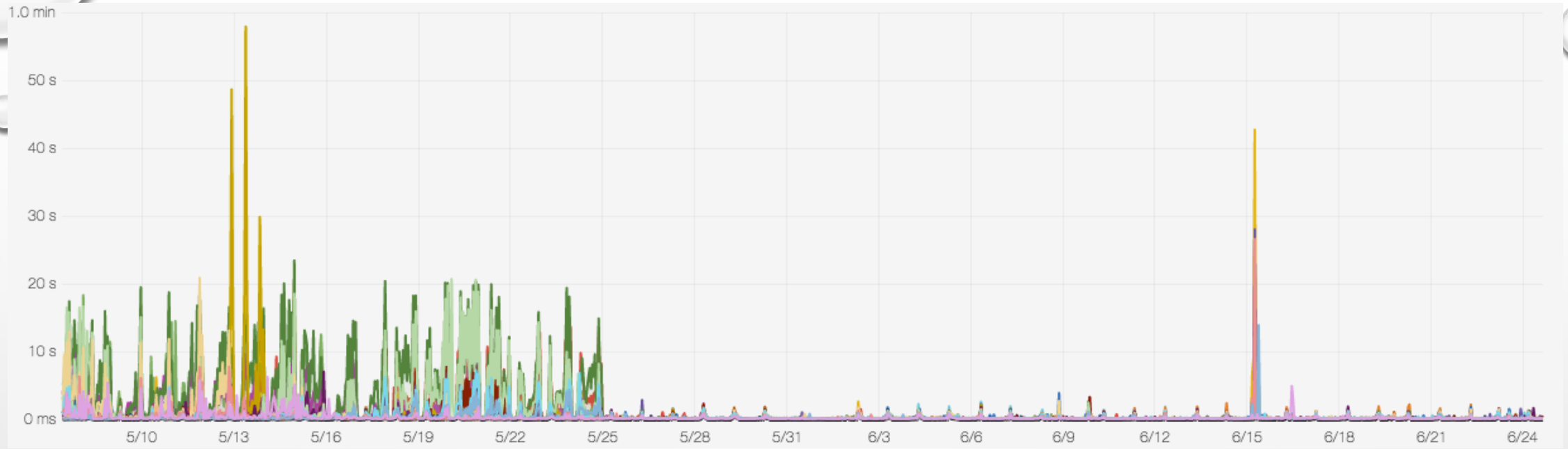
➢ 3.1 collectd + grafana + graphite

# Hbase优化

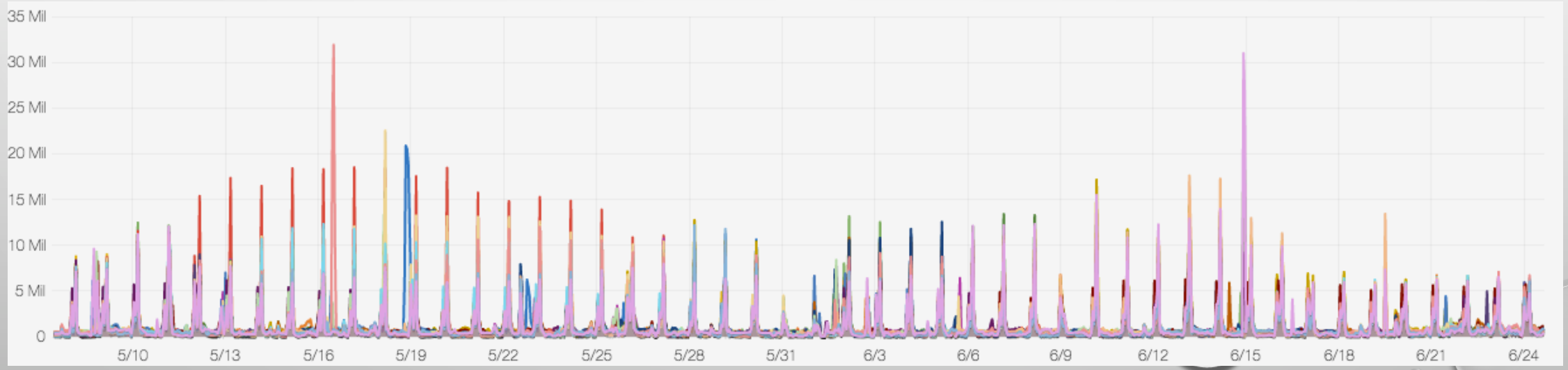
- 4， HDFS + HBASE硬件升级，SSD作为读缓存，WAL日志输出，数据本地性(major_compact)
- 5， 软件调优
  - 读写缓存(LRU -> BUCKET)，读写分离，HDFS补偿读，并行SCAN， IPC读写队列，Metastore MSLA pool，…
- 6， 搭建Rest服务
  - 6.1，添加IP受限(认证），服务健康检查
  - 6.2，GET / PUT / SCAN 指标监控
  - 6.3，用户代理

