ZTE中兴

NoSQL技术的电信业务实践

Bringing you closer

高洪 中兴通讯股份有限公司

2011年12月

目录

What is NoSQL?

- NoSQL（Non relational，或者Not Only SQL），泛指非关系型数据库，早期就有人提出，发展至2009年趋势越发高涨。

- NoSQL这个名词，业界并没有明确的定义。nosql-database.org对NoSQL有一个较为全面的解释，指出NoSQL的特点有：非关系、分布式、开源、水平扩展、schema-free、easy replication support、simple API、最终一致性（不支持ACID）、支持海量数据（Huge Data）

Why NoSQL?

在web2.0时代，各种应用层出不穷，数据量急剧增长，对数据库技术提出了新的需求：

- High performance —— 对数据库高并发读写的需求
- Huge Storage —— 对海量数据的高效率存储和访问的需求
- High Scalability && High Availability —— 对数据库的高可扩展性和高可用性的需求
- Low cost —— 对数据库低成本的需求

传统关系型数据库无法解决以上问题，具有如下缺陷：

- 扩展困难
- 并发读写慢
- 成本高
- 无法支撑海量数据（支持Big Data）

NoSQL的使用现状

NoSQL因互联网而生，故目前的主要应用大多在互联网企业中，自2009年以来，更是得到了空前的发展

公司	应用NoSQL	自研/应用	应用场景
Google	BigTable	自研	Web indexing，Google Earth，Google Finance
Facebook	Cassandra	自研	收件箱搜索
Amazon	Dynamo	自研	购物车
新浪	Redis	应用	微博
淘宝	OceanBase	自研	淘宝收藏夹，逐步在其他应用上线
	Tair	自研	登录/查看商品详情/淘江湖等
视觉中国	MongoDB	应用	视觉中国网站
优酷	MongoDB	应用	在线评论业务
	HBase	应用	运营数据分析及挖掘处理
	Tokyo Tyrant	应用	搜索产品
	Redis	应用	频繁修改的在线业务
中国移动	HandlerSocket	应用	飞信空间
豆瓣网	BeansDB	自研	豆瓣社区

优势和挑战

NoSQL优势：

- 大数据量，高性能，高可用
- 弹性扩展
- 灵活的数据模型
- 低成本
- 管理简单

NoSQL面临的挑战：

- 成熟度 ，目前大部分NoSQL产品都还不够成熟，还有大量的关键特性有待实现。
- 支持力度，大部分的NoSQL系统都是开源项目，没有能力提供全球的支持，没有足够的支持资源，或者没有类似于Oracle、Microsoft或者IBM的信用。
- 分析与商业智能
- 管理，NoSQL的设计目标是提供零管理的解决方案，但现实是，此目标远远没有实现。目前的NoSQL系统需要大量的技能来进行安装，以及需要大量的努力来进行维护。

