阿里Hbase的业务和容灾实践

穆公(朱金清 suinking@gmail.com)
微博：淘穆公
2013.8.21

大纲

- 简介
- 数据模型
- 业务设计
- 产品线使用建议
- 容灾
- 总结

简介

- Nosql: column-based storage system
- Large volume of data
- High write (esp. random) through-put / Good ramdon read performance
- Range query
- Row-base transaction
- Auto-sharding
- Compare to Bigtable
  - Hbase Based on Hadoop HDFS or other HDFS
  - Bigtable based on GFS

Large volume

- 三层索引结构
  - Region的大小默认最大是256M
  - 按照平均128M算；

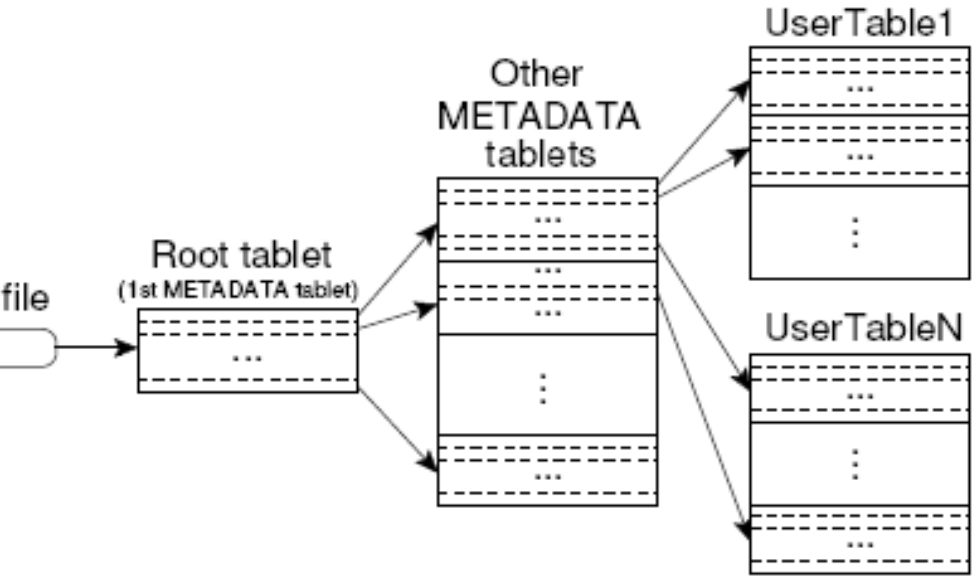


Figure 4: Tablet location hierarchy.

假设：一个rowkey 1KB

1. Root table： 128M=128*1024KB
   即2^7 * 2^10 = 2^17 bucket
2. Meta table： (2^17)^2 = 2 ^34 bucket
3. **记录数：2^51 条记录**

其它特征

- 三层B+树的扩展LSMTree[1]
  - 适合于范围查询
  - Rowkey的字母顺序来排序(byte数组存储)
- Row-base
  - 事务级别仅限于rowkey级别
- Auto-sharding
  - Region的自动split/move

问题：牺牲了**CAP**中的？

- [1] Jim Gray and Franco Putzolu, "The Five Minute Rule for Trading Memory for Disk Accesses and The 10 Byte Rule for Trading Memory for CPU Time", *Proceedings of the 1987 ACM SIGMOD Conference, pp 395-398.*

已有适合的使用场景

- 海量数据写入
  - 历史数据　　批量写入
- 消息类（类似Facebook的message）
  - 消息类
- Schema-free
  - 业务监控
- LOG-Append类的业务
  - 全网志　　全网每天上百亿
- 大表的复杂/多维度索引
  - 检索索引，主数据在mysql
- 分析类

- 大批量读取
  - HBase+缓存TAIR

现有集群状况

集群名称	TPS(avg)	11.11最高	QPS(avg)	11.11最高	版本
业务	7k	1.8w	1.6w	3.4w	0.90.2
业务	1.8w	2w	1.2w	1.4w	
业务	7k	3w	2w	5w	
业务	1k	2k	2k	6k	
业务	**2.5w**	5w	2w	6w	
业务	**10w**	**25w(最高50w)**	1w	2w	0.94
业务	4w	20w (压测)	2k	3w(压测)	0.94
业务	每天2-3kw	-	RT在ms级别	-	0.90.2-定制版
业务	10w	25w	15w	100w	0.94
业务	3k	1.4w	3k	6k	0.94
业务	1.5w	2w	6k	8k	0.94

与MYSQL的对比

场景	HBase优点	HBase缺点	MySQL优点	MySQL缺点
业务表使用	使用简单，一张表即可	不过没有SQL	有SQL；分库分表，灵活	分库后
更新模式	插入多的适合	RKupdate差	DML	
二级索引策略		需借助索引表	强	DDL问题
客户端接口	灵活自己掌握	无标准SQL	SQL	
写性能	非常强	顺序写入时瓶颈在一台rs	较强	几千tps/单套库
读性能	较强；支持scan	依赖内存	很强；支持scan	依赖索引
可扩展性	强		借助愚公/datax工具可动态扩展	弱
运维方便	自己定制	不够成熟	成熟	
DDL	时间短；92版本可以在线	若有索引表，需要自己填充	Create index即可	时间长；block读写
稳定性CAP	CP	A	AP	C

