BDTC 2016中国大数据技术大会
Big Data Technology Conference 2016

小米融合云实践

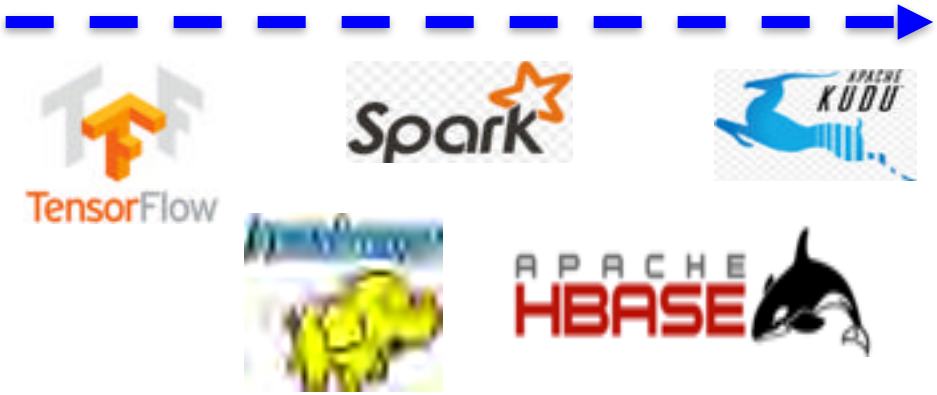
小米融合云组

融合云动机

- 业务快速发展
  - 管理成本
  - 接入复杂
  - 隔离需求
- 生态链需求
  - 公网访问

小米内部业务

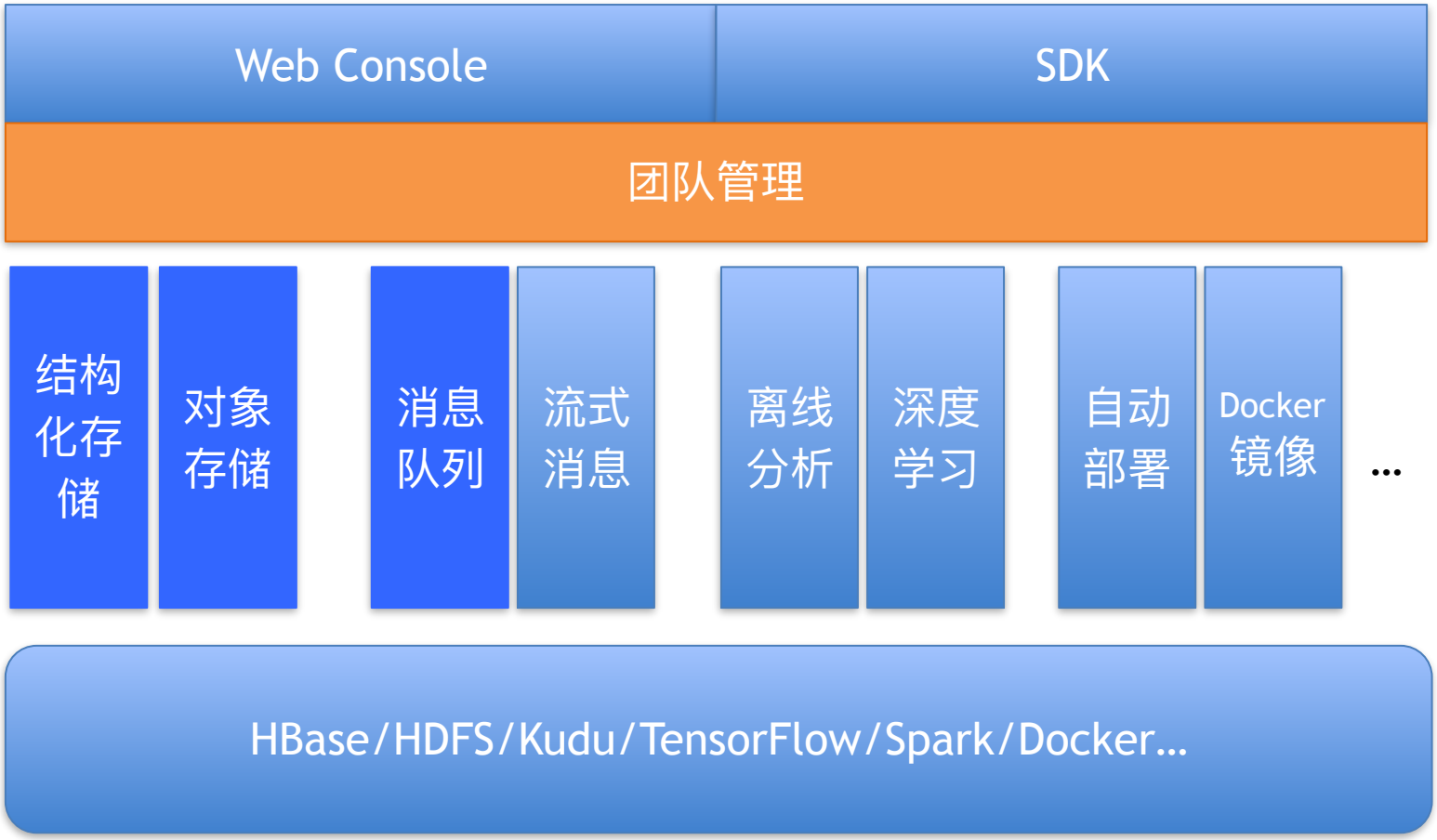
生态链业务



融合云设计

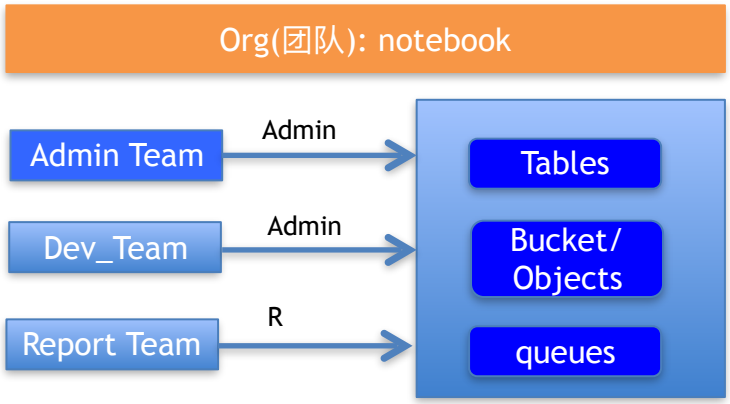
- 服务化
  - 统一身份认证
  - 权限管理
  - 多租户
  - Web console
- 基于开源
- 功能扩展
  - 基于HBase做二级索引
- 系统扩展
  - 基于HBase和HDFS做对象存储
- 公网调用支持
  - Http请求/无状态