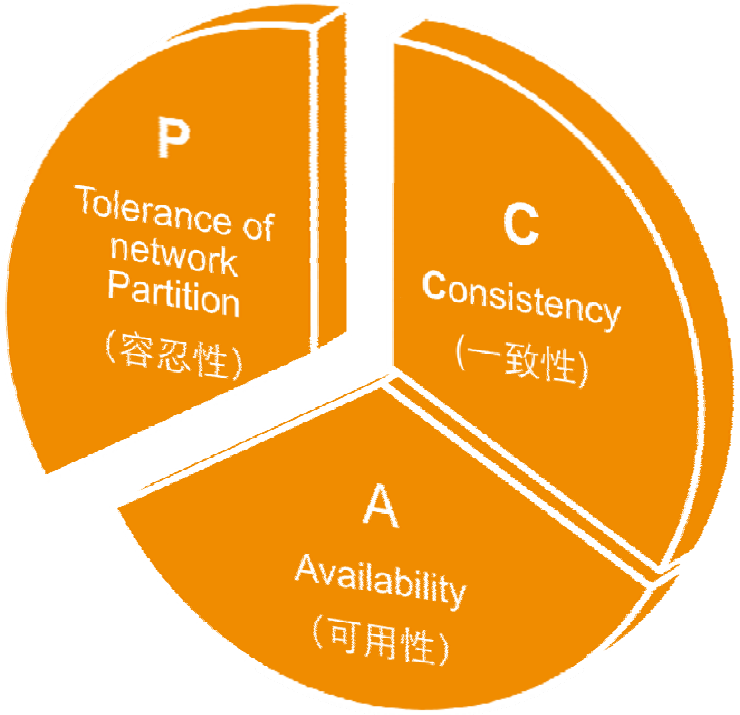
NoSQL vs. RDBMS

比较项	RDBMS	NoSQL	说明
一致性	强一致性	最终一致性	RDBMS遵循ACID，要求强一致性；NoSQL对一致性要求相对较弱，采用最终一致性
数据级	大(Big)	超大(Huge)	RDBMS随着存储量的增大性能骤降，所以一般无法支持海量数据；NoSQL理论上可以通过简单的增加设备来扩大存储量。
扩展性	一般	容易	RDBMS由于join，扩展困难；NoSQL通过增加节点线性扩展
可用性	好	很好	RDBMS当数据量大时由于其保证强一致性可用性差；NoSQL放宽了一致性要求，可以达到更好的可用性
成熟度	很高	低	RDBMS发展了40年，技术成熟；NoSQL刚刚起步，无业界标准
应用复杂度	低	高	RDBMS支持SQL工业标准；NoSQL无标准，相关细节功能需要软件开发者具体实现
Schema	固定	灵活	RDBMS的表要定义好，维护起来复杂；NoSQL是schema free的，可随时修改schema结构
管理	复杂	简单	RDBMS要经常进行调优，要高度受训的DBA；现阶段NoSQL不成熟，管理也很复杂
支持力度	高	低	RDBMS使用者和专家很多，问题解决容易；NoSQL刚刚起步
成本	高	低	RDBMS运行在昂贵的高性能服务器上；NoSQL运行在廉价的设备上

目录

CAP理论

- 任何分布式系统只可同时满足二点，没法三者兼顾。
- 架构师不要将精力浪费在如何设计能满足三者的完美分布式系统，而是应该进行取舍。
- 一般来说，分布式存储系统可以通过配置在CAP间权衡，满足不同应用的需要。于是在电信业务系统中具有广泛的适用性。



ACID&BASE思想

- ACID

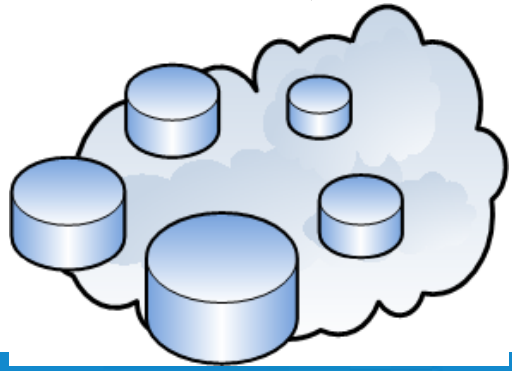
ACID模型侧重高一致性+数据可靠性，牺牲可用性。

- BASE
- **B**asically **A**vailable（基本可用）：支持分区失败(e.g. sharding碎片划分数据库)
- **S**oft state（软状态）：状态可以有一段时间不同步，异步。
- **E**ventually consistent（最终一致）：最终数据是一致的就可以了，而不是时时高一致。

