# 支付宝数据平台及应用

## Hadoop in china

蒋杰(花名:平原君)

2011-12-2

# 目录


1.Hadoop在支付宝相关应用

2.经验分享—血的教训

3.面临问题:海量数据

4.支付宝数据平台整体架构

4.重点产品介绍—海狗系统

# 目 录

第一篇:Hadoop在支付宝相关应用

# 案例1:Hbase相关--历史消费记录查询

- 客户需求:

1.支持海量数据（总数据30T）下的快速随机读取

2.支持按userid的快速数据导出

3.支持多个字段的分词查询

- 实现结果:

1.按单个数据的查询响应在10ms以内

2.按多个分词的查询并且支持分页的平均响应在40ms左右

# 案例2:Hbase相关--CTU风险数据项目

## 客户需求:

1.支持宽表的字段灵活变更和数据快速

2.支持ctu风控模型海量数据（40TB）的在线高并发读写每天10亿次的调用量，高峰期读是5.4W/S 写 10W/s

## 实现结果:

98%的读请求在10ms内完成， 95%的写请求在10ms内完成

# 案例3:hadoop应用—hadoop集群数据的资源管理

## 海豚系统

**一站式资源服务**

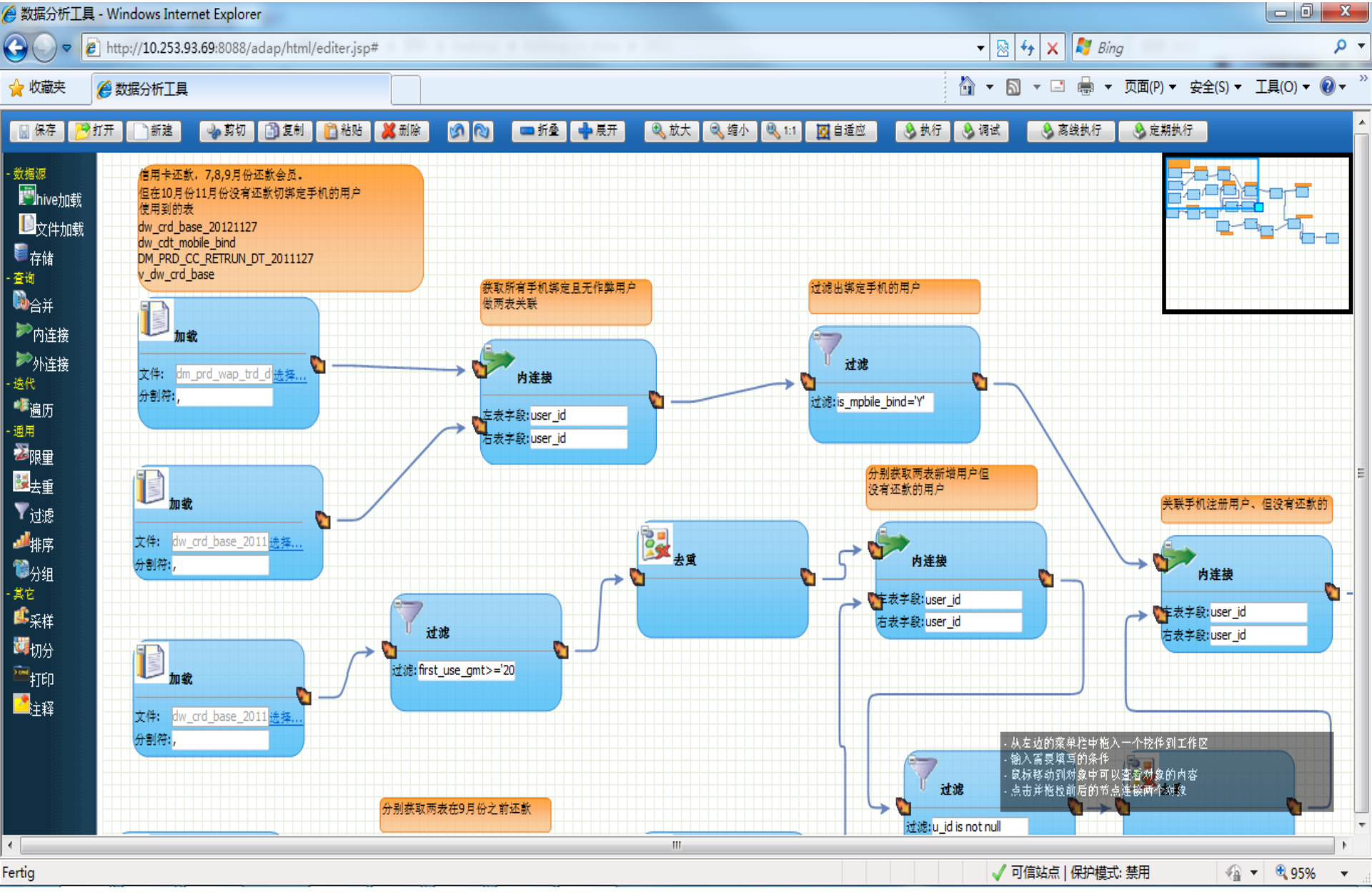
1:访问 http://adc.alipay.com/

2:通过公司的域帐号登录

3:申请计算存储资源，获得批准

4:通过客户端访问集群资源

# 案例4:Pig相关—可视化用户自主查询



# 目录

第二篇:经验分享—血的教训

# 经验分享(一):Hbase相关优化

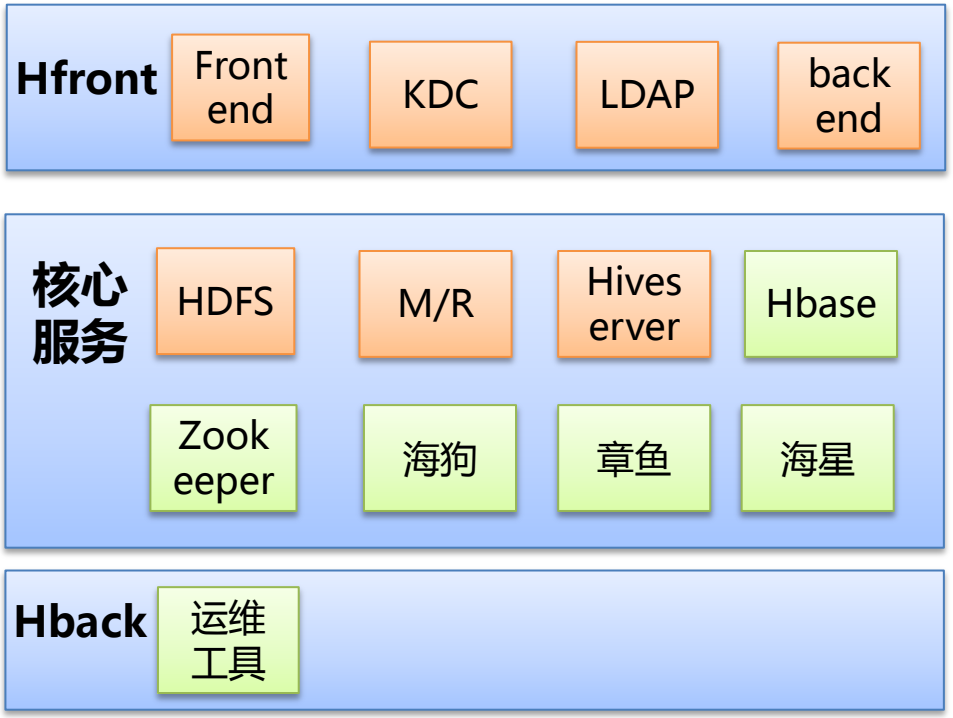
## 历史消费记录查询项目

- ✓ 优化hregion下的minor compact算法，加快minor compact的速度
- ✓ 优化客户端的多个get的查询请求速度
- ✓ 设计合理的blocksize，支持快速的随机读和顺序读