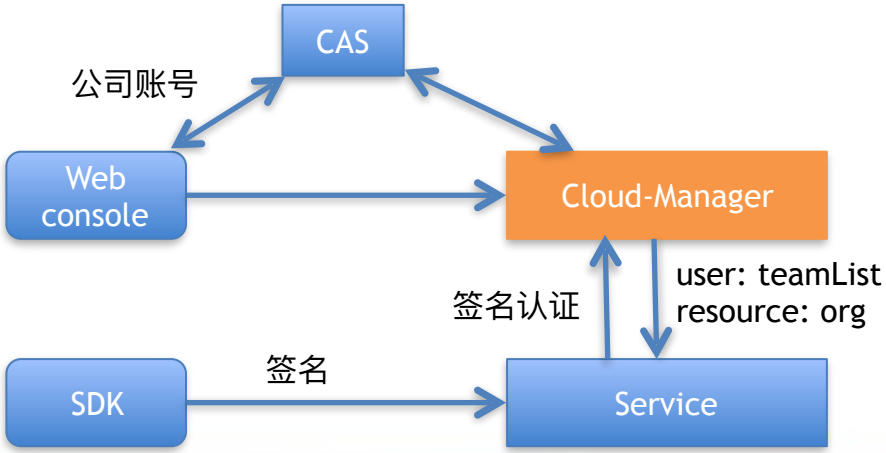
融合云架构


Web Console
SDK
团队管理
结构化存储
对象存储
消息队列
流式消息
离线分析
深度学习
自动部署
Docker镜像
...
HBase/HDFS/Kudu/TensorFlow/Spark/Docker...

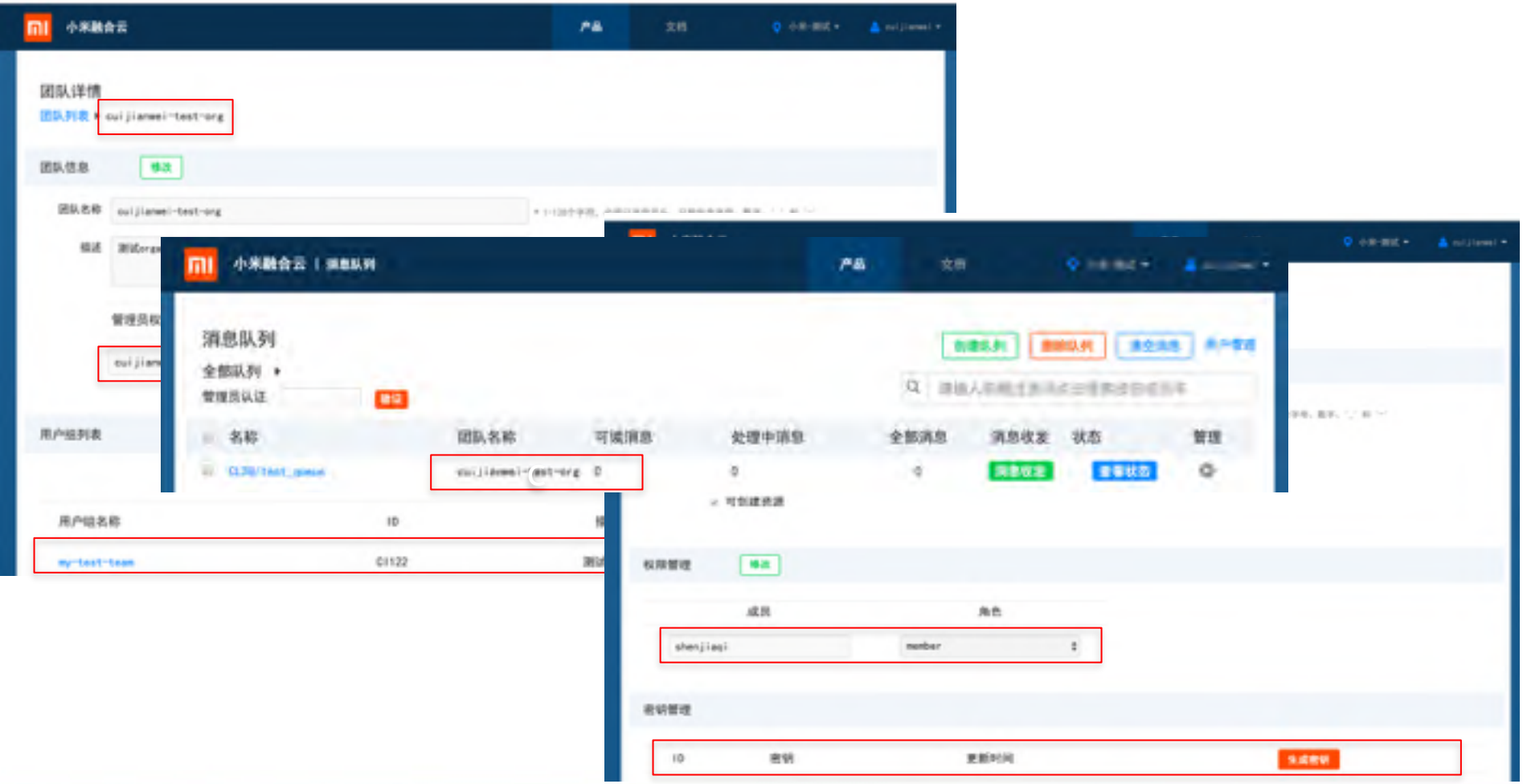
团队管理(Cloud-Manager)

- Organization(Org)
  - 资源隶属
- Team
  - 用户身份
  - 隶属于Org
  - Admin/Member
  - Org admin Team
- 认证
  - CAS
  - 签名认证
    - 生成秘钥
    - 秘钥轮换
- 权限管理
  - <Team, Resource, Perm>