BASE思想反ACID，完全不同与ACID模型，牺牲高一致性，获得可用性和可靠性

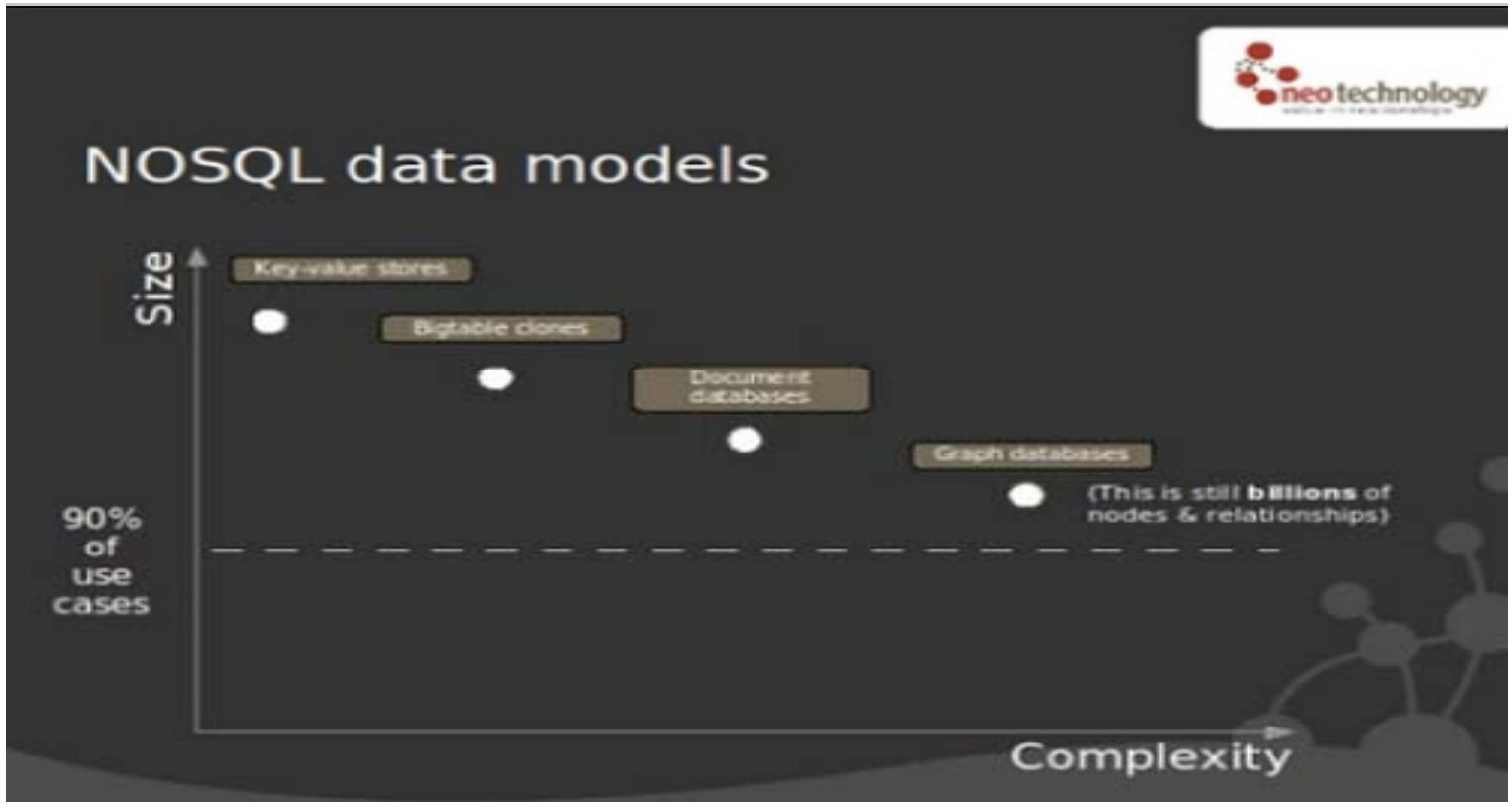
最终一致性

- **强一致性**：系统中的某个数据被成功更新(事务成功返回)后，后续任何对该数据的读取操作都得到更新后的值。这是传统关系数据库提供的一致性模型，也是关系数据库深受人们喜爱的原因之一。
- **弱一致性**：系统中的某个数据被更新后，后续对该数据的读取操作得到的不一定是更新后的值，这种情况下通常有个“不一致性时间窗口”(inconsistency window)存在：即数据更新完成后在经过这个“不一致性时间窗口”，后续读取操作就能够得到更新后的值。
- **最终一致性**：属于弱一致性的一种，即某个数据被更新后，如果该数据后续没有被再次更新，那么最终所有的读取操作都会返回更新后的值。



数据模型及操作类型

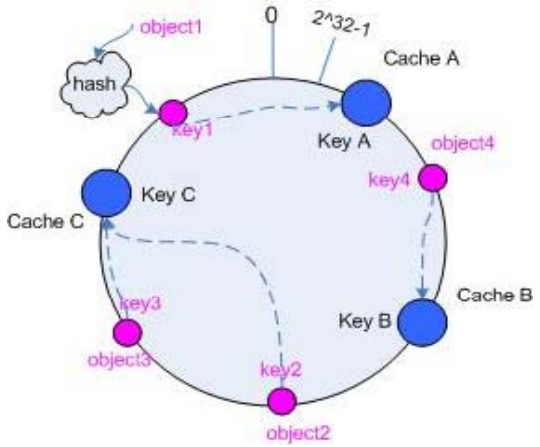
NoSQL在存储数据模型上，放弃了关系型模型，遵循schema-free原则，目前存在的NoSQL数据模型大致分为四类：Key-Value（键值对）、Column-Oriented（列式）、Document-Oriented（文档型）、Graph-Oriented（图型）。



分区

分区的目的，由于单机存储容量的限制以及单机负载过重，将数据分布在各个不同的节点上，使数据跨节点分布，同时，负载能力得到提升,两种扩展类型：

- Scale up
- Scale out



Scale out :

- 一致性HASH

在类Dynamo的系统中，如Riak、Voldemort、Cassandra均用到了一致性hash算法。

- 连续范围划分

BigTable、HBase、HyperTable均采用的这种分区策略

一致性

在NoSQL中，通常有两个层次的一致性：第一种是强一致性，即集群中的所有机器状态同步保持一致。第二种是最终一致性，即可以允许短暂的数据不一致，但数据最终会保持一致。