## CTU风险数据项目

- ✓ 合理的设计rowkey，将数据平均分散到各台hregionServer，避免数据热点
- ✓ 合理设计合适的hregion大小，避免split和compact造成的响应时间波动
- ✓ 解决高并发写请求，单个regionserver发生写请求挂住的bug

# 经验分享(二):海豚系统




## 易用

- ✓ 基于kerberos的用户认证
- ✓ 基于ldap的服务端组关系解析
- ✓ 用户执行空间/存储空间隔离

## 安全

- ✓ 通过登录WebUI接口开始使用集群资源
- ✓ 一站式注册、申请资源、管理资源
- ✓ HDFS/MR/Hive/HBase多种类型的资源服务化

## 高效

- ✓ 资源服务化，开发成本低(取消gateway存在的必要，节省数10台机器)
- ✓ 数据和处理越靠越近，整体效率高(自动化添加帐号、管理资源操作，节省管理员和用户时间,同时降低手工操作疏忽引发的故障几率)
- ✓ 提供计算和存储成本明细，有助于用户降低成本，我们优化节约成本

# 目录

第三篇:面临问题:海量数据

# 支付宝---数据云


Alipay Data Cloud (ADC)
数据分析
海量
适时
数据

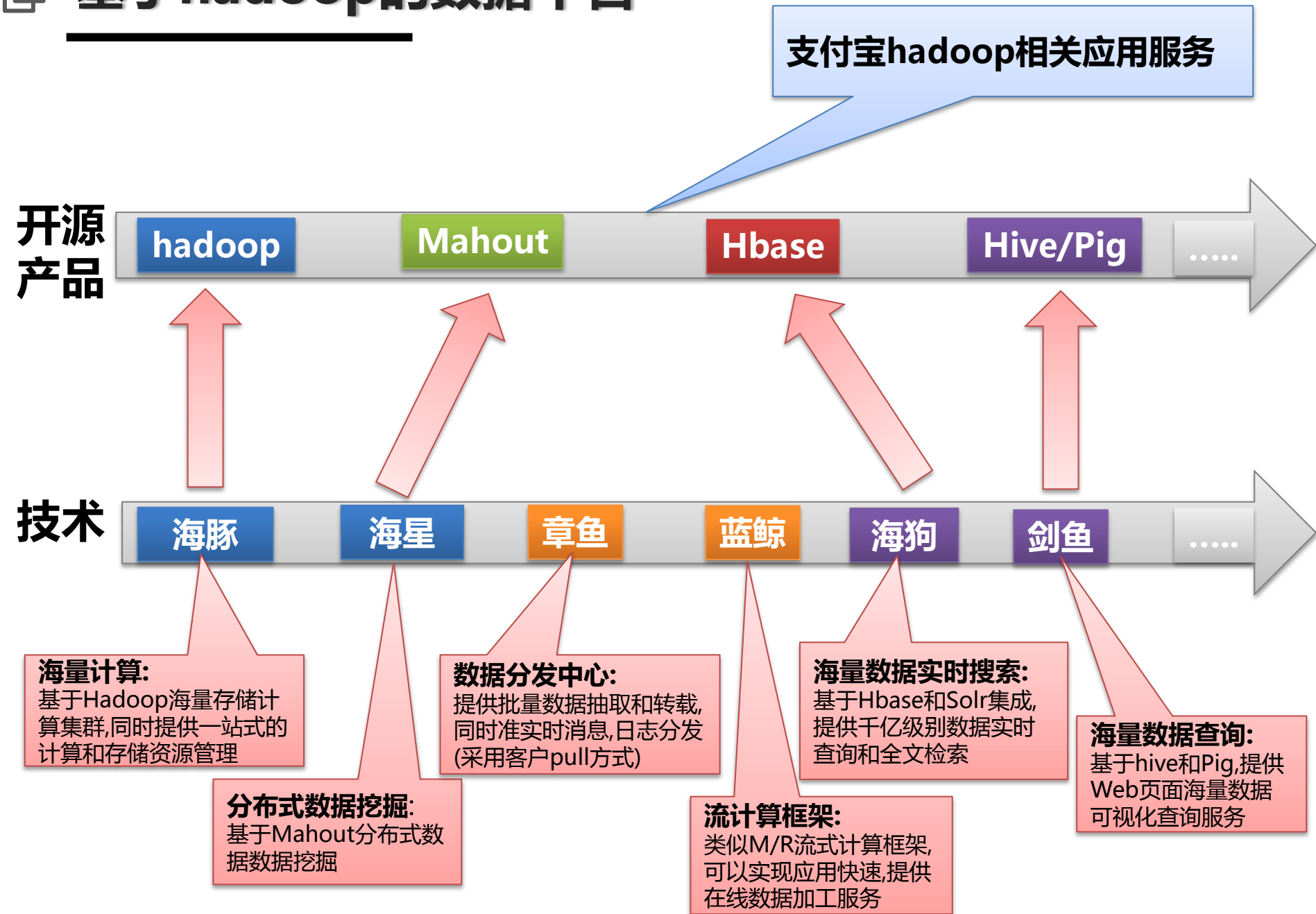

新客户
潜在客户
成熟客户


流失挽留
维系提升
获取阶段

# 目录

第四篇:支付宝数据平台架构

# 基于hadoop的数据平台

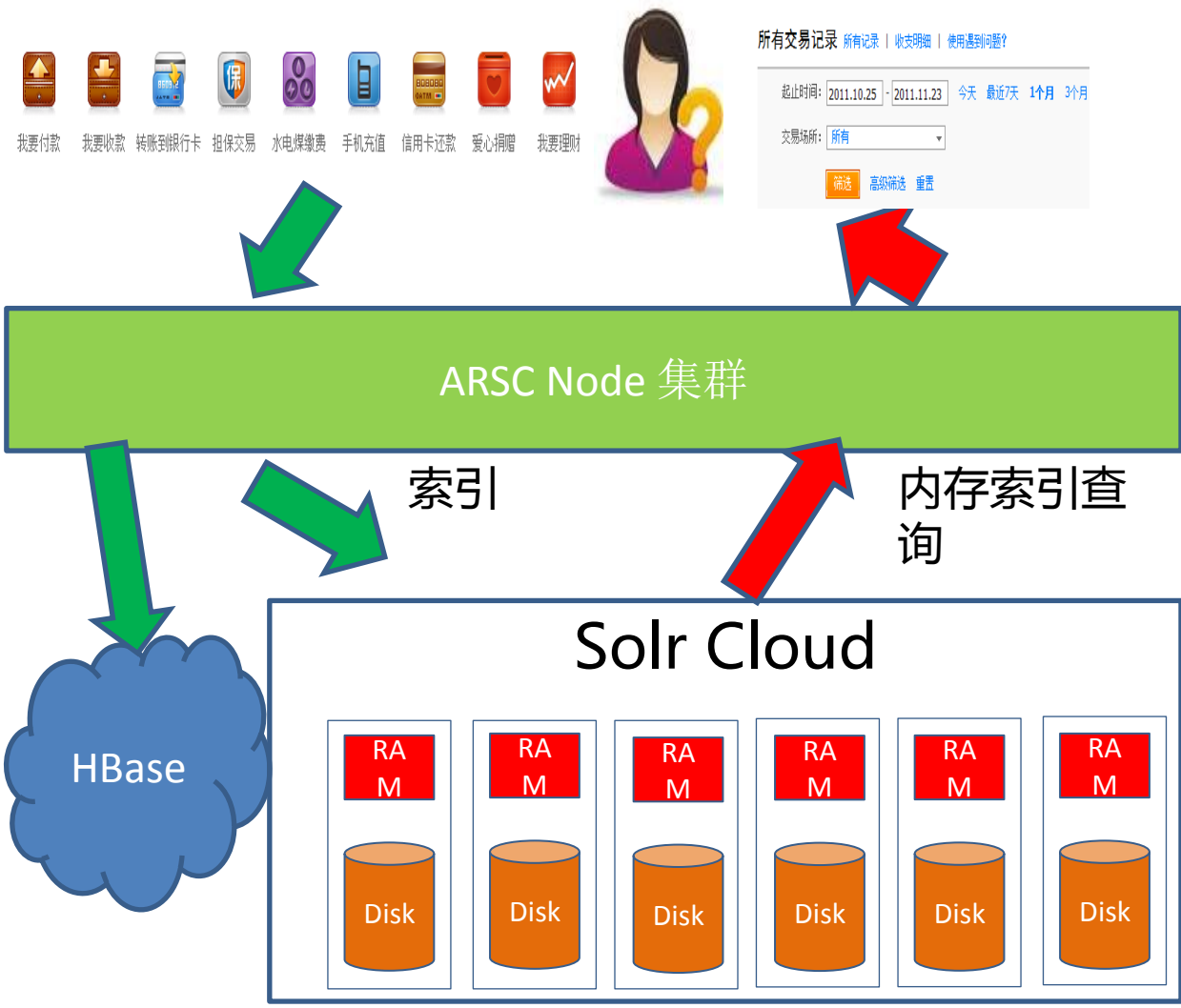


# 目录

第五篇:重点产品介绍---海狗(ARSC)

# 海狗系统(ARSC)—准实时搜索查询

- 项目价值
  - ✓ 提供千亿级别数据**实时查询**和**全文检索**
  - ✓ 支持每天10亿+级别的数据更新
- 实时
  - ✓ 实时搜索延迟：3s
  - ✓ 查询和插入TPS：>1.5WTPS
- 数据容量
  - ✓ 线性扩展
- Schema扩展
  - ✓ Schema Free (基于Hbase列式扩张)
- 自动容灾
  - ✓ 基于ZK动态感知节点状态



# What is ARSC ?

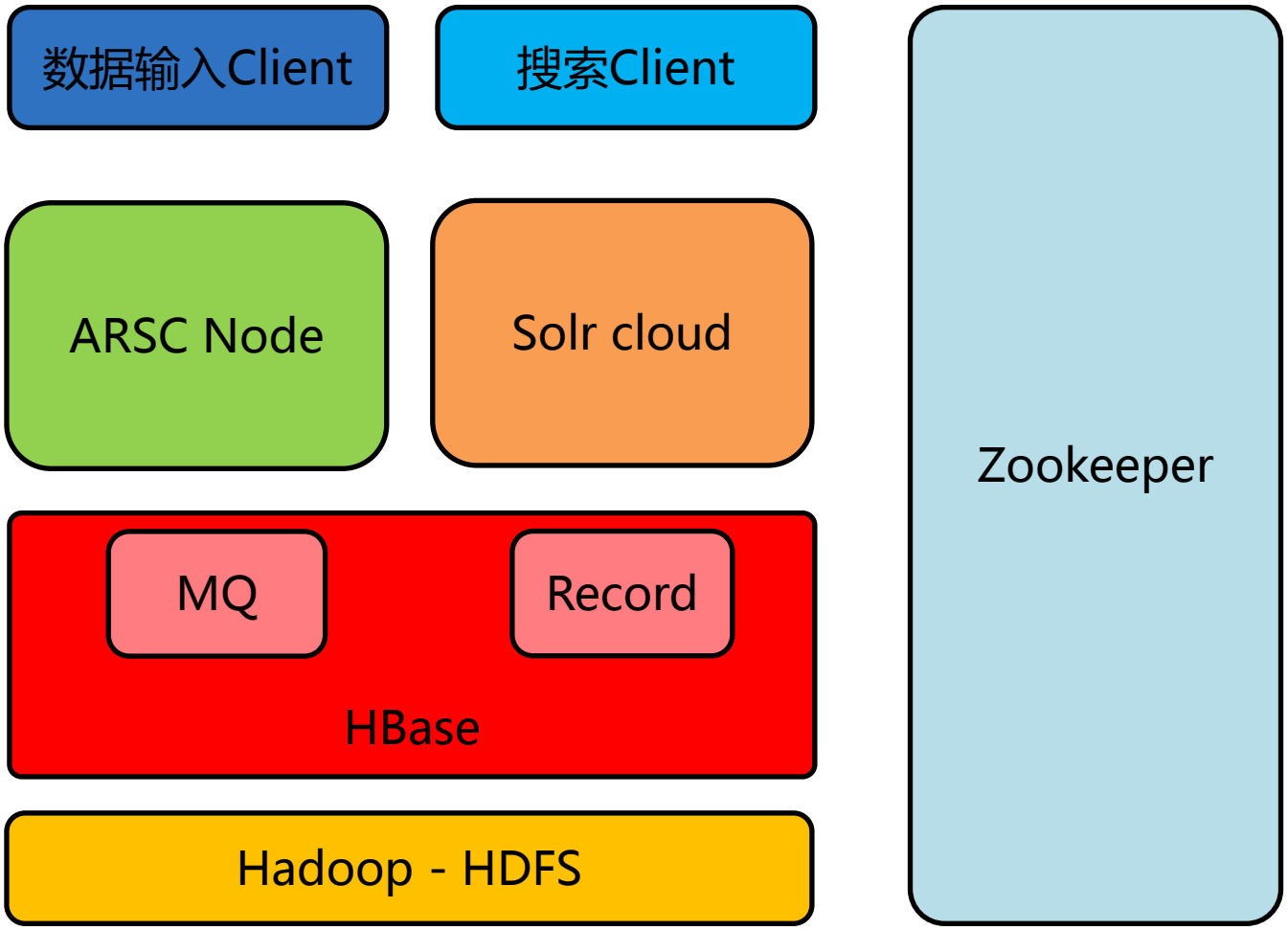
- 海狗(ARSC )
  - ✓ 支付宝实时搜索集群平台
  - ✓ Alipay Real-time Search Cluster (音同Ask)