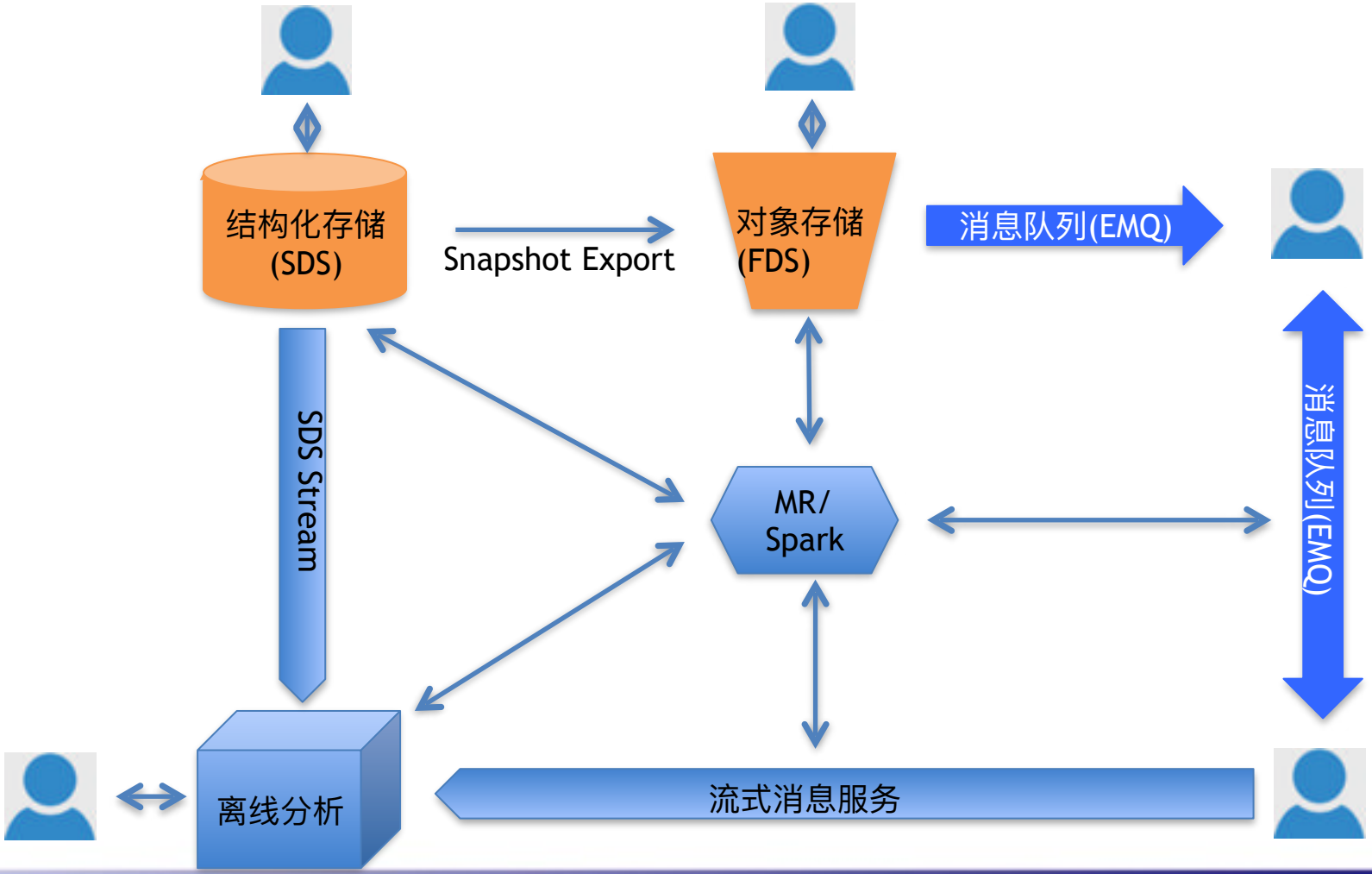




示例


小米融合云
产品
文档
团队详情
团队信息
团队名称
消息队列
全部队列
名称
团队名称
可读消息
处理中消息
全部消息
消息收发
状态
管理
用户组列表
用户组名称
ID
my-test-team
权限管理
成员
角色
shenjiaqi
member
密钥管理
生成密钥

存储/计算数据流

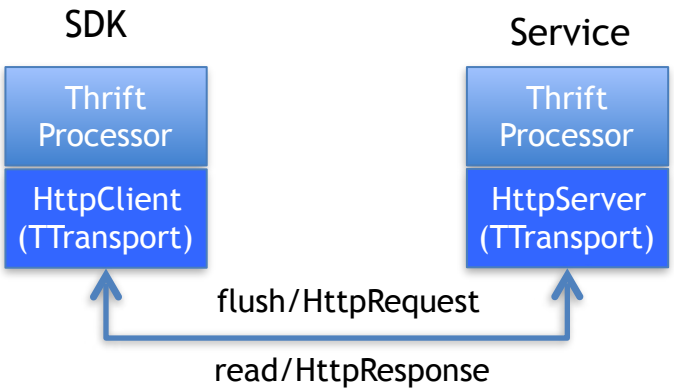

结构化存储
(SDS)
Snapshot Export
对象存储
(FDS)
消息队列(EMQ)
SDS Stream
MR/
Spark
消息队列(EMQ)
离线分析
流式消息服务

Structure Data Store(SDS)

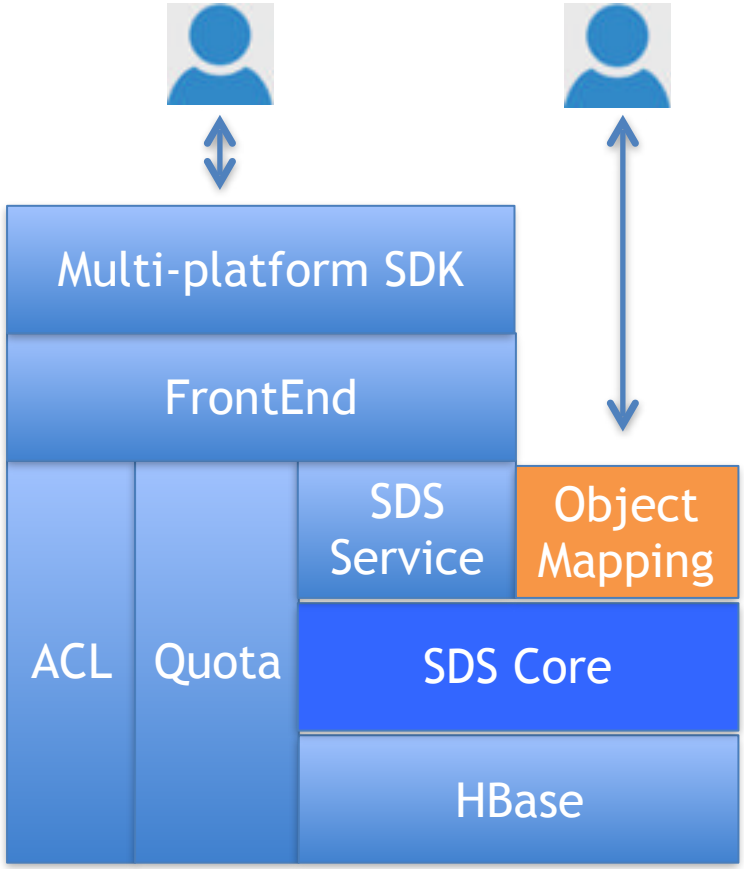
- 分布式数据库
- 基于HBase
- 数据模型扩展
  - 支持EntityGroup/Primary Key
  - 支持数据类型
- 多语言SDK支持
- ACL扩展
- 功能扩展
  - 局部二级索引
  - 全局二级索引
  - Stream
  - 实时恢复