- 强一致性

Quorum NWR

- 最终一致性

Vector Clock，解决不一致窗口期间的版本冲突

可靠性

由于NoSQL为了降低成本采用的都是商用PC而非高性能服务器，硬件故障在所难免。所以系统必须提供高可靠性来保证节点宕机或硬盘故障时，系统仍是可用的

- 单机可靠性

  它的定义是写操作不会由于机器重启或者断电而丢失。通常单机可靠性的保证是通过把数据写到磁盘来完成的，而这通常会造成磁盘IO成为整个系统的瓶颈

- 多机可靠性

  主从和多副本策略

  跨数据中心的异步多机备份

扩展性和集群成员管理

NoSQL需要支持良好的扩展性，当系统容量不足或负载过重时，可随时添加节点而不影响现有业务，系统自动完成数据均衡。而集群中机器之间需要通讯，以了解彼此的状态信息。通常有两种架构

- Master/Slave

类BigTable的系统均采用Master/Slave架构

Master负责整个集群的命名空间管理、集群状态监控、负载均衡等

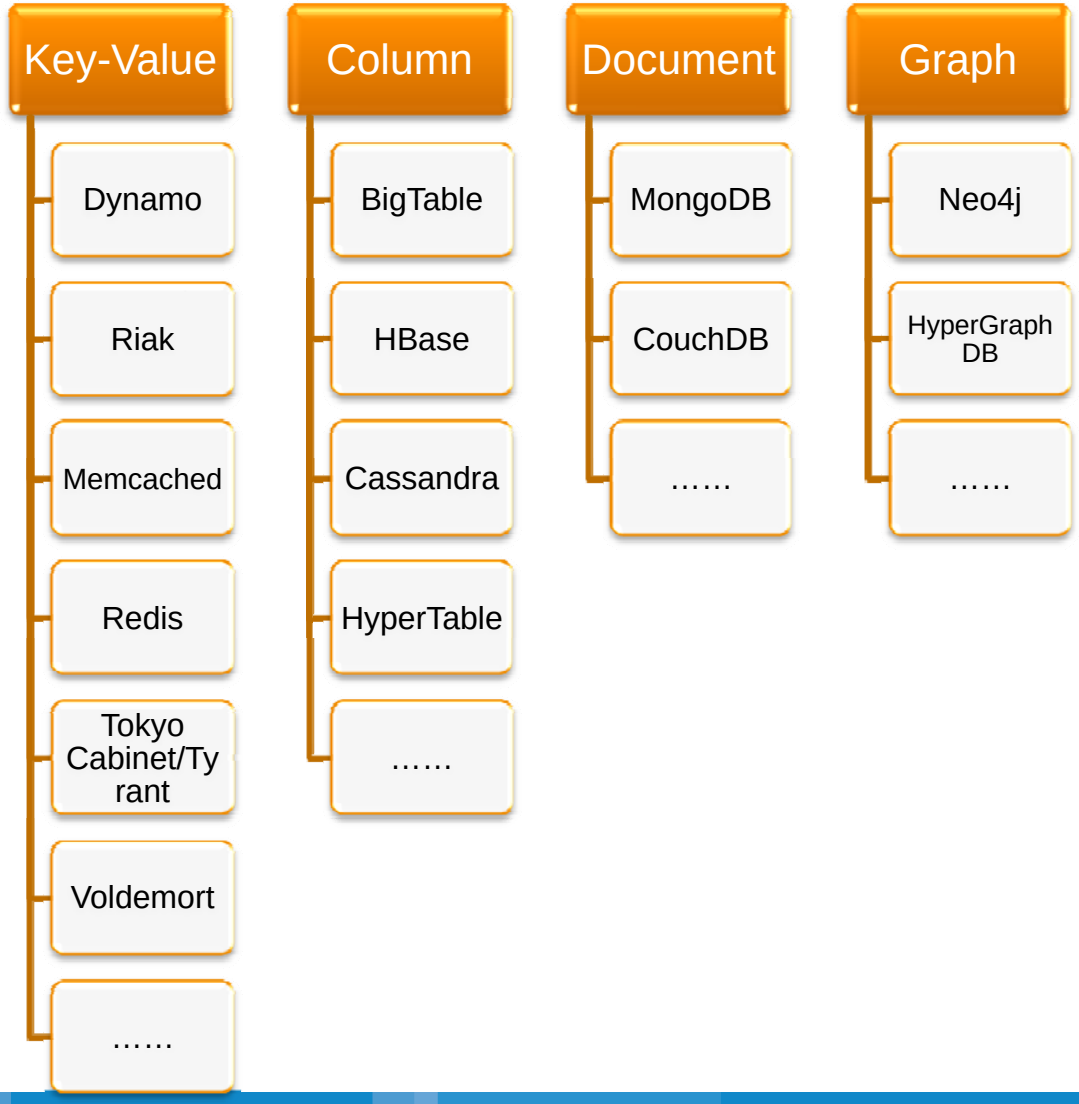
Slave来负责存储实际的数据，并定时的向Master报告自己的状态信息

- **P2P**

类Dynamo的系统均采用P2P架构，与Master/Slave不同，它是完全去中心化，不存在Master，各个节点是对等的关系，**gossip**

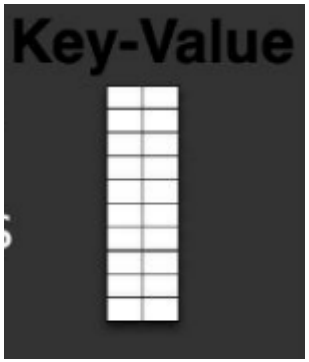
目录

NoSQL四大家族



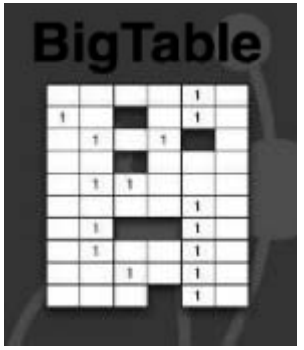