- 海狗相关开源产品
  - ✓ Hadoop
  - ✓ HBase
  - ✓ Zookeeper
  - ✓ Solr
  - ✓ Zoie

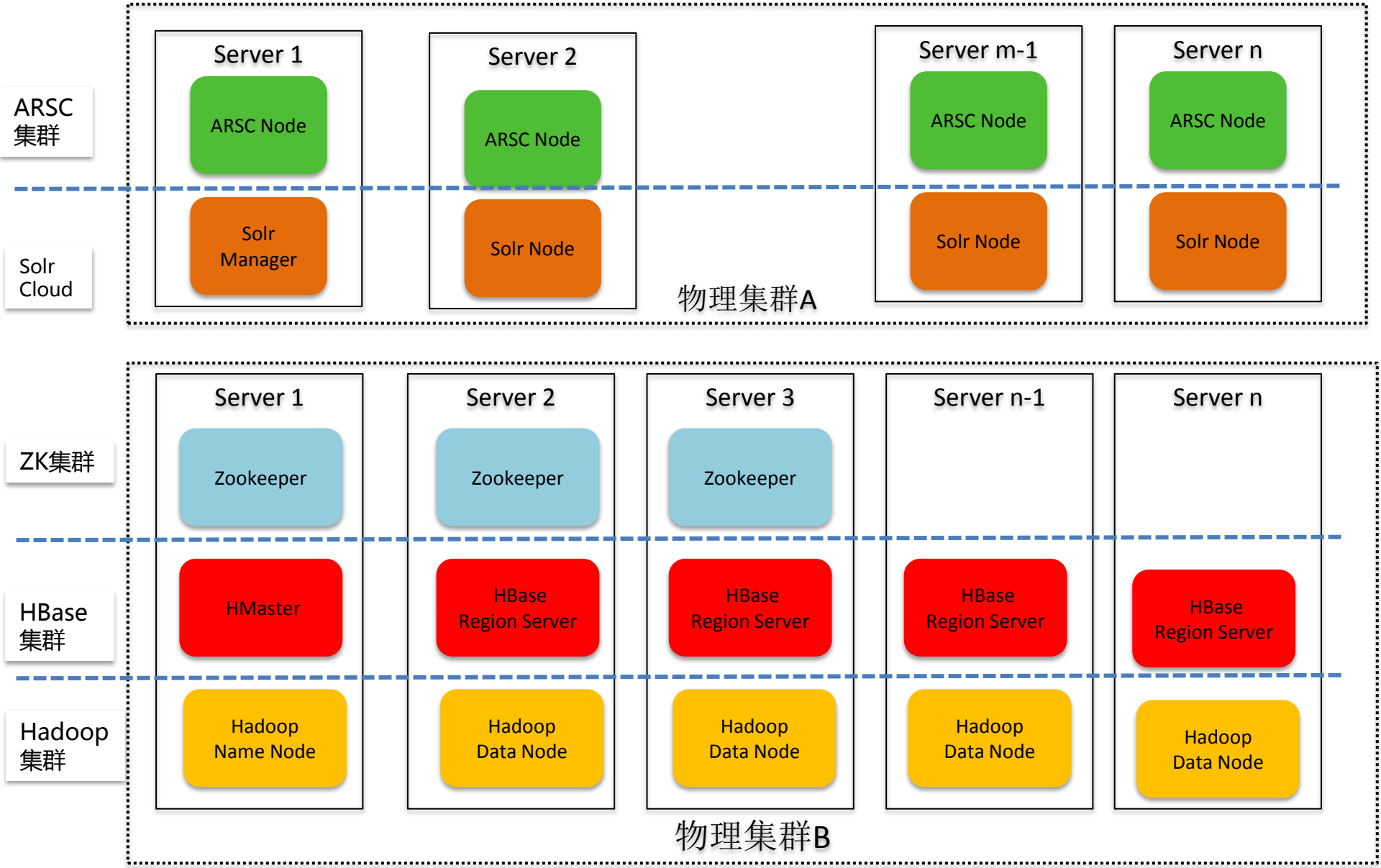
# 海狗系统项目价值

- 数据库无法支持海量数据的检索/全文检索
- 数据库存在Schema动态扩展问题
- HBase无法支持多维度检索
- 普通搜索引擎无法做到实时更新数据索引

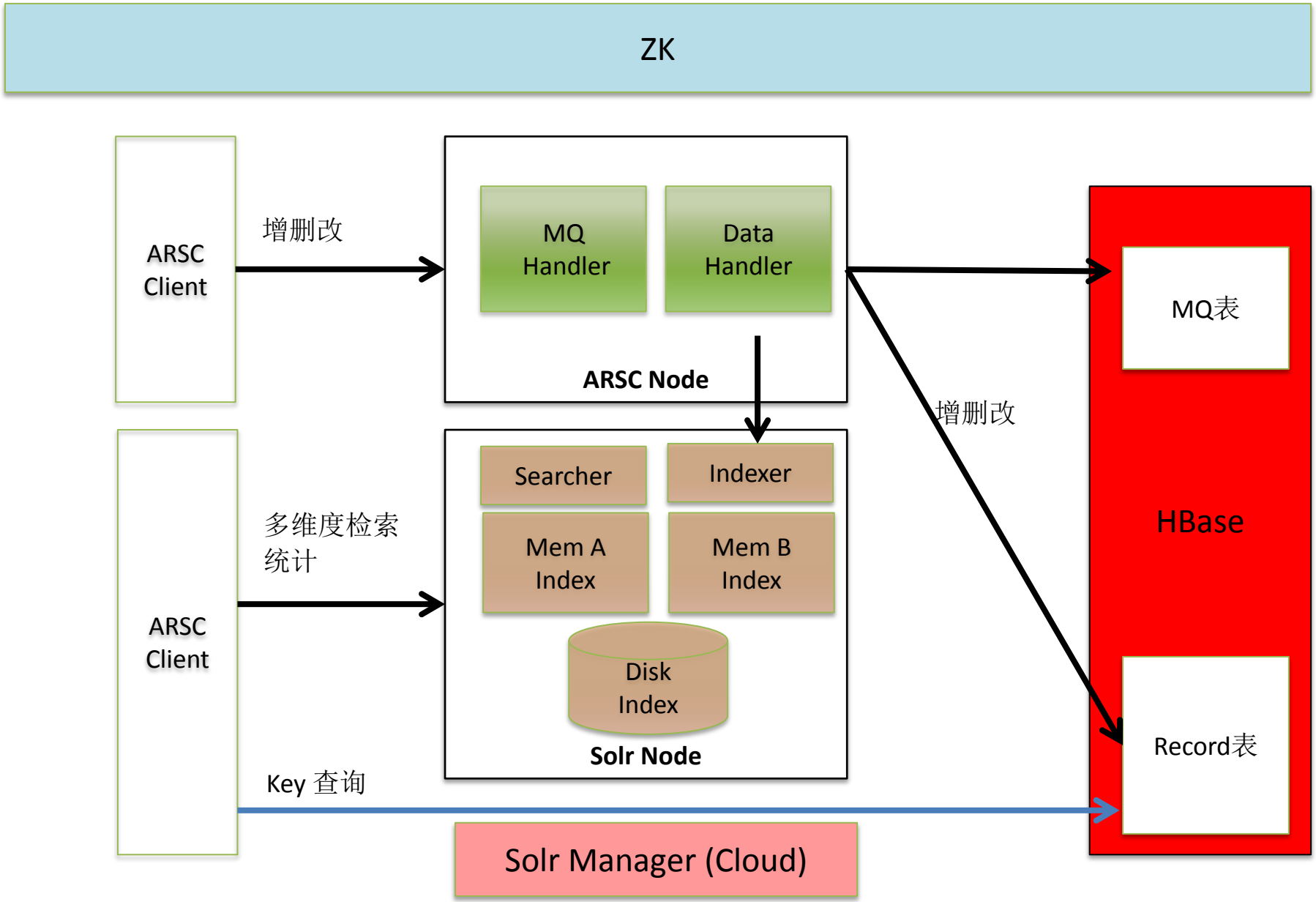
# 海狗系统逻辑架构


数据输入Client
搜索Client
ARSC Node
Solr cloud
MQ
Record
HBase
Hadoop - HDFS
Zookeeper

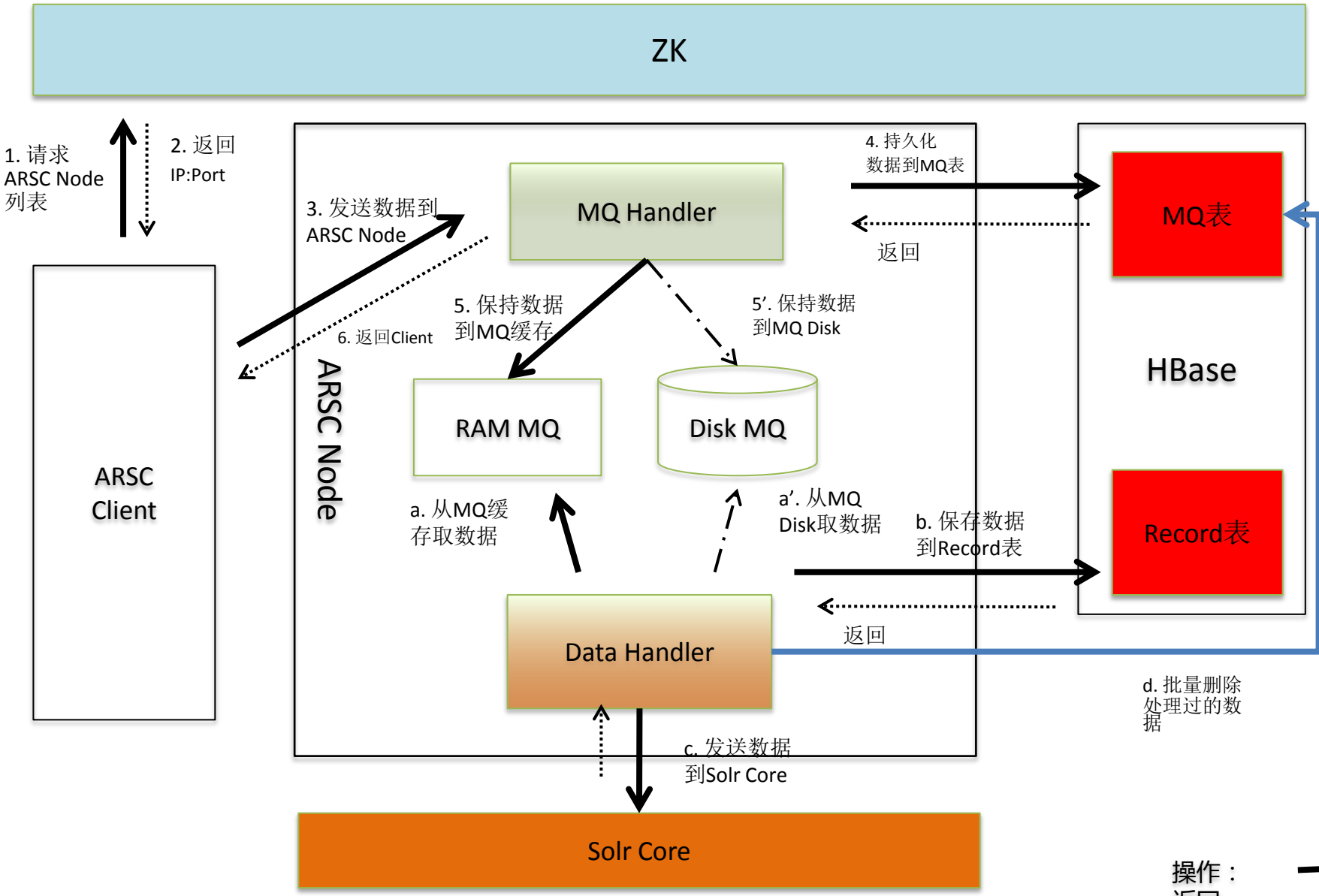
# 海狗集群架构


ARSC 集群
Solr Cloud
Server 1
ARSC Node
Solr Manager
Server 2
ARSC Node
Solr Node
Server m-1
ARSC Node
Solr Node
Server n
ARSC Node
Solr Node
物理集群A
ZK集群
HBase 集群
Hadoop 集群
Server 1
Zookeeper
HMaster
Hadoop Name Node
Server 2
Zookeeper
HBase Region Server
Hadoop Data Node
Server 3
Zookeeper
HBase Region Server
Hadoop Data Node
Server n-1
HBase Region Server
Hadoop Data Node
Server n
HBase Region Server
Hadoop Data Node
物理集群B

# 海狗功能模块


ZK
ARSC Client
增删改
MQ Handler
Data Handler
ARSC Node
MQ表
增删改
HBase
Searcher
Indexer
Mem A Index
Mem B Index
Disk Index
Solr Node