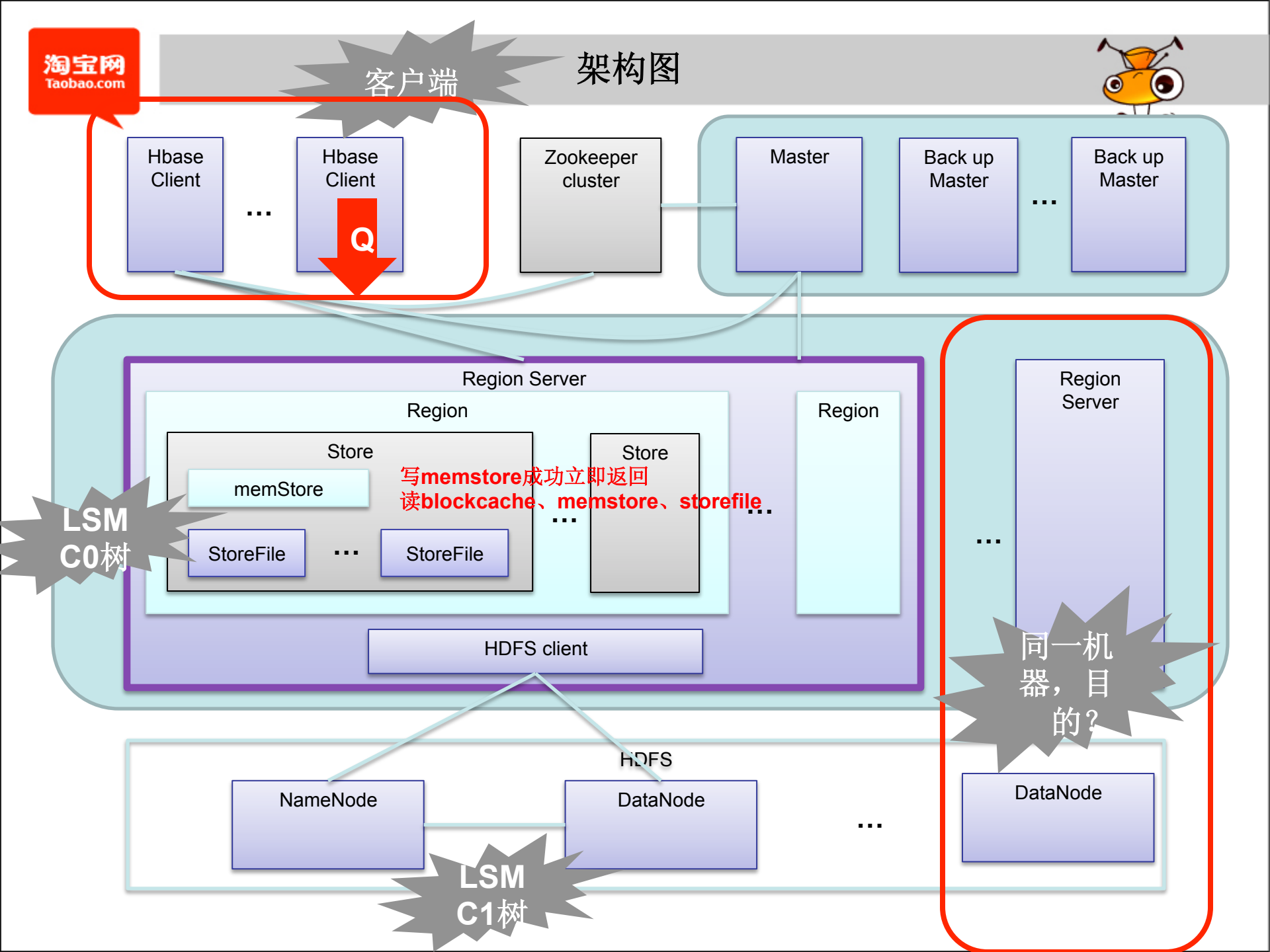
NoSQL使用情况

- ALIBABA
  - Hbase/OTS
- BAIDU
  - BAILING/ARMOR[1]
- TENCENT
  - TDB/TSSD
- FACEBOOK
  - HBASE

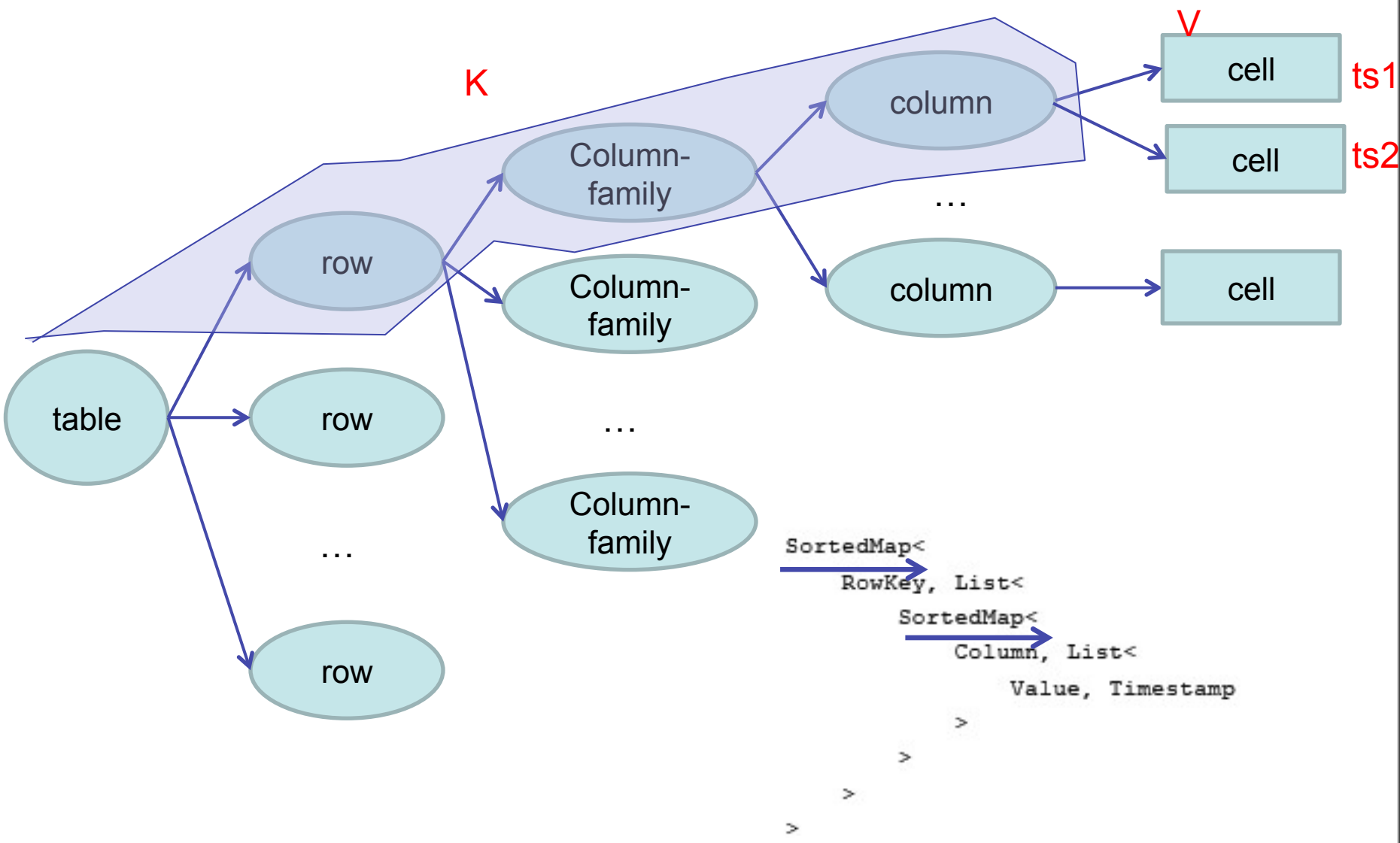
[1]http://wenku.it168.com/d_000926299.shtml