SDS架构

- SDK
  - Java/php/python/go/c++/node.js

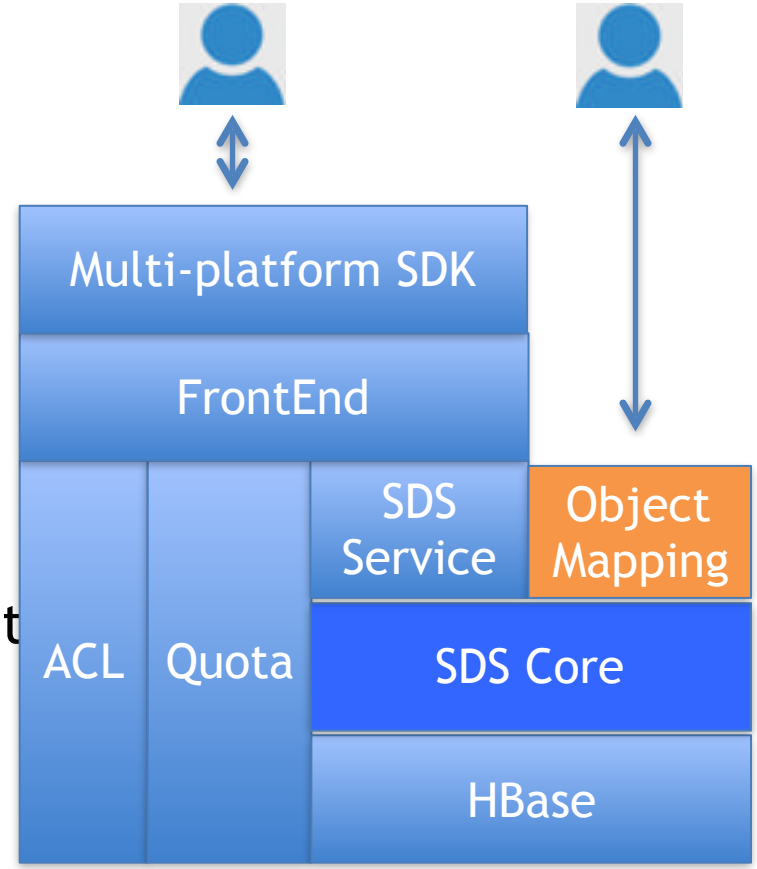


- SDS Service
  - 认证
    - 与Cloud-Manager交互
  - 元信息管理
    - UserTable => TableId => HBase Table



SDS架构

- SDS Core
  - SDS Request <-> HBase Request
- Object Mapping
  - Java Object <->SDS Request
- ACL
  - 表级别权限控制
  - EntityGroup间权限隔离
- Quota
  - <grantee, table, operation, limit
  - TokenBucket算法
  - Soft Limit
    - 引入集群过载quota
    - 集群过载前允许超发

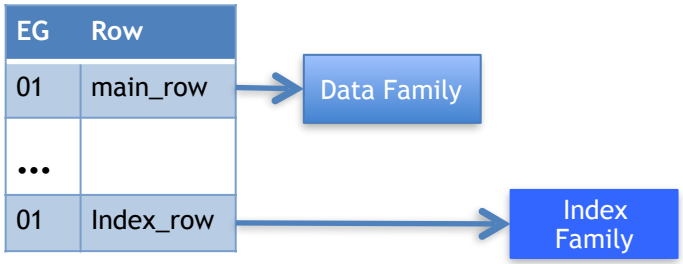


SDS Core主要功能

- 实现Entity Group(EG)功能
  - 业务特性: 以用户为中心
  - 定义Entity Group Keys
  - EG在同一个region
  - EG内跨行写原子性
  - EG内局部二级索引
    - 强一致
- Schema管理
  - EntityGroup/Primary Index/Secondary Index/Column
- 数据类型
  - Bool/Int8/Int16/Float/Double/String/Binary, Set
  - 使用OrderedBinary编码确保顺序
- 条件查询

二级索引

- 局部二级索引
  - 数据Family/索引Family
  - HBase coprocessor中更新索引
    - 读取主记录
    - 构造索引rowkey
    - 更新主记录/删除旧索引/写入新索引
  - 支持Eager/Lazy/Immutable Index
    - Lazy invalid index定期清理
  - Eager/Immutable Index支持属性投影
- 全局二级索引
  - 使用单独的HBase table存储索引
  - 使用Themis提供HBase跨表更新原子性
    - https://github.com/xiaomi/themis
    - 全局单调递增时间戳: https://github.com/xiaomi/chronos
    - 强一致性



Object Mapping对象映射

- 使用Java annotation和反射实现
  - toRecord: Object => Map<String, Object>
  - fromRecord: Map<String, Object> => Object

```
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.FIELD)
public @interface Column {
  String name() defau
  String fullname() d
  DataType type() def
  boolean collection(
  Class elementClass(
  SerializationType s
  boolean keyOnly() d
  String family() def
```

```
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.ANNOTATION_TYPE)
public @interface LocalSecondaryIndex {
  String name();

  String family();

  Key[] columns();

  String[] projections() default { };

  SecondaryIndexConsistencyMode mode() default SecondaryIndexConsistencyMode.LAZY;

  boolean unique() default false;
}
```

Table Schema定义(SQL)

Example: cloud notebook

```
-- Equivalent SQL definition
CREATE TABLE note (
  userId VARCHAR(64) NOT NULL, -- Entity group key
  noteId INT8 NOT NULL, -- Primary key
  title VARCHAR(256),
  body VARCHAR(2048),
  mtime BIGINT,
  tag VARCHAR(16),
  version INT,

  PRIMARY KEY(userId, noteId),
  INDEX(userId, mtime),
  INDEX(userId, tag)
);
```

Table Schema定义

```
@Record(table = "note", family = "B")
public class Note {
  @Column(keyOnly = true)
  String uid; // user ID

  @Column(keyOnly = true)
  Long id; // note ID

  @Column String title;

  @Column(serialization = Column.SerializationType.UNIX_TIME)
  private Date mtime;

  @Column(collection = true, elementClass = String.class, type =
  private Set<String> tags;
```

CF	Rowkey	Values
B	hash(userId) userId noteId	title, mtime, tags, body, version
I	hash(userId) userId idx-mtime mtime noteId	title[1]
I	hash(userId) userId idx-tags tag1 noteId	title
I	hash(userId) userId idx-tags tag2 noteId	title

查询

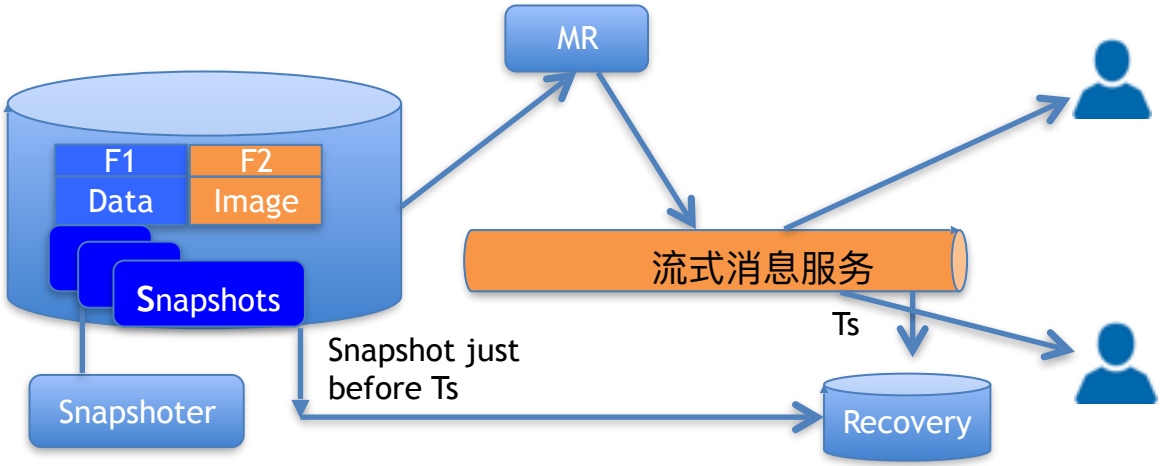
```
// random read
@Override public Note findNoteById(String userId, long nid) {
  Note key = new Note(userId, nid, null, null, null, ...);
  return typedAccessClient.get(key);
}

// range query, same as SELECT * FROM note
//                       WHERE uid=userId AND title LIKE 'Test%'
//                       ORDER BY mtime DESC LIMIT N
@Override public List<ListViewItem> searchNLatestItems(
    String userId, int N, String title) {
  return typedAccessClient.scan(Note.class,
         ListViewItem.class,
         Constants.IDX_MTIME, // implicitly specify index name
         Note.entityGroupNote(userId),
         Note.entityGroupNote(userId),
         "title␣REGEX␣'" + match + "'",//title REGEX 'Test.*'

         N).getRecords();
```