ARSC Client
多维度检索
统计
Record表
Key 查询
Solr Manager (Cloud)

# ARSC Node介绍


ZK
1. 请求
ARSC Node
列表
2. 返回
IP:Port
ARSC
Client
3. 发送数据到
ARSC Node
6. 返回Client
ARSC Node
MQ Handler
4. 持久化
数据到MQ表
返回
MQ表
HBase
5. 保持数据
到MQ缓存
5'. 保持数据
到MQ Disk
RAM MQ
Disk MQ
a. 从MQ缓
存取数据
a'. 从MQ
Disk取数据
b. 保存数据
到Record表
Record表
Data Handler
返回
d. 批量删除
处理过的数
据
c. 发送数据
到Solr Core
Solr Core
操作：
返回：
可选步骤：

# ARSC Node介绍

## ARSC Node主要作用

- ✓ 高效的接收Client输入数据
- ✓ 同步HBase Record数据到Solr索引(WAL作用)
- ✓ 缓存瞬时高并发数据

## ARSC Node处理流程

1. Client 请求 ZK
2. ZK取得 ARSC Node列表返回给Client
3. ARSC Node接收Client CRUD请求
4. ARSC Node通过MQ Handler模块持久化数据到MQ-Shard表
5. 通过MQ Handler模块写MQ内存缓存
   - 若内存缓存写满，那么开始写本地硬盘上
6. 返回客户端

a) Data Handler从内存MQ取数据
   - 若内存MQ为Empty，那么从本地硬盘读取MQ数据

b) 保持数据到Record表，并返回响应结果
c) 发送数据到Solr Core，并返回响应结果
d) 批量删除处理过的数据

# Solr Cloud介绍--- Solr Node(基于Solr 二次开发)

- Solr Node主要作用
  - ✓ 接收ARSC Node发送数据
  - ✓ 创建实时索引
  - ✓ 提供实时搜索

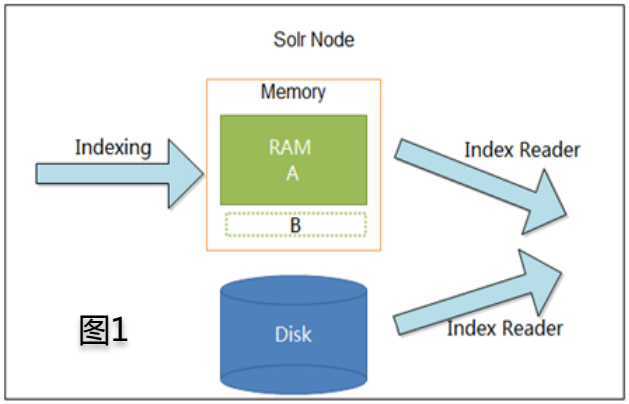
- 实时索引和搜索 (参见下图)