大纲

- 简介
- 数据模型
- 业务设计
- 产品线使用建议
- 容灾
- 总结

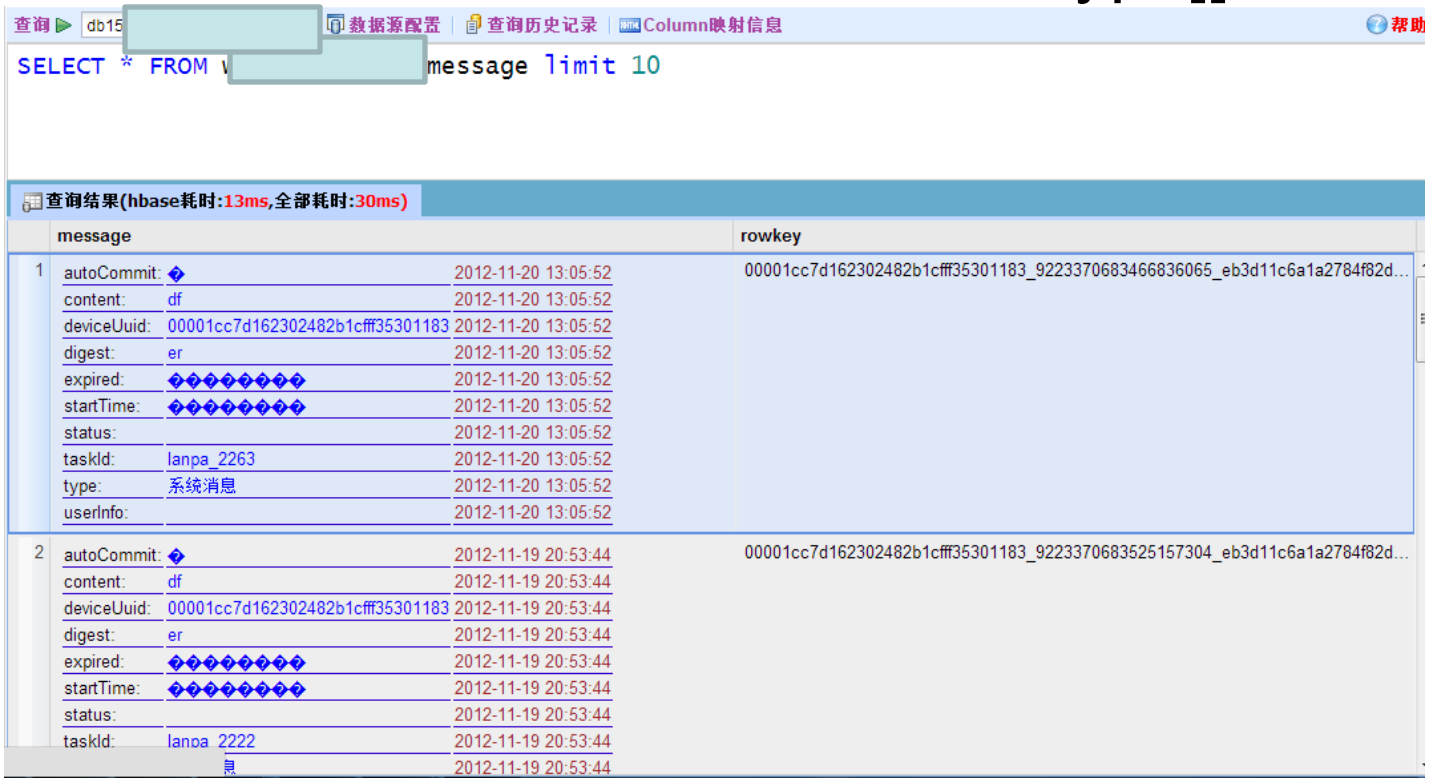

淘宝网
Taobao.com
客户端
架构图
Hbase Client
Hbase Client
Q
Zookeeper cluster
Master
Back up Master
Back up Master
Region Server
Region
Store
memStore
StoreFile
StoreFile
Store
Region
写memstore成功立即返回
读blockcache、memstore、storefile
LSM C0树
HDFS client
Region Server
同一机器，目的？
HDFS
NameNode
DataNode
DataNode
LSM C1树