Python => https://github.com/barseghyanartur/starbase

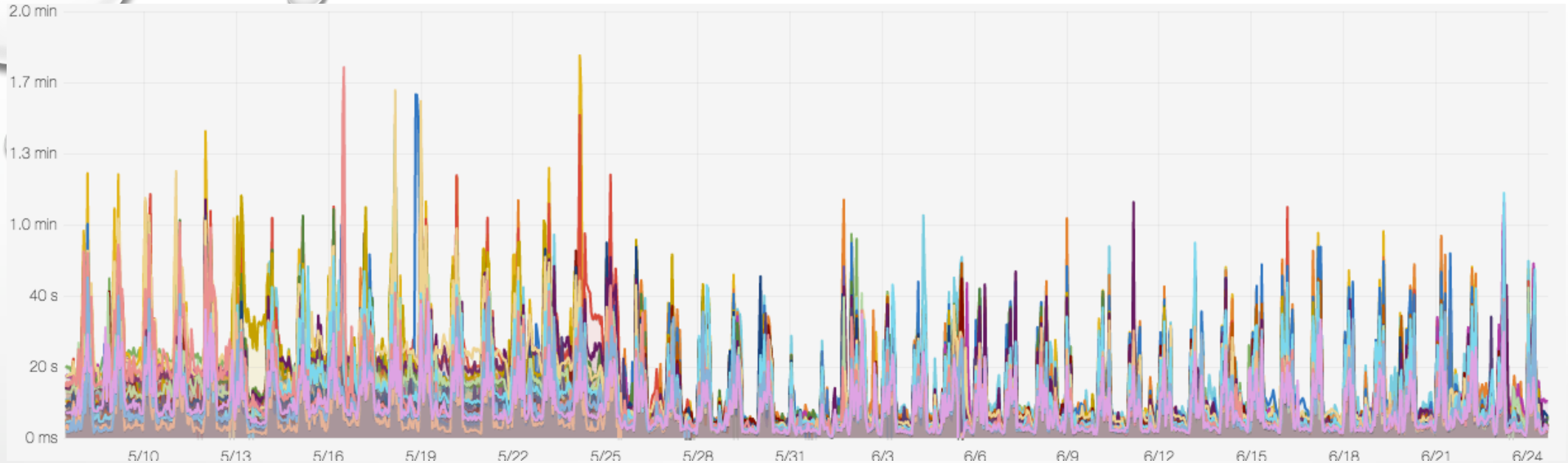
- c1-hd-dn*.rs_metric_op.gauge.op_ipc_processcalltime_99th_percentile (20170507~20170624)