1. Solr Core接收从MQ Push过来的数据，保存到内存索引A (B为空) [图1]
2. 内存索引A是每添加完文档后立刻更新索引，保证实时性 [图1]
3. 内存索引A和硬盘上的索引Disk，同时对外提供搜索服务 [图1]

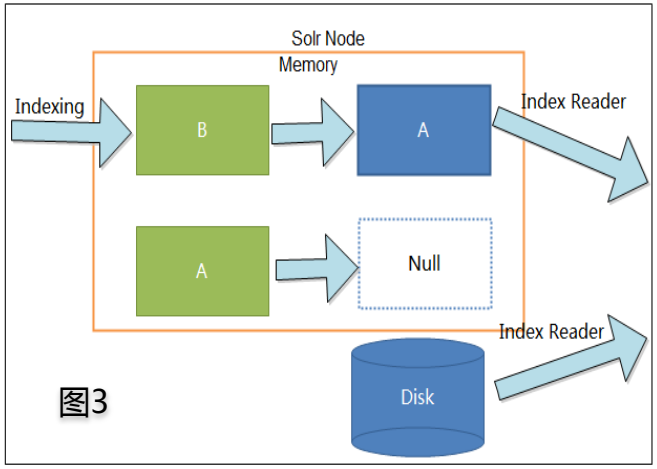
4. 当A中的文档数量达到一定的数量时，需要同硬盘上的索引进行合并，这时候会创建内存索引B，在合并过程中新添加的文档全部放入内存索引B中 [图2]
5. A，B和Disk Index共同对外提供搜索服务(PS: A中的索引不会重复索引，索引一致性保证) [图2]

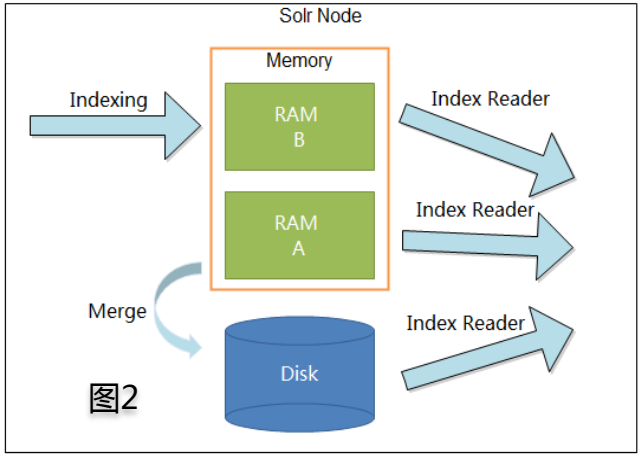
6. A和Disk index 合并之后，原来的索引A变为null， B改名为A [图3]
7. 重新打开Disk索引提供搜索（Disk Index= A + Old Disk Index）[图3]

# Solr Cloud介绍---Solr Node实时索引与搜索




A：初始状态
B：内存索引A满后状态
　　合并内存索引A和硬盘索引
C：磁盘索引和A内存索引合并结束之后状态

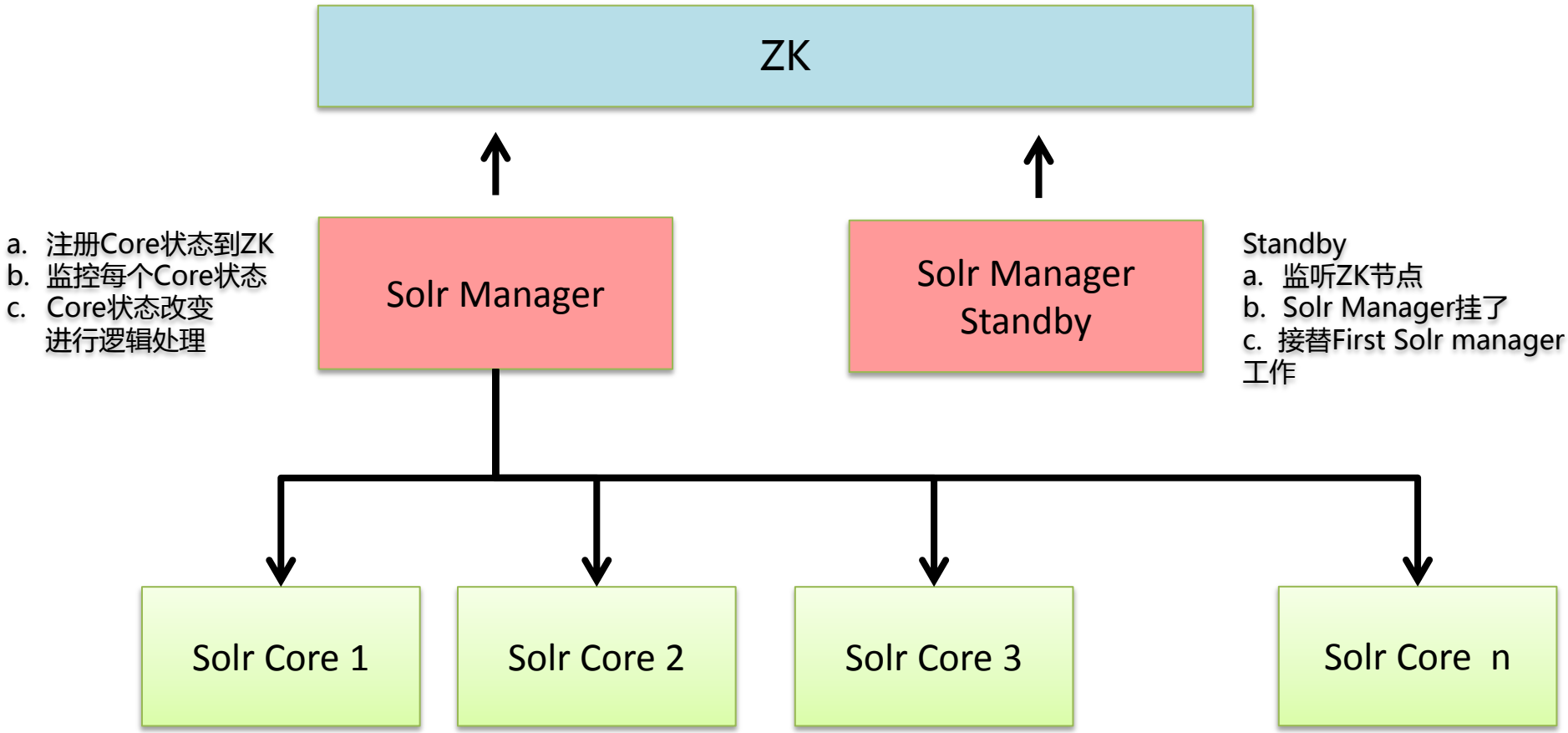


# Solr Cloud介绍---Solr Manager

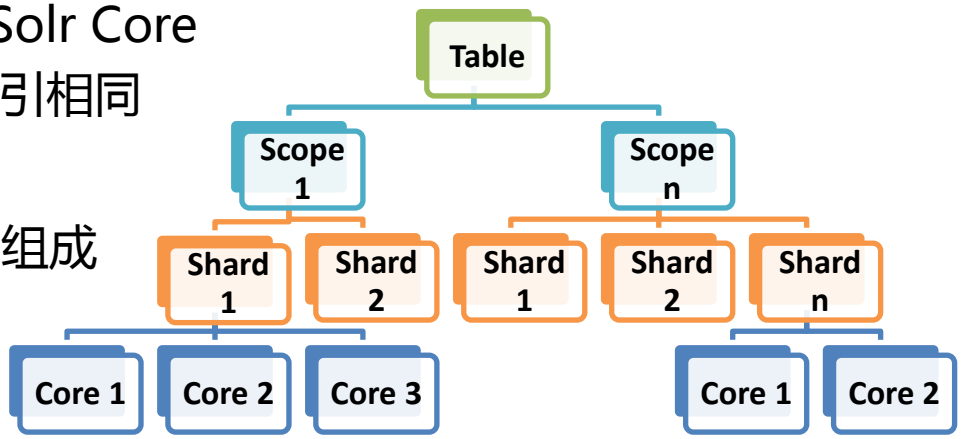
- Solr Manager作用 (Solr Core容灾)
  - ✓ 监控所有Solr Core的状态（每隔3s， 遍历所有Core的状态）
    - Online
    - Offline
    - Error
    - Read-only
  - ✓ Solr Core容灾
    - 当Core Down之后，分配一个空的Core给此节点，从其他节点同步索引，完成容灾
    - 当一个Shard下所有Core Down之后，调用Map/reduce程序从
    - 动态增加一个Core，索引分配