数据模型


K
V
table
row
row
row
Column-family
Column-family
Column-family
column
column
cell
cell
cell
ts1
ts2
…
…
…
SortedMap<
RowKey, List<
SortedMap<
Column, List<
Value, Timestamp
>
>
>
>

示例

- 消息表
  表结构： message [CF: message [col: autoCommit, deviceUuid, status, type]]



- http://dbidea.corp.taobao.com:8888/

message对应到RDBMS

[redacted]message信息

表名: [redacted]message　　MaxFileSize: 256 MB　　cfNum: 1个

size: 159.1 MB　　MemstoreFlushSize: 64 MB　　RegionNum: 1个

ColumnFamily信息 | Regions分布信息

视图 | 表格 | 图表 | 实时数据 | 刷新 | 帮助

	name	bloomFilter...	复制份数	压缩类型	versions	ttl	blockSize	blockcache	是否常驻内存	size
1	message	ROW	0	LZO	3个	7948800	64 KB	是	否	159.1 MB

	message			rowkey
1	autoCommit:	◆	2012-11-20 13:05:52	00001cc7d162302482b1cfff35301183_9223370683466836065_eb3d11c6a1a2784f82d...
	content:	df	2012-11-20 13:05:52	
	deviceUuid:	00001cc7d162302482b1cfff35301183	2012-11-20 13:05:52	
	digest:	er	2012-11-20 13:05:52	
	expired:	◆◆◆◆◆◆◆◆	2012-11-20 13:05:52	
	startTime:	◆◆◆◆◆◆◆◆	2012-11-20 13:05:52	
	status:		2012-11-20 13:05:52	
	taskId:	lanpa_2263	2012-11-20 13:05:52	
	type:	系统消息	2012-11-20 13:05:52	
	userInfo:		2012-11-20 13:05:52	
2	autoCommit:	◆	2012-11-19 20:53:44	00001cc7d162302482b1cfff35301183_9223370683525157304_eb3d11c6a1a2784f82d...
	content:	df	2012-11-19 20:53:44	
	deviceUuid:	00001cc7d162302482b1cfff35301183	2012-11-19 20:53:44	
	digest:	er	2012-11-19 20:53:44	
	expired:	◆◆◆◆◆◆◆◆	2012-11-19 20:53:44	
	startTime:	◆◆◆◆◆◆◆◆	2012-11-19 20:53:44	
	status:		2012-11-19 20:53:44	
	taskId:	lanpa_2222	2012-11-19 20:53:44	
		息	2012-11-19 20:53:44	

大纲

- 简介
- 数据模型
- 业务设计
- 产品线使用建议
- 容灾
- 总结

业务设计

- 适合场景 （综合考虑）
  - 表数据量大（至少亿级别以上）
  - 日志append型业务，（比如定期保留10天数据等）
  - 原则上：
    - 能分库分表来用mysql就用mysql来解决
    - mysql 单表一般500w，能使用mysql的场景
  - 无跨行跨表事务要求
  - 写入量大 (每天千万及以上)
  - 读取量相对少 (读取:写入 <= 1/10)
  - 读取场景简单、不经常变化
  - 无正序、逆序的排序要求
  - 类似dw等全量读取，不太合适。
  - Rowkey不经常更新 （必须先删除再添加）

业务设计

- 适合场景 （综合考虑）
  - 表数据量大（至少亿级别以上）
  - 日志append型业务，（比如定期保留30天数据等）
  - 无跨行跨表事务要求
  - 写入量大 (每天千万及以上)
  - 读取量相对少 (读取:写入 <= 1/10)
  - 读取场景简单、不经常变化
  - 无正序、逆序的排序要求
  - 分库分表类似dw等全量读取，不太合适。
  - Rowkey不经常更新 （否则必须先删除再添加）

- 典型的设计
  - Rowkey非单一ID （单一ID可以用mysql解决）
  - Rowkey为组合性

表名	CF属性(一行一个)	rowkey的信息
A	address latlng date	areaID_geohash_companyId 长度：6位数字 + 12位小写字母 + 小于5位数字 前面的6为数字穷举约在4000个左右

表名	列名/读取	写入	rowkey无法覆盖的查询
lA	scan（） scan的时候进行前缀匹配，rowkey中的areaId是必须的参数，设置startRow limit	set（）每天定时插入数据，数据量在1000W，**同时删除一个月前的那一天的数据**	业务查询基本上是对于rowkey的查询，只有在删除数据的时候，是根据value中的date来进行的

- 最终方案：

表名	CF列表(一行一个)	rowkey的信息
A	Info （address、latlng）	areaID_geohash_companyId 长度：6位数字 + 12位小写字母 + 小于5位数字 前面的6为数字穷举约在4000个左右

业务设计

- 适合场景 （综合考虑）
  - 表数据量大（至少亿级别以上）
  - 日志append型业务，（比如定期保留30天数据等）
  - 无跨行跨表事务要求
  - 写入量大 (每天千万及以上)
  - 读取量相对少 (读取:写入 <= 1/10)
  - 读取场景简单、不经常变化
  - 无正序、逆序的排序要求
  - 分库分表类似dw等全量读取，不太合适。
  - Rowkey不经常更新 （否则必须先删除再添加）