Key-Value

- 数据模型：简单的键值对
- 典型应用：
  - Cache
  - 高读写的简单存储
- 优势：
  - 简单、读写速度快
- 劣势：
  - 存储数据缺少结构化，只能存储Key-Value对
  - 复杂的结构需要上层应用处理



Column-Oriented

- 数据模型：BigTable式表格，稀疏矩阵
- 典型应用：
  - 海量数据存储与分析
- 优势：
  - 查找列速度快，可扩展性强，更容易进行分布式扩展
- 劣势：
  - 功能相对局限，对于跨表的join性能不高



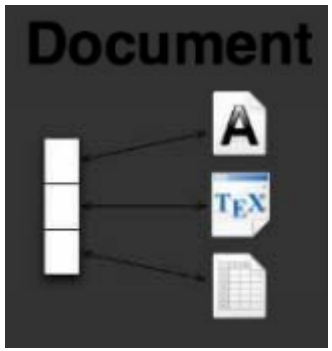
Google 谷歌

Cassandra

HBASE

Document-Oriented

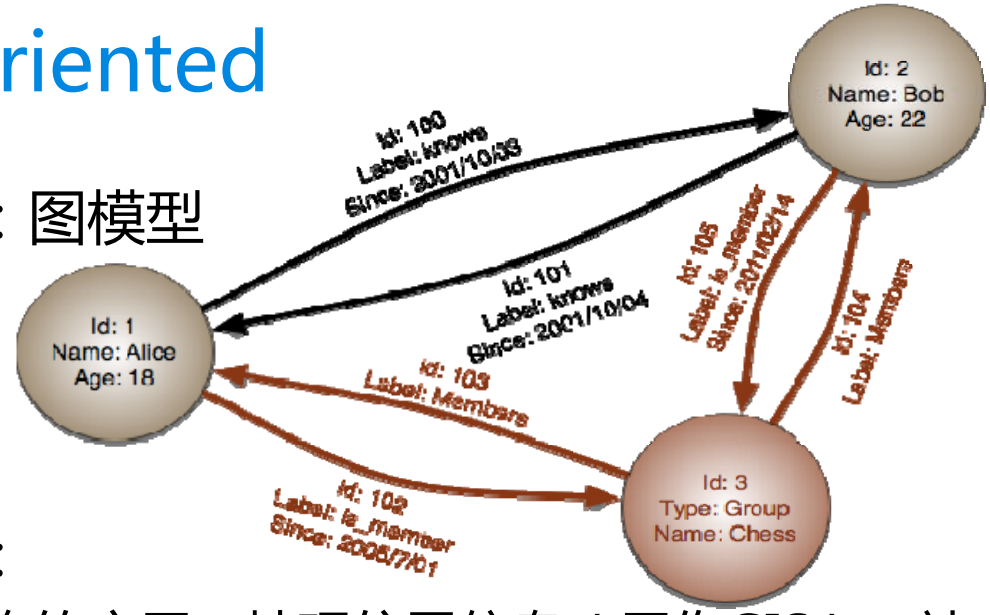
- 数据模型：灵感来源于Lotus Notes，JSON-like

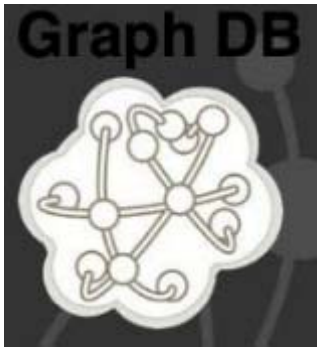



```
"Subject": "I like Plankton"
"Author": "Rusty"
"PostedDate": "5/23/2006"
"Tags": ["plankton", "baseball", "decisions"]
"Body": "I decided today that I don't like baseball. I like plankton."
```