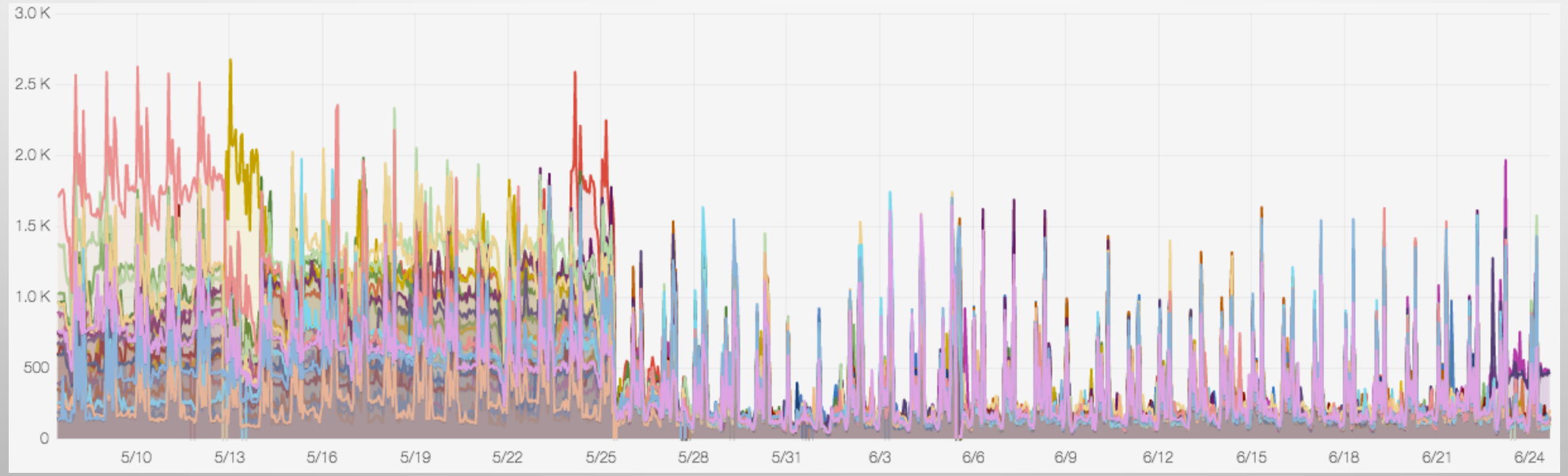

- derivative(c1-hd-dn*.rs_metric.gauge.regionserver_totalrequestcount)



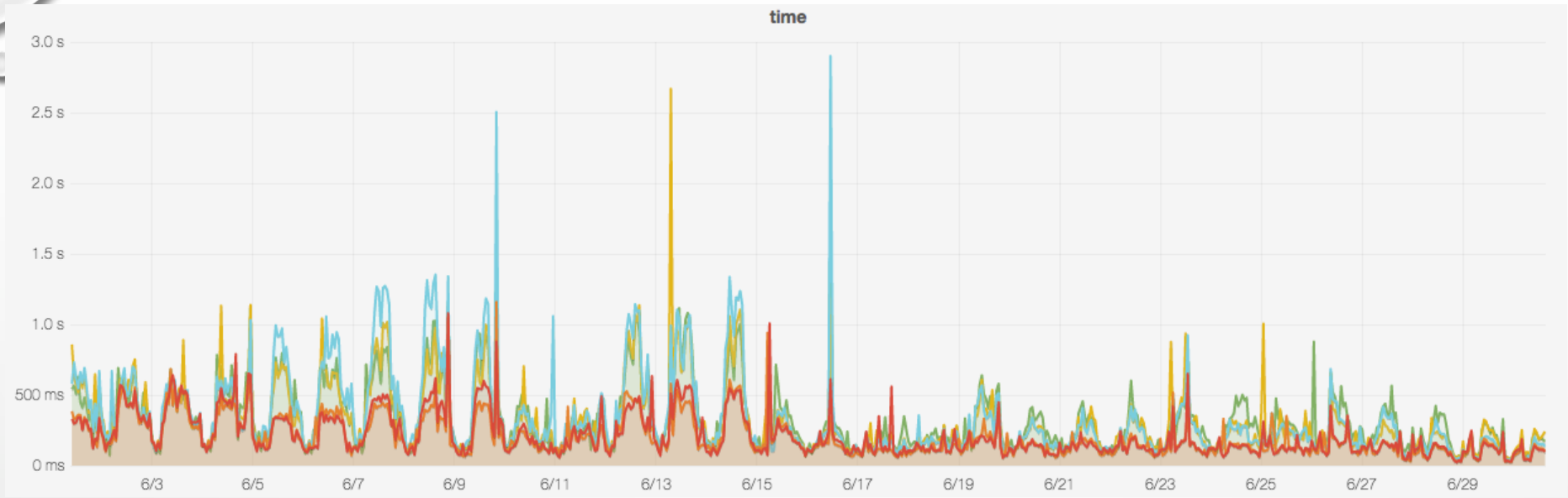
- derivative(c1-hd-dn*.rs_metric.gauge.jvmmetrics_gctimemillis) (20170507~20170624)