# Solr Cloud介绍---Solr Manager



# ARSC 扩展和容灾 ---概念

- 逻辑数据结构
  - ✓ Table
  - ✓ Scope
    - 一张表由多个Scope组成(通过算法来划分)
  - ✓ Shard
    - 一个Scope由多个Shard组成
    - 每一个Shard是一部分索引
  - ✓ Solr Core
    - 每个Shard对应系统若干个Solr Core
    - 同一Shard下所有Core的索引相同
  - ✓ Solr Node
    - 一个Solr Node由多个Core组成



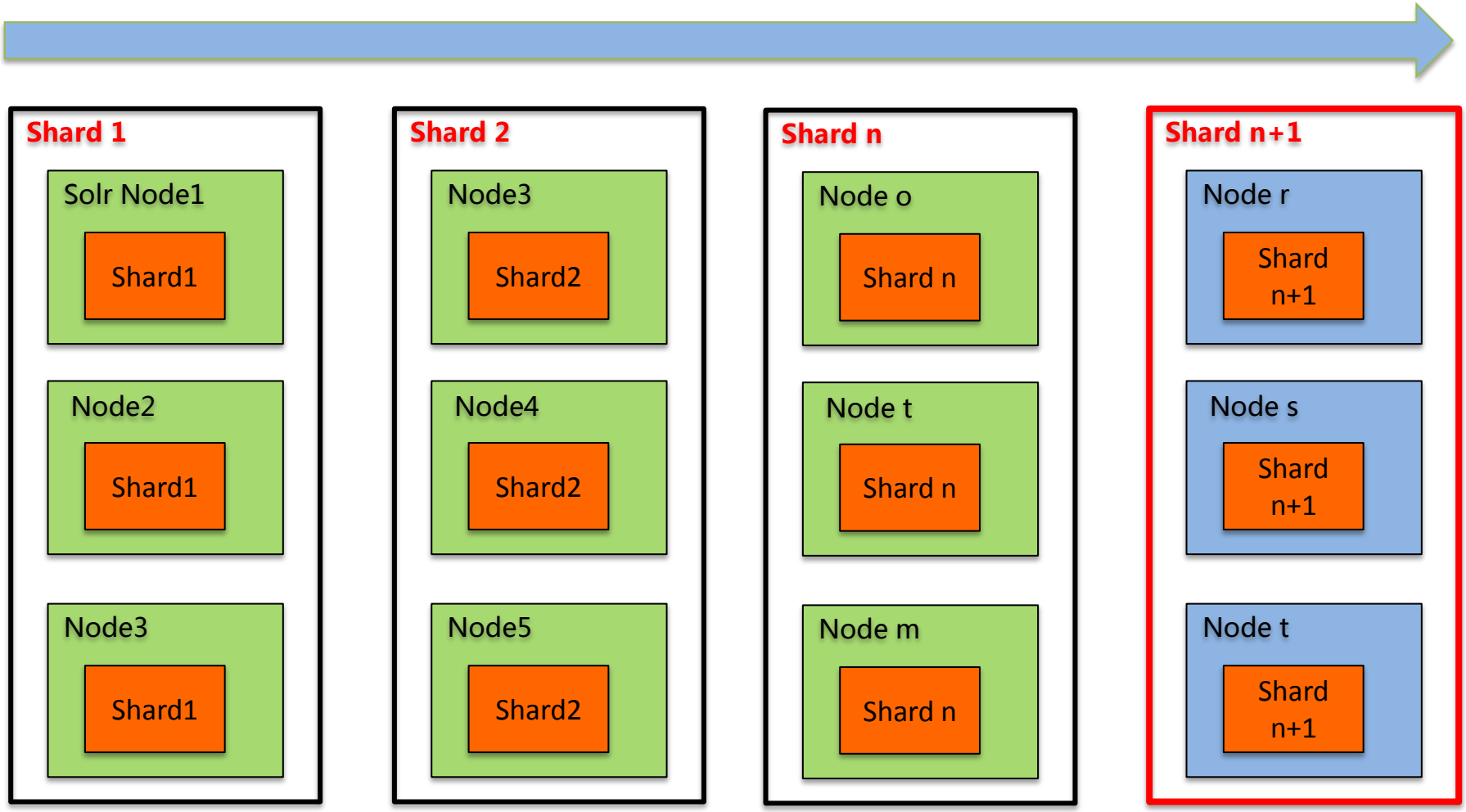
# 海狗 扩展和容灾

- 动态扩展
  - ✓ 容量扩展
    - Hadoop/HBase动态增加机器
    - Solr Cloud增加Shard数量
  - ✓ 性能扩展
    - HBase：性能扩展通过增加机器
    - Solr Cloud：增加同一Shard下Core的数量，分担负载
    - ARSC Node：动态增加机器，分担负载过重

- 容灾
  - ✓ 存储容灾
    - HBase：当RS Down，HBase可以自动容灾
    - Hadoop：文件保存3个副本
    - Solr Node： 同一份索引缺省保存3份
  - ✓ ARSC Node：
    - 当一台ARSC Node Down，ZK感知到会分配任务到其他ARSC Node
  - ✓ Solr Core容灾：
    - Solr Manager每隔固定时间间隔会扫描Solr Core的状态，若发现Solr Core Down，Solr Manager启动恢复进程；并且阻止同一Shard下的数据接收，直到恢复完成。
  - ✓ Solr Manager容灾
    - 同时有2台Solr Manager存在，一个是Master，另一个是Standby