查询

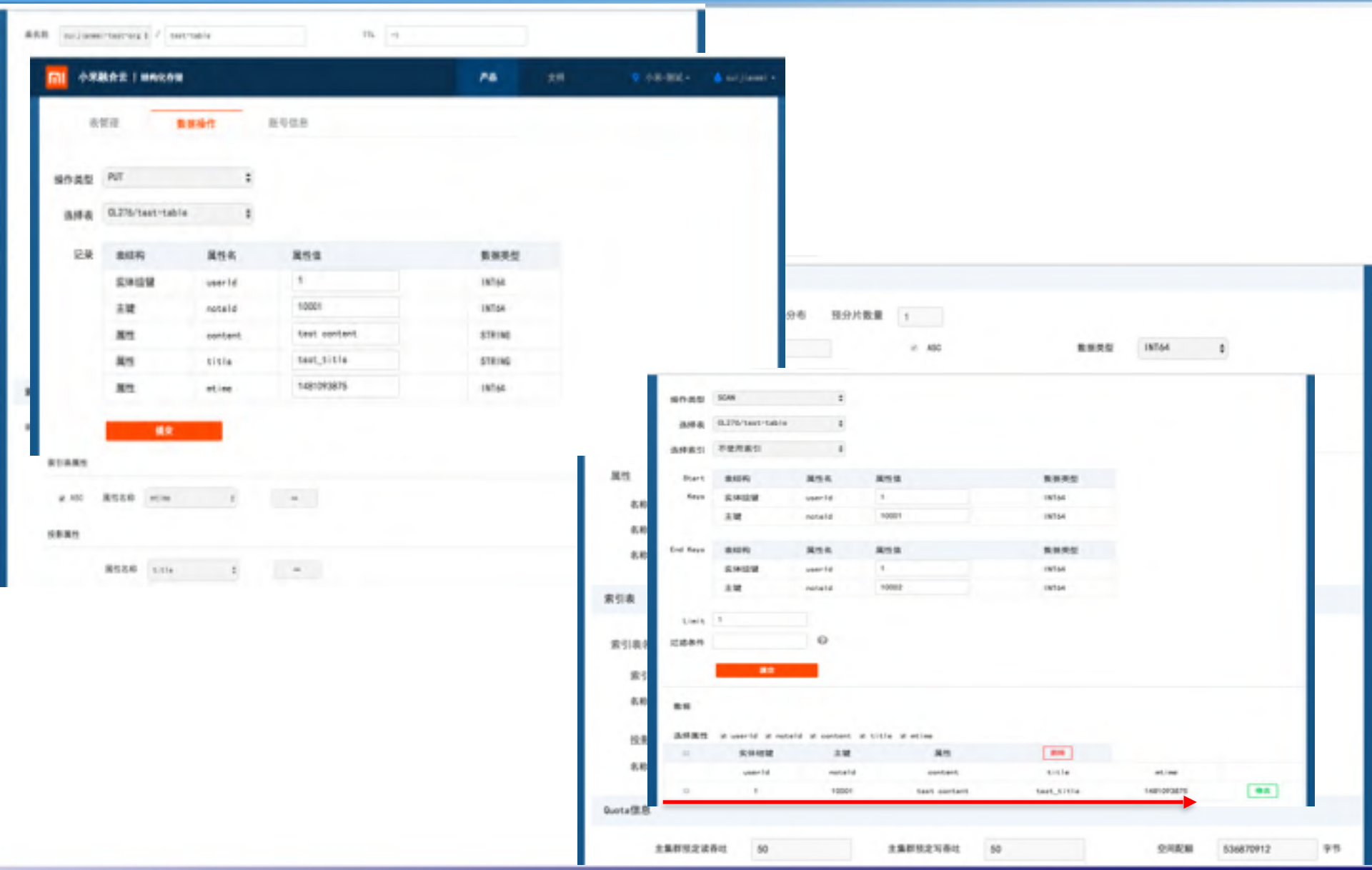
```
// same as UPDATE note SET version = oldVersion + 1,
//                 mtime = NOW, contents = '...'
//            WHERE version = oldVersion
//            AND uid = userId AND id = noteId
@Override public boolean updateNote(Note note) {
  int currentVersion = note.getVersion();
  try {
    SimpleCondition versionPredicate =
        SimpleCondition.predicate(note.getVersion(),
            CompareFilter.CompareOp.EQUAL,
            Constants.VERSION_FIELD);
    note.setMtime(new Date());
    note.setVersion(currentVersion + 1);
    return typedAccessClient.put(note, versionPredicate);
  } finally {
    note.setVersion(currentVersion);
  }
```

Stream

- 场景
  - OLAP
  - 增量数据处理
- 原理
  - Stream Family
    - 存储image
    - 使用hbase coprocessor build images
  - 推送image到流式消息服务
- 实时恢复



Web console示例



性能测试

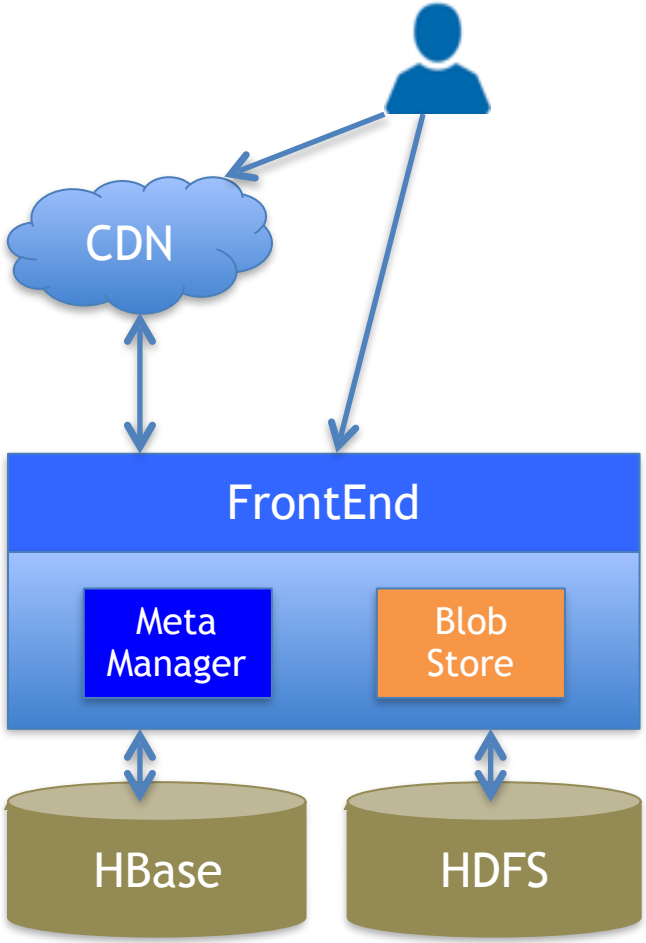
- SDS rest server
  - 24 Intel(R) Xeon(R) E5620 @2.4 GHz, 64G, 万兆网
- HBase region server
  - Cpu/内存/网络同上，8 * 800G ssd，5台
-