- 典型的设计
  - 交互性的应用消息
  - 数据双写，当做索引（类似买卖家）

Rowkey	Column Value
fromID + time	Toid: ***;content:***

查找穆公最近一周发布的消息？
查找穆公最近一周发送给马云的消息？
发送给马云的消息？

业务设计

- 适合场景 （综合考虑）
  - 表数据量大（至少亿级别以上）
  - 日志append型业务
  - 无跨行跨表事务要求
  - 写入量大 (每天千万及以上)
  - 读取量相对少 (读取:写入 <= 1/10)
  - 读取场景简单、不经常变化
  - 无正序、逆序的排序要求
  - 分库分表类似dw等全量读取，不太合适。
  - Rowkey不经常更新 （否则必须先删除再添加）

- 表结构设计
  id是订单ID，可以是业务主键

Rowkey	Column Value
Id	valueString (93 fields)

默认用户订单索引表

Rowkey	Column Value
UId + time + id	valueString (32 fields)

能否覆盖查询：穆公一个星期内买的商品？ 穆公一个月买的书？

业务设计

- 适合场景 （综合考虑）
  - 表数据量大（至少亿级别以上）
  - 日志append型业务
  - 无跨行跨表事务要求
  - 写入量大 (每天千万及以上)
  - 读取量相对少 (读取:写入 <= 1/10)
  - 读取场景简单、不经常变化
  - 无正序、逆序的排序要求 （单向时间排序ok）
  - 分库分表类似dw等全量读取，不太合适。
  - Rowkey不经常更新 （否则必须先删除再添加）
  - 特殊的搜索固定需求

- 分词索引表

Rowkey	Column Value
Uld + 分词 + time + id	null

能否覆盖查询：穆公一个星期内买的商品？ 穆公一个月买的书？

- 结论：
  - 覆盖搜索场景、无法用数据库解决
  - 查询类型固定
  - 只会按照时间排序
  - 永久的大表保存
  - 数据一致性要求低

Rowkey	Column Value
UID+ time + id	null
UID+ 分词 + time + id	null

业务设计-无结构化数据

- SchemaFree业务

Rowkey	Column Value
id	列数不定，有的有2个列；有的有N个

查询结果(hbase耗时:7ms,全部耗时:170ms)

	rowkey	CF		
		login_90d_ip_cnt:	2	2012-11-19 10:14:52
		login_90d_max_city:	智利	2012-11-19 10:14:52
		login_90d_max_country:	CL	2012-11-19 10:14:52
		login_90d_max_country_cnt:	2	2012-11-19 10:14:52
		login_90d_max_ip:	186.9.114.17	2012-11-19 10:14:52
		login_90d_max_ip_cnt:	1	2012-11-19 10:14:52
		login_90d_max_language:	es	2012-11-19 10:14:52
		login_90d_xman_cnt:	2	2012-11-19 10:14:52
		member:	132000000	2012-11-19 10:14:52
25	000000321_member_9223370683383565993	member:	123000000	2012-11-21 12:13:29
26	000000321_member_9223370683468441300	member:	123000000	2012-11-20 12:38:54
27	000000321_member_9223370683471726604	member:	123000000	2012-11-20 11:44:09
28	000000321_member_9223370683563612849	member:	123000000	2012-11-19 10:12:43
29	000000331_member_9223370683383114603	login_90d_city_cnt:	1	2012-11-21 12:21:01
		login_90d_cnt:	3	2012-11-21 12:21:01
		login_90d_country_cnt:	1	2012-11-21 12:21:01
		login_90d_days:	1	2012-11-21 12:21:01
		login_90d_emaillog_cnt:	1	2012-11-21 12:21:01
		login_90d_have_ip_cnt:	2	2012-11-21 12:21:01
		login_90d_have_ipcountry_cnt:	2	2012-11-21 12:21:01
		login_90d_ip_cnt:	1	2012-11-21 12:21:01
		login_90d_max_city:	英国	2012-11-21 12:21:01
		login_90d_max_country:	GB	2012-11-21 12:21:01
		login_90d_max_country_cnt:	2	2012-11-21 12:21:01
		login_90d_max_ip:	90.194.148.35	2012-11-21 12:21:01
		login_90d_max_ip_cnt:	2	2012-11-21 12:21:01

大纲

- 简介
- 数据模型
- 业务设计
- 产品线使用建议
- 容灾
- 总结

产品线、客户端使用建议

- 海量数据，rowkey范围和分布已知，建议进行预分配
- Rowkey一定要尽量短 （如：时间用时间戳整数表示、编码压缩）
- CF设计：尽量少，建议CF数量在1-2个

  flush和compaction是region级别；某个CF引发其它CF的memstore的flush→大量store file →导致compaction(当store file的个数>value)

  问题： 还有其他的原因么？ 1CF -> 6CF
- Rowkey设计：写入要分散；如历史交易订单：biz_order_id做reverse后做rowkey

产品线、客户端使用建议(2)