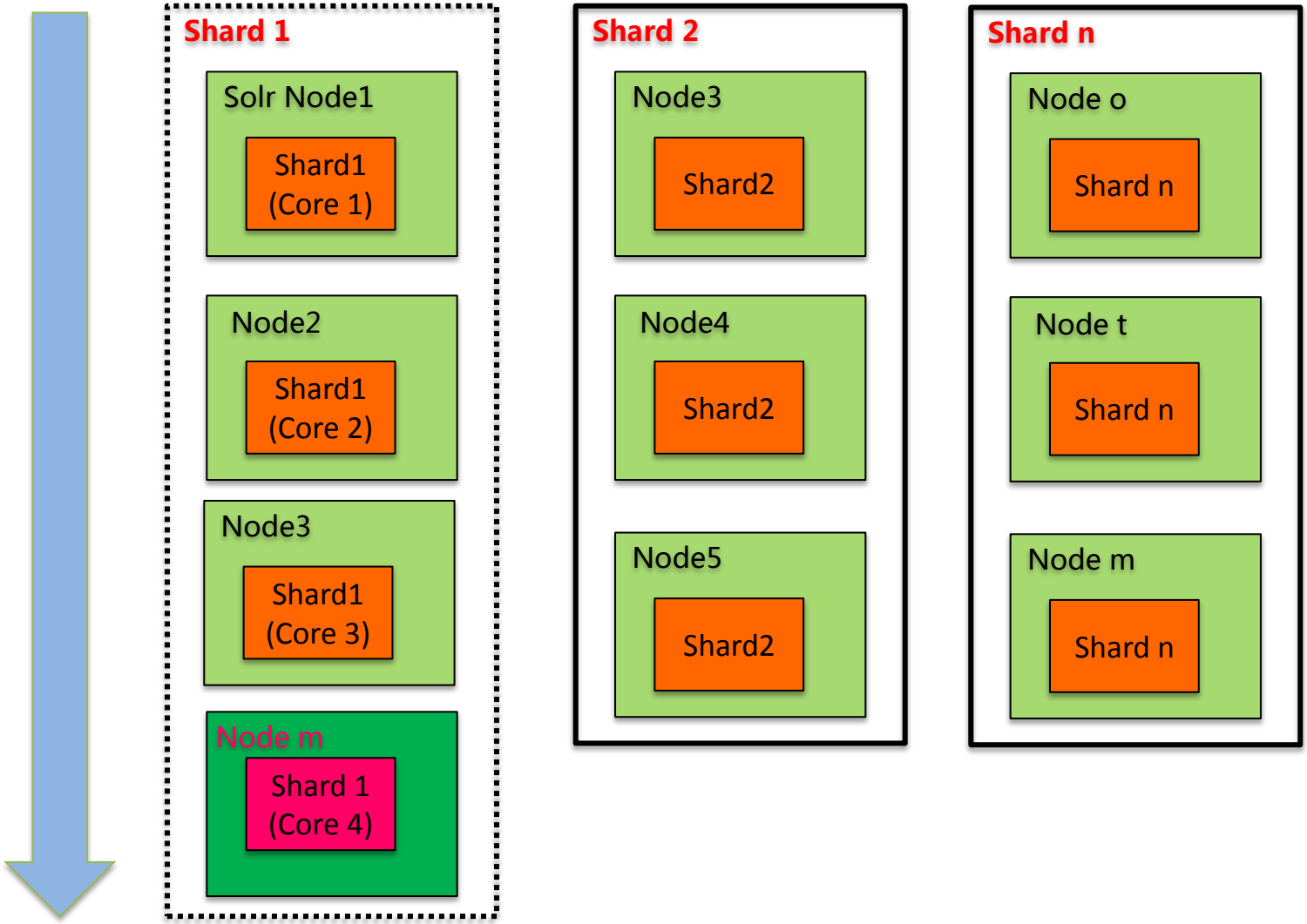
# 海狗性能扩展---Solr 容量扩展

- Solr Cloud容量扩展
  - ✓ Solr Cloud增加Shard数量（增加Shard n+1 )来达到增加容量目的



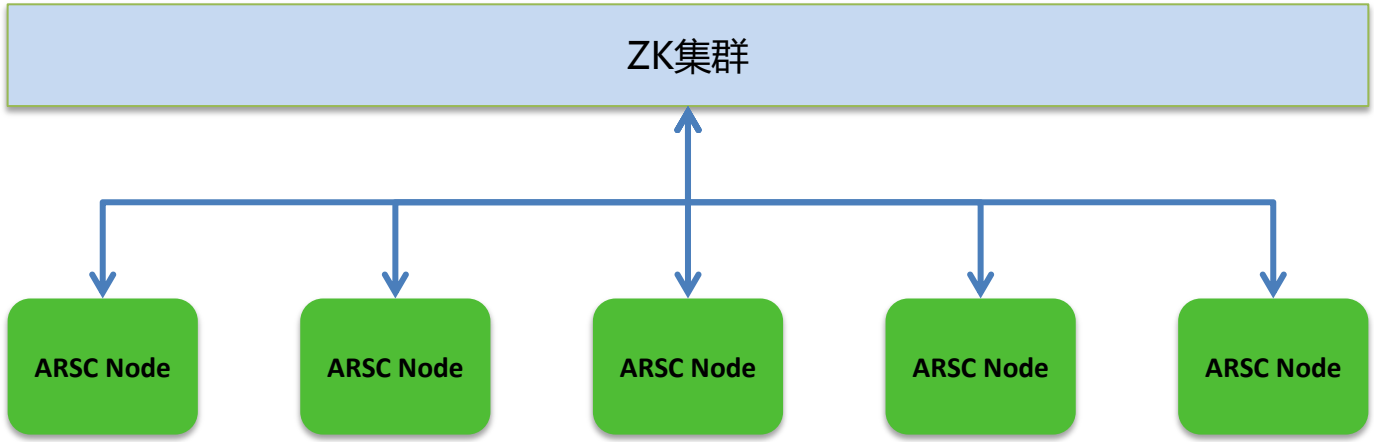
# 海狗性能扩展---Solr 性能扩展

- Solr性能扩展 (增加同一Shard下Core的数量，分担负载)
  - ✓ 原来Shard 1下面有3个Core (分布在 Node 1, Node 2, Node 3 ) 增加一个Core 4 (在Node m上)
  - ✓ Core 1~4 数据完全相同，4个Core 可以平均分担查询负载



# ARSC Node容灾

- ARSC Node容灾
  - ✓ 当一台ARSC Node Down，ZK感知到，选择其中一个正常的ARSC Node接管工作




- Solr Core & Solr Node容灾
  - ✓ 由Solr Manager完成

# 海狗 优势

- 实时
  - ✓ 实时数据更新和检索
  - ✓ 实时多维度检索，支持数值检索，枚举检索，全文检索
  - ✓ 搜索结果统计
    - Max，Min，Avg，Sum, Count
    - Group By，Order By
    - 自定义统计函数扩展
  - ✓ 异步的批量查询
  - ✓ 类SQL查询语句
- 自动容灾
  - ✓ Hadoop/HBase自动容灾
  - ✓ ARSC Node自动容灾
  - ✓ Solr Manager针对Solr Core自动容灾
- 扩展灵活
  - ✓ 性能动态扩展
  - ✓ 容量线性扩展
  - ✓ 动态负载均衡
  - ✓ 动态的Schema扩展

# 海狗不足

- CAP理论:
  - ✓ 一致性(Consistency)：任何一个读操作总是能读取到之前完成的写操作结果，也就是在分布式环境中，多点的数据是一致的;
  - ✓ 可用性(Availability)：每一个操作总是能够在确定的时间内返回，也就是系统随时都是可用的。
  - ✓ 分区容忍性(Partition Tolerance)： 在出现网络分区(比如断网)的情况下，分离的系统也能正常运行。
- NOSQL通常只能满足其中的两个特点，ARSC满足的是
  - ✓ 可用性
  - ✓ 分区容忍性
- ARSC满足最终一致性
  - ✓ 数据插入Solr Cloud之后，同一份Shard数据的3个Solr索引节点之间存在数据不一致的窗口现象，最终3个Solr节点数据一致

# 海狗性能测试结果

- 测试环境(一共12台机器)
  - ✓ 6台物理机
    - 部署Hadoop
    - 部署HBase
    - 部署Zookeeper
  - ✓ 6台物理机
    - 部署6个ARSC Node
    - 部署6个Solr Node(每个机器5个Core一共30个Core)
    - 部署2个Solr Cloud (Zolr Manager)
  - ✓ 测试结果
    - **插入TPS： 15K/s+**
    - **平均实时更新时间：3s**
    - 插入平均响应时间：15ms
    - 每天吞吐量：10亿+
  - ✓ 测试结果总结
    - 系统性能可以线性扩展，增加机器可以增加系统TPS

# 努力,为了明天,数据创造价值

## Q/A

Email:jie.jiangj@alipay.com