Put Througput/Latency				
记录大小	Througput	Avg(ms)	P95(ms)	P99(ms)
0.1k	33738	0.52	7	9
0.5k	30047	0.46	6	11
1k	25652	0.46	6	10

Get Througput/Latency				
记录大小	Througput	Avg(ms)	P95(ms)	P99(ms)
0.1k	30213	0.26	4	7
0.5k	27457	0.28	3	9
1k	26932	0.29	3	12

File Data Store(FDS)

- KB到TB级别的文件
- 基于HBase & HDFS
- Bucket & Object
- Restful API
- 多语言SDK
- 权限管理
- 事件通知
- Quota管理
- 30+业务使用

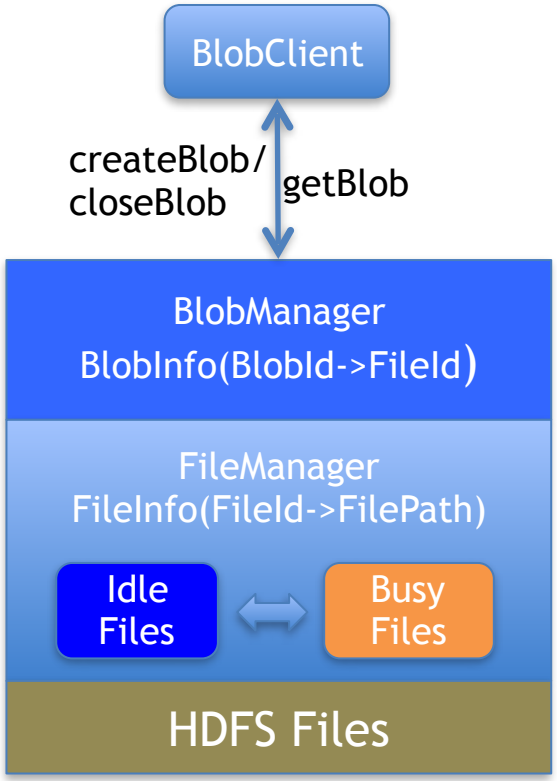


元数据管理

- Bucket Table
  - Bucket基本信息/ACL/Quota
- Object Table
  - Object基本信息/ACL/meta/通知信息
  - Object name => blobUri
- BlobInfo Table
  - BlobUri => FileId
  - 抽象Blob write/read/get接口，隐藏具体实现
- FileInfo Table
  - FileId => Path, basicInfo
- MultiPart Upload Table
  - uploadId => partInfo

BlobStore

- BlobManager/FileManager
  - Blob存储
    - magic/data/md5
    - offset
  - 管理打开文件
    - MaxOpenNum/MaxFileSize
    - Idle Files <=> Busy Files
- BlobClient
  - 创建Blob: BlobWriter
    - Wrapped OutputStream
  - 关闭Blob
  - 获取Blob: BlobReader
    - Wrapped InputStream

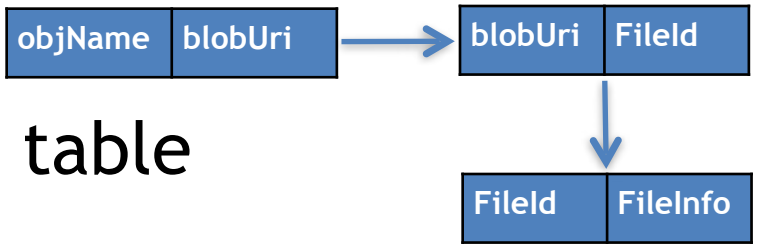


对象操作

- 创建对象
  - BlobClient: createBlob
  - 将object信息写入hbase object table
- 读取对象
  - 从hbase object table获取blobUri
  - BlobClient: getBlob
- 对象列表
  - 扫描hbase object table
- 删除对象
  - 删除hbase object table中对应记录