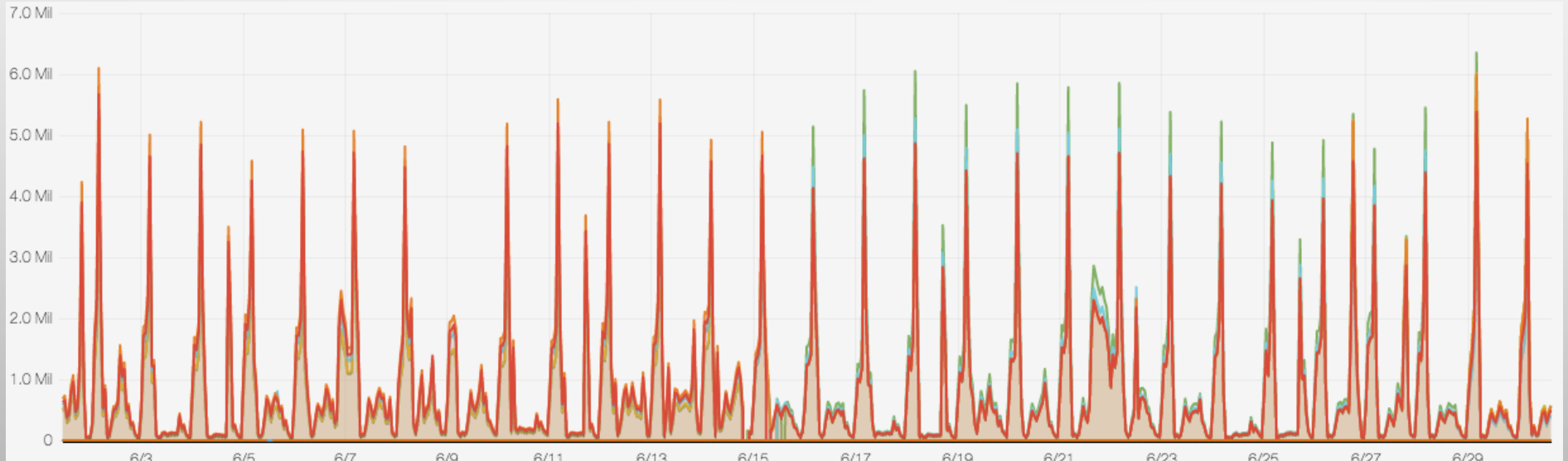

- derivative(c1-hd-dn*.rs_metric.gauge.jvmmetrics_gccountparnew)



- c1-hd-dn*.rs_metric_op_ol.gauge.op_ipc_processcalltime_99th_percentile(20170601~20170630)




- derivative(c1-hd-dn*.rs_metric_ol.gauge.regionserver_totalrequestcount)



# Hbase使用

- 设计表

➢ 1, 考虑数据的获取方式(Map)，Metrics？; Column Family的数量 <= 3
(http://hbase.apache.org/book.html#number.of.cfs)