- 典型应用：
  - Web应用
- 优势：
  - 数据结构要求不严格，灵活
- 劣势：
  - 查询性能不高，而且缺乏统一的查询语法

Graph-Oriented





- 数据模型：图模型
- 典型应用：
  - 基于网络的应用，地理位置信息（用作GIS）、社交网络（SNS）、网络拓扑等
- 优势：
  - 可以利用图结构相关算法
- 劣势：
  - 需要对整个图做计算才能得出结果，不容易做分布式的集群方案

目录

- NoSQL的前世今生
- NoSQL的理论和关键技术
- NoSQL典型产品与分析
- NoSQL之分布式存储产品
- 电信业务应用实践

中兴通讯云计算总体架构

分布式存储产品

- 致力于为业务产品和互联网产品提供一个分布式、高性能、高可靠，低时延，具备电信级运营要求的NoSQL存储产品

高性能，性能在同类开源产品中领先

可扩展，可以在线扩展，对应用透明

大容量，集群规模可以做到海量

高可靠，单点故障对集群无影响，并支持临时故障和永久故障后的数据冗余恢复

无中心化，避免中心节点不可用而引起系统的不可用，且效率更高

可伸缩，冗余策略可针对不同的应用特点配置

高可用性，单点故障完全不影响应用访问

多应用空间，同一集群中为不同应用提供适合的存储策略

多级存储，支持内存，SSD，硬盘三级存储，满足不同层次需求

支持大数据的存储

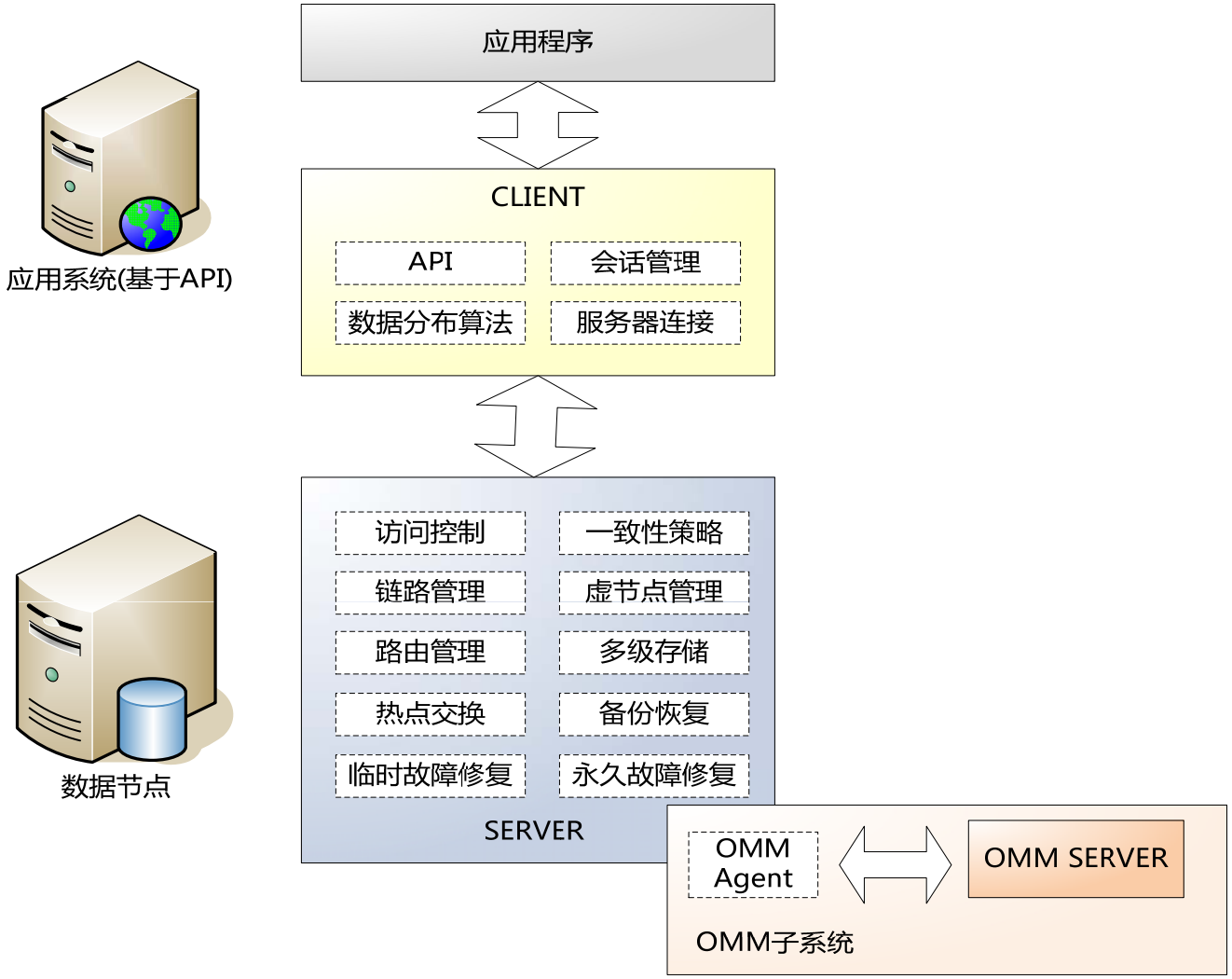
无中心化的配置管理，避免了管理控制台的单点故障，并且支持Web方式管理

一致性优化，保证可用性前提下一致性程度较高

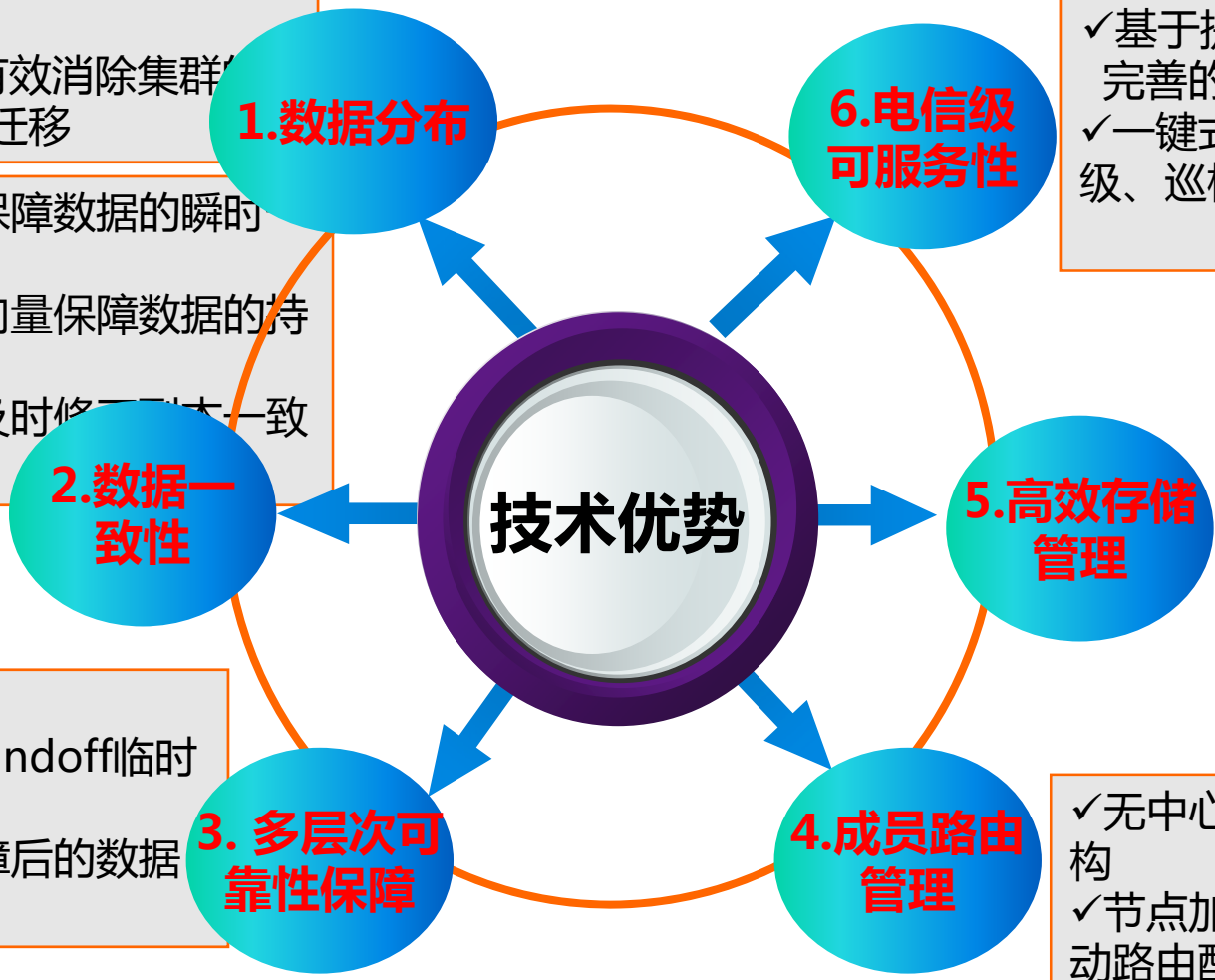
支持高速顺序持久化方式，高效持久化、多磁盘或磁阵的方式