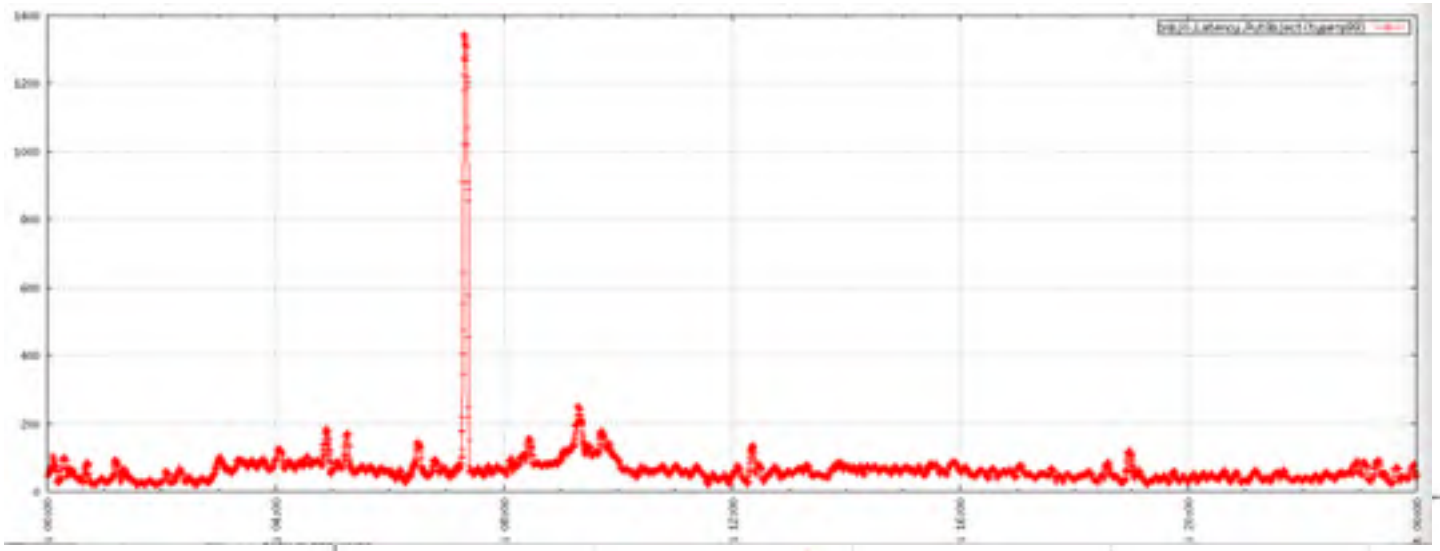
Cleaner & Compactor

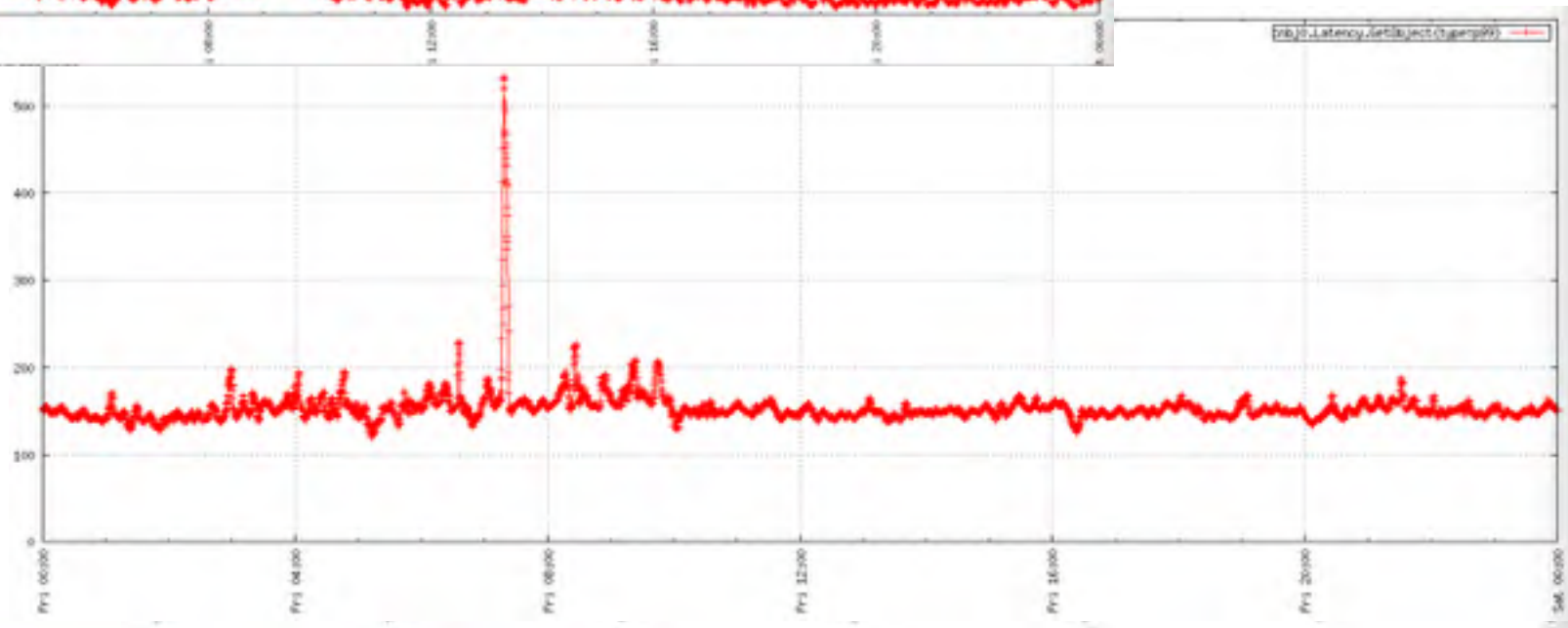
- Object Cleaner
  - 根据object table删除blob table



- File Cleaner & Compactor
  - 基于BlobInfo Table和FileInfo Table
  - 根据BlobInfo Table统计File使用率
  - 归并低使用率文件
  - Archive无Blob引用文件
  - 删除过期Archive文件

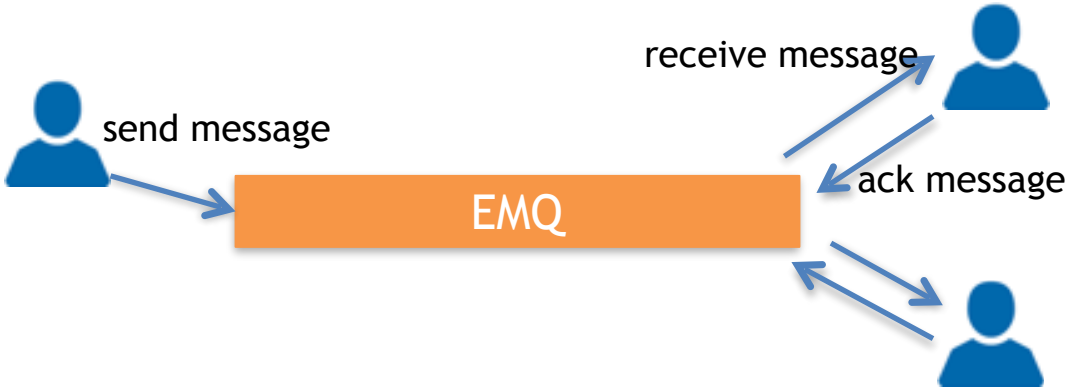
FDS性能




500
400
300
200
100
0
Fri 00:00
Fri 04:00
Fri 08:00
Fri 12:00
Fri 16:00
Fri 20:00
Sat 00:00

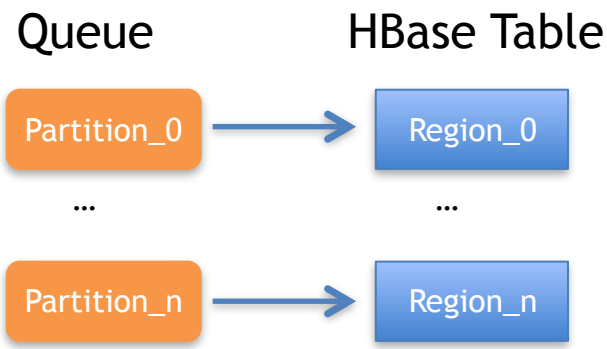
Elastic Message Queue(EMQ)

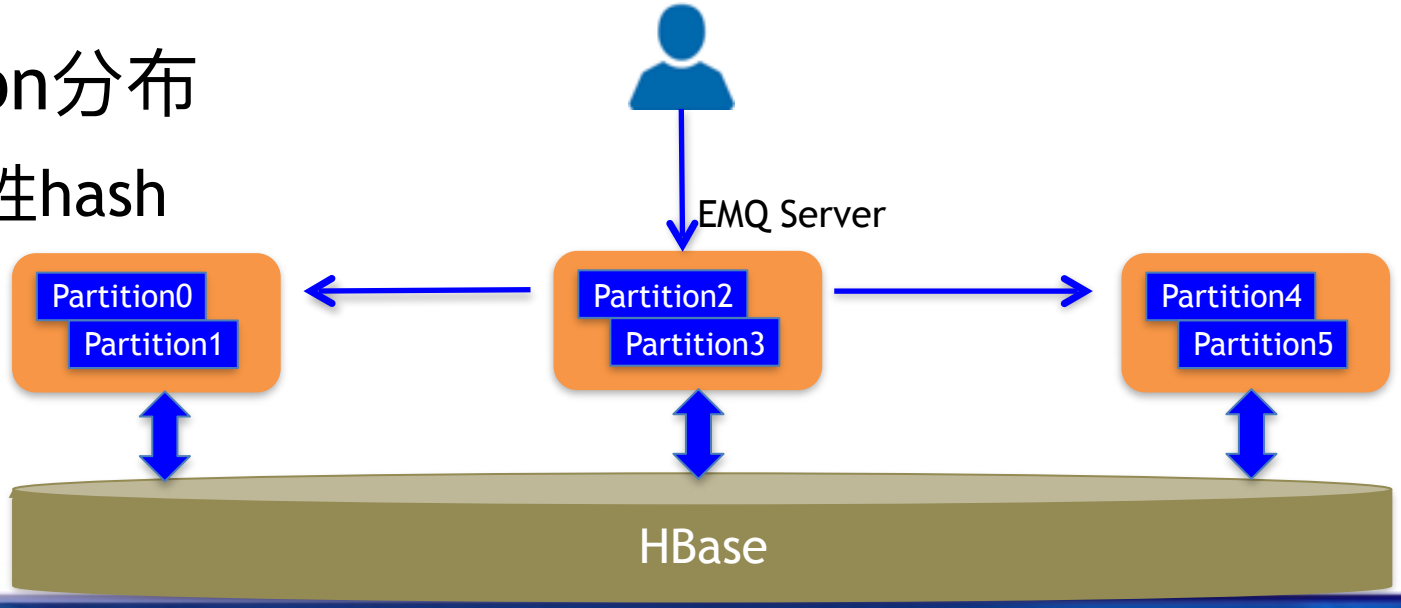
- 应用间异步通信
- 支持P2P, Pub/Sub两种模式
- 基于HBase提供分布式能力
- At least once语义
- 多读者
- 消息优先级
- 死信队列
- Short/Long Pooling