➢ 2, Key salt, *Hashing & Reversing the Key,* *分散热点数据*

➢ 3, Pre split Region，读写吞吐量，性能，预分配策略？

➢ 4, KeyValue size

HFileBlock => 默认64K， HFileBlock大小超过4M时，读缓存(BucketCache)不会缓存该数据

MSLA(MemStore-Local Allocation Buffer) 默认大小为2M

# HBase使用

- 设计表

➢ 5，减小row和column family的大小

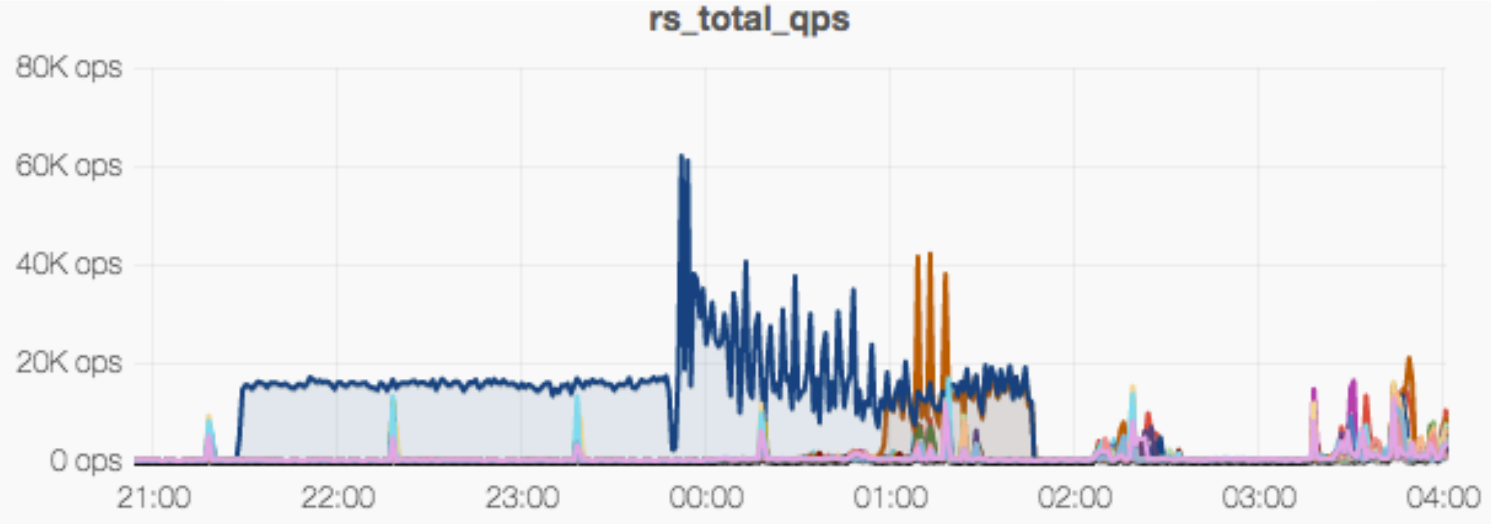
➢ 6，数据压缩

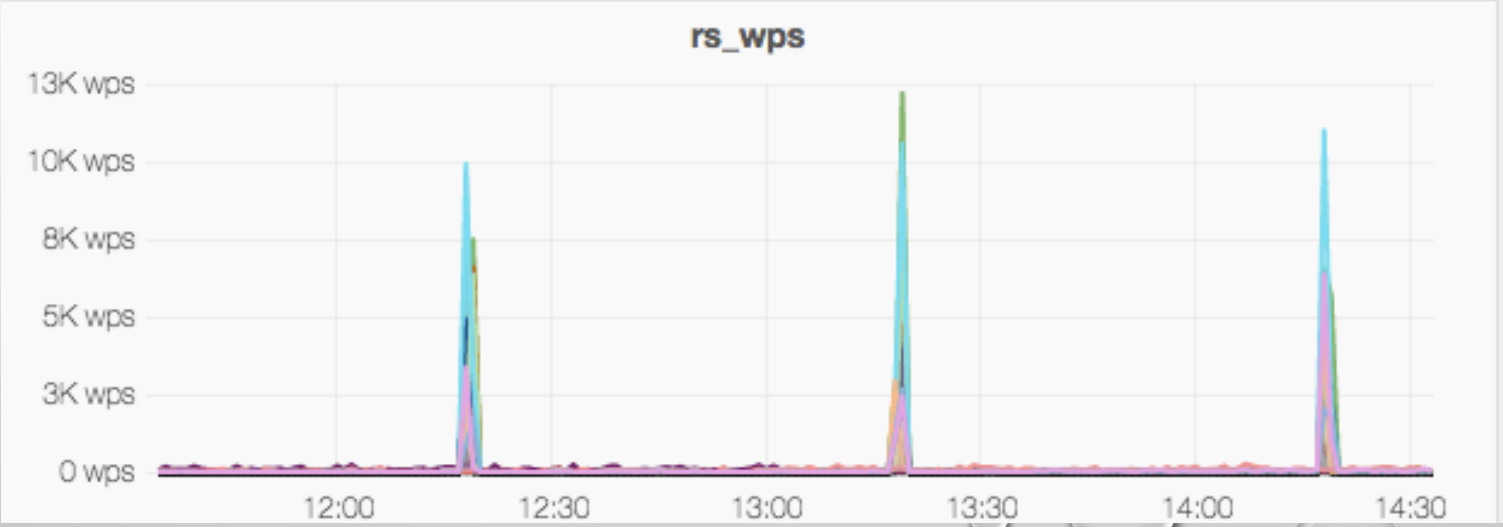
存储压缩： Snappy > LZO

KV Encoding： RowKey比Value长度要长，或者一行存在许多列时，使用 DATA_BLOCK_ENCODING => 'FAST_DIFF'

➢ 7，ES or hbase table 来存储索引

scale(derivative(c1-hd-dn*.rs_metric.gauge.regionserver_totalrequestcount), 0.0167)



# Scan&Put

- Scan with rowkey，ScannerTimeoutException, lease(不会自动刷新, 不同于HDFS lease)
- setBatch vs setCaching
  - Scan caching默认是关闭的，可以通过setCaching方法来开启，减少RPC总次数(row level)。
  - setBatch可控制的是一次RPC请求可以获取多少列，内存无法容下一行的数据时，可考虑采用这个方法
- Cache block？
  - setCacheBlocks()，当读取一次表后就不再读取对应的数据时，将这设置成false，比如mr或者一些不同集群之间表同步等
- Always close scan in finally clause
- filter？
- Batch puts, enable client-side write buffer by table.setAutoFlush(false)

# 常见问题

- StopWatch NoSuchMethodException
  - hadoop版本上使用的是guava 18版本， 与hbase中的11版本存在一些类冲突的问题

  mapreduce.job.user.classpath.first = true/ export HADOOP_USER_CLASSPATH_FIRST=true
- BlukLoad is more preferred than batch puts(64M, spark streaming/mr)
  - 在github上有很多这样的示例
- Take a snapshot before disable a table if you want to reuse in the future?

# 常见问题

- 出现问题怎么办？
  - 异常栈信息，包括时间点，异常栈输出，表， region…
  - 请求的访问方式 (Client Api/ Thrift/ Rest)
  - 监控指标
  - google，社区…

谢谢