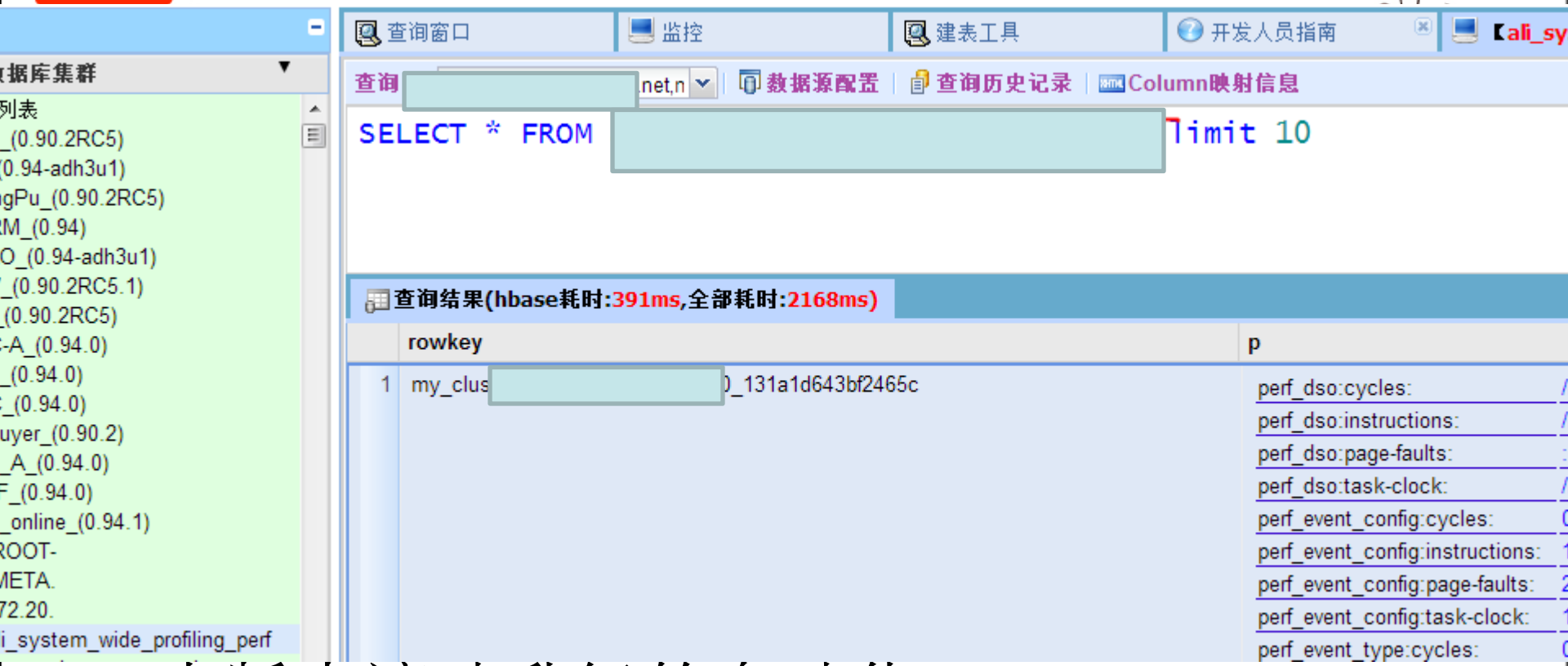
- Autoflush参数设置为true；否则极端情况下会丢失数据
- Hbase client的重试次数为3次以上。否则会由于split导致region not onle；从而导致写入失败(udc集群出现过)。
  - hbase.rpc.timeout　　一次rpc的timeout；默认60秒
  - hbase.client.pause　　客户端的一次操作失败，到下次重试之间的等待时间
  - hbase.client.retries.number　　客户端重试的次数
  - hbase.regionserver.lease.period　客户端租期超时阀值；scan量大时可以考虑增大；否则”Lease Exception: lease -7000000000000001 does not exist”

产品线、客户端使用建议(3)

- ZK连接/HTable对象的使用注意
  - Configure对象的使用
    - 必须是static or singleton模式
  - 默认：每台机器与zk直接的连接数不超过30个
  - HTable的使用
    - 线程不安全
    - 使用HTableV2
    - HTablePool (推荐的方式)

业务接入Ork平台 http://nosql.*.com

自动运维平台



- 小版本滚动升级的自动化：region move -> region为0 -> 本地升级
- 大版本升级：关闭split -> distcp -> 解析hlog -> 自定义同步增量（平滑迁移）