EMQ架构

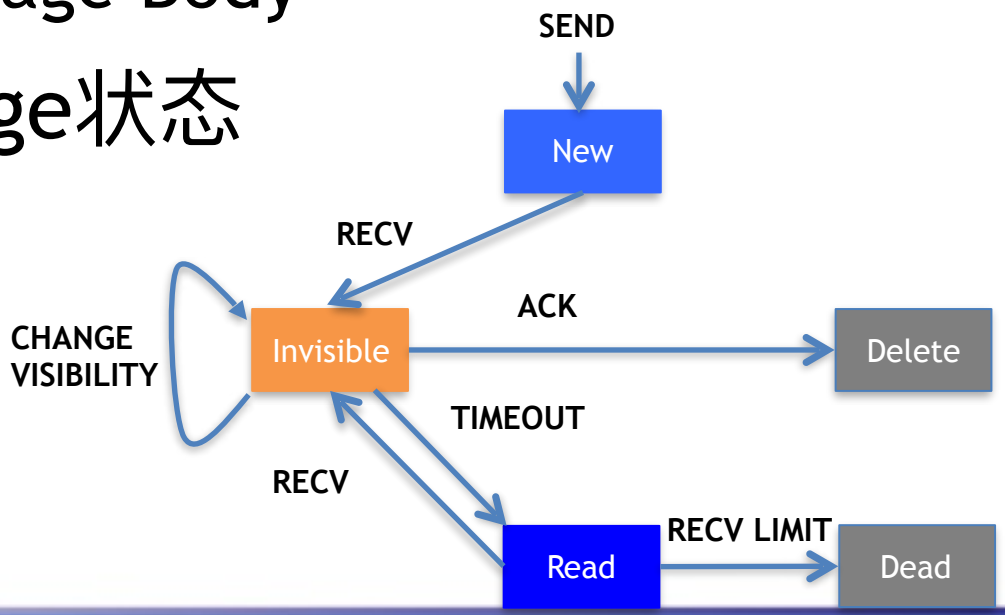
- Queue
  - HBase Table
- Partitions
  - PartitionId
  - Partition分布
    - 一致性hash





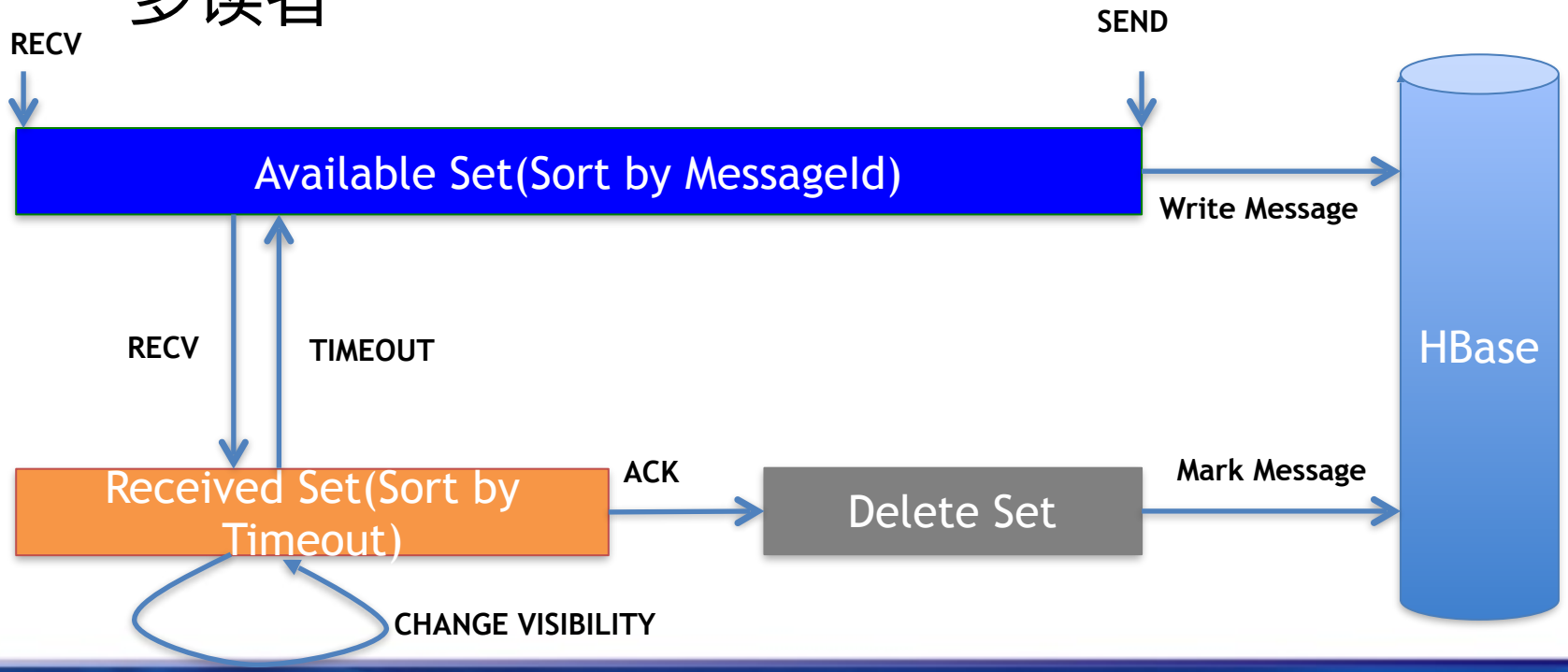
消息状态

- Message
  - MessageId
    - messageId = hbaseRow = <partitonId><timestamp><sequenceId>
    - Timestamp = currentTime + delayTime
  - Message Body
- Message状态



Partition内存状态

- 内存信息
  - Available set/Received set/Delete set
  - 多读者



主要操作

- 发送消息
  - 批量发送
- 读取消息
- 消息删除
  - MsgIdSet：MsgIds - AckedMsgIds - DeadMsgIds
  - StartPoint=min(MsgIdSet)
- 消息加载
  - 死信队列


THANK YOU