大纲

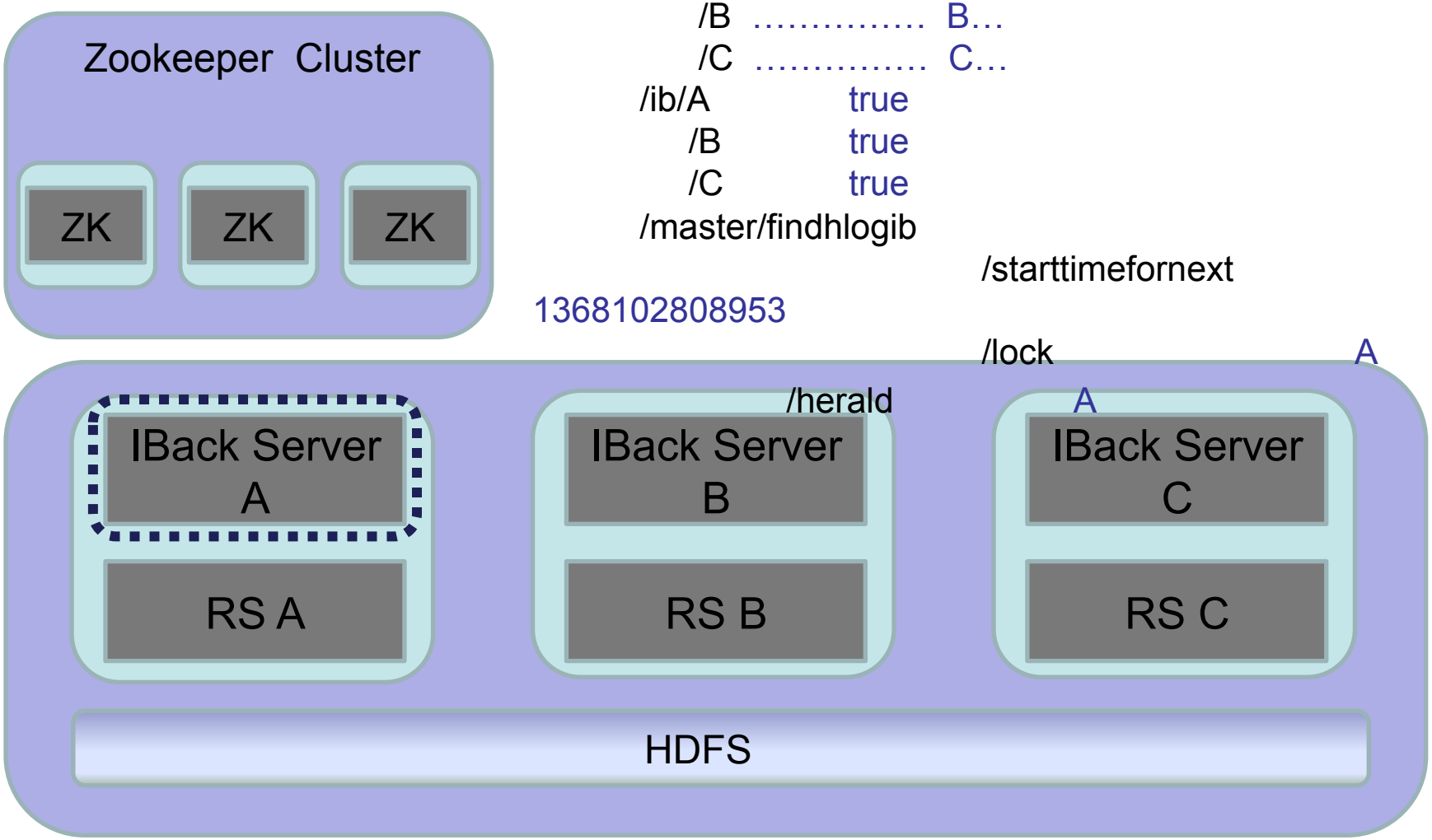
- 简介
- 数据模型
- 业务设计
- 产品线使用建议
- 容灾
- 总结

目前的主要问题

- regionserver的单点问题
  - 导致部分数据短暂不可用
- 跨机房容灾
  - 大部分还是部署在单个机房
  - 跨机房性能衰减
- 实现：
  - 程序双写
  - 消息中间件实现(异步消息)
  - **复制的方案(后端研发团队已经有replication的方案，支持push方式) + 数据补齐iback**

IBack 架构

/iback
/config/tables
qianzhen_iback_test,qianzhen_iback_test2
/rs/A/hlog1 path and position
/ hlog2 path and position
/lock A
/B ..... B…
/C ..... C…
/ib/A true
/B true
/C true
/master/findhlogib
/starttimefornext
1368102808953
/lock A
/herald A



IBack 组件

1. IBack Server

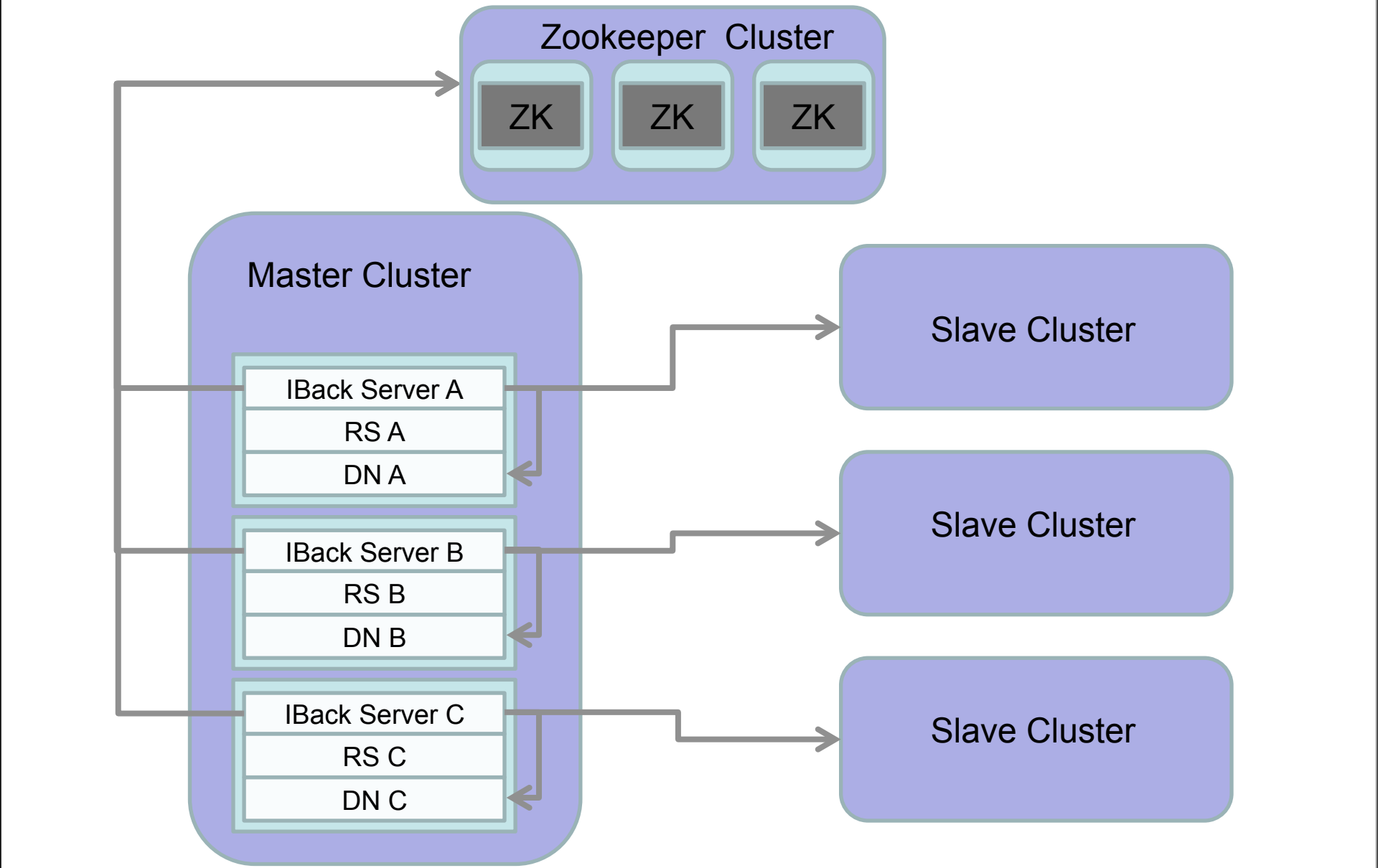
主要用来解析并处理HLog,并将解析的结果写到指定的集群。IBack既可以部署在Master集群上，也可以部署Slave集群上。在IBack集群内部会自组织形成一个虚拟Master IBack Server,有三个主要功能：

a. 发现新生成的Hlog并加入Zookeeper.
b. 将没有IBack server 处理的RS分配给相应的IBack server.
c. 执行动态配置相关参数

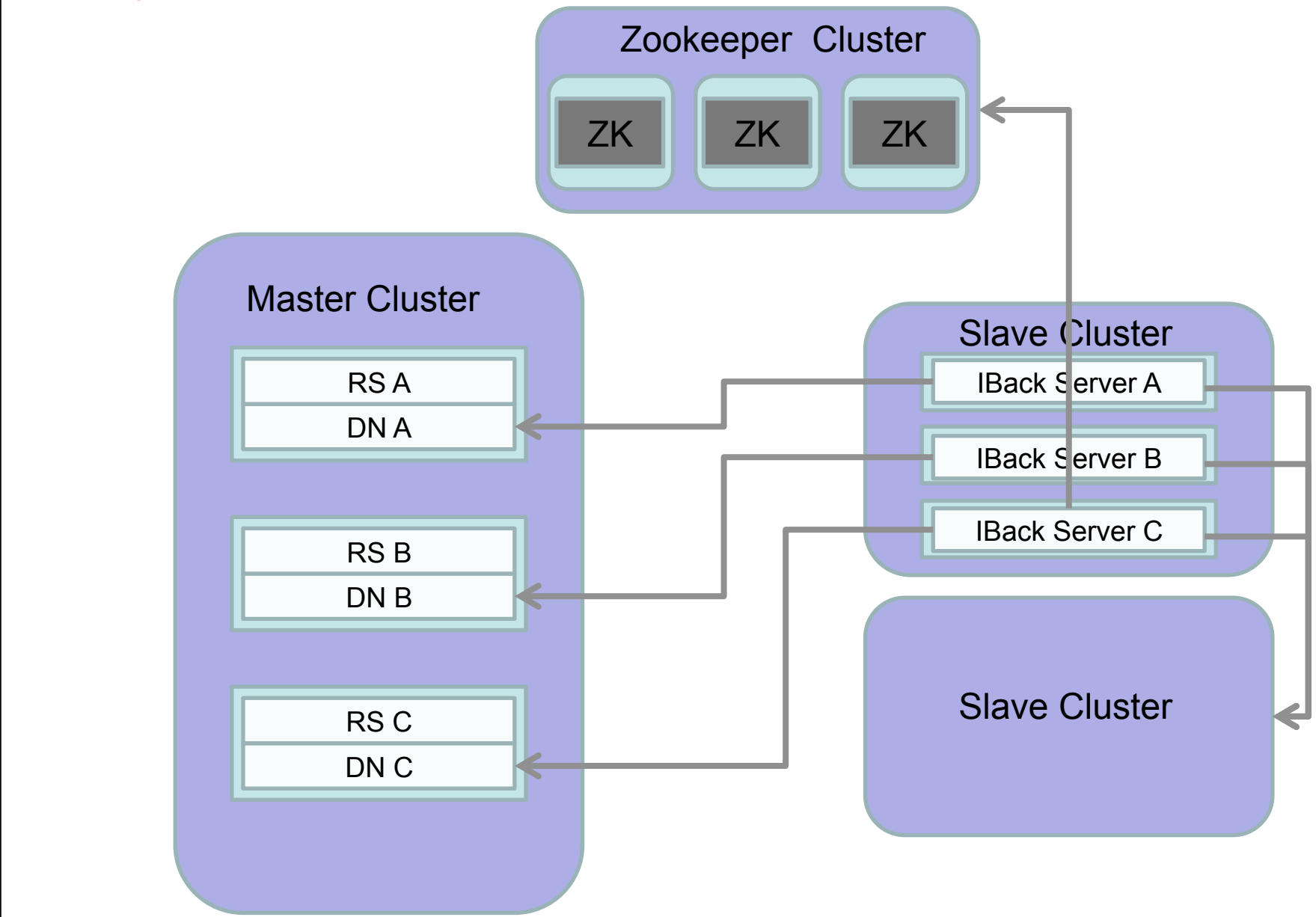
2. Zookeeper集群

ZK集群主要用于提供分布式的参数协同，IBack是无状态的，状态是保存在ZK上的，包括现在的有哪些IBack server、有哪些RS、配置信息、IBack的集群管理、IBack处理Hlog相关的参数，对账开关等。

IBack Push 方式



IBack Pull 方式



总结

- Hbase作为一个NOSQL存储，作为在线存储的一个重要组成
- 业务设计和选型尤为重要，依特性合理使用
- 容灾是走出Hbase存储使用更广阔的的前提
- 异常处理，先恢复服务，再深入排查

Thanks!
Q&A

穆公/朱金清
微博：淘穆公
http://www.weibo.com/**suinking**