平台通用性，API的多平台支持

分布式存储产品架构



分布式存储产品的技术优势



✓基于一致性Hash和虚节点的数据分布
✓结合虚节点，有效消除集群
热点,更方便数据迁移

✓通过版本向量保障数据的瞬时
致性
✓通过集群时钟向量保障数据的持久一致性
✓通过读修复，及时修
性

✓多副本保障
✓基于Hinted-handoff临时故障解决方案
✓磁盘或节点故障后的数据重构

✓基于提供Dashboard完善的系统运行监控
✓一键式安装部署、升级、巡检、信令跟踪

✓多种存储方式支持和各自优势互补
✓支持内存/SSD/硬盘三种方式混合存储

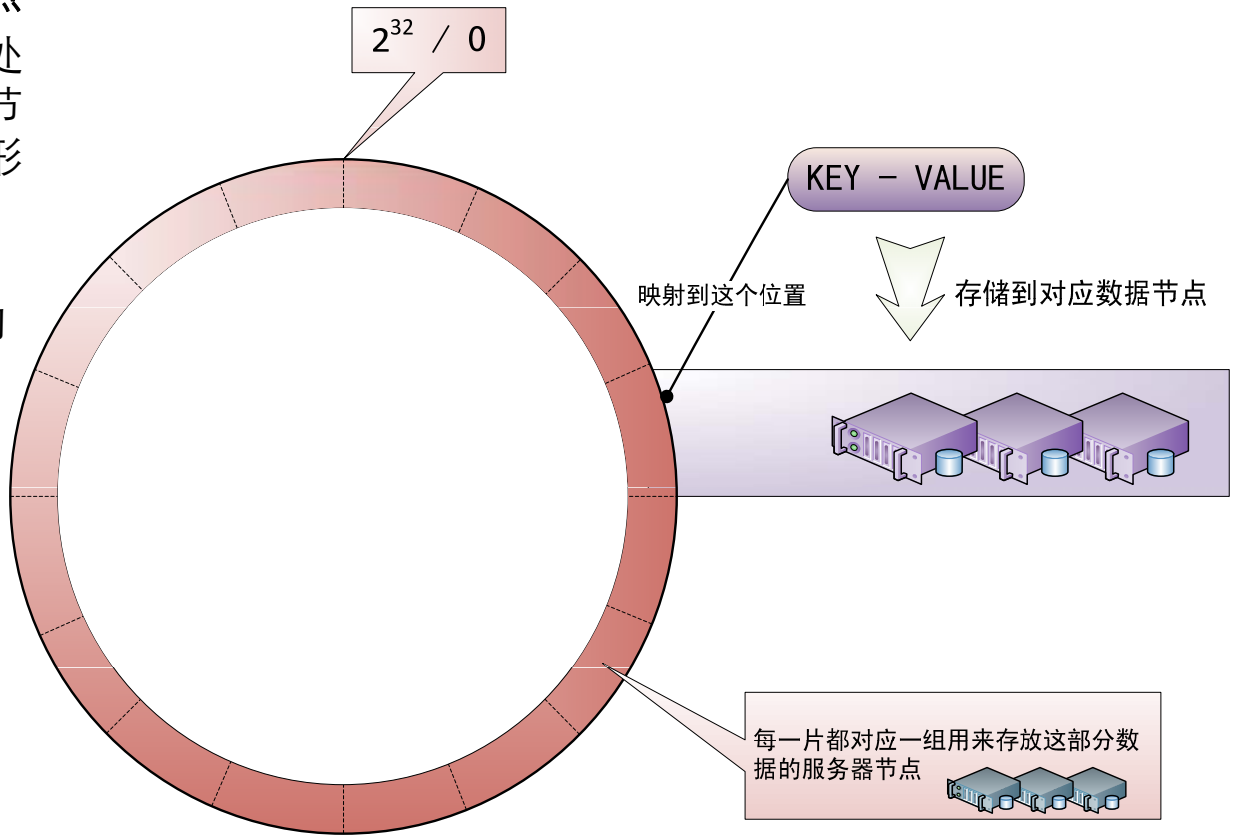
✓无中心化的分布式架构
✓节点加入和退出的自动路由配置
✓路由变更和数据迁移服务不中断

数据分布：一致性哈希和虚节点

一致性Hash和虚拟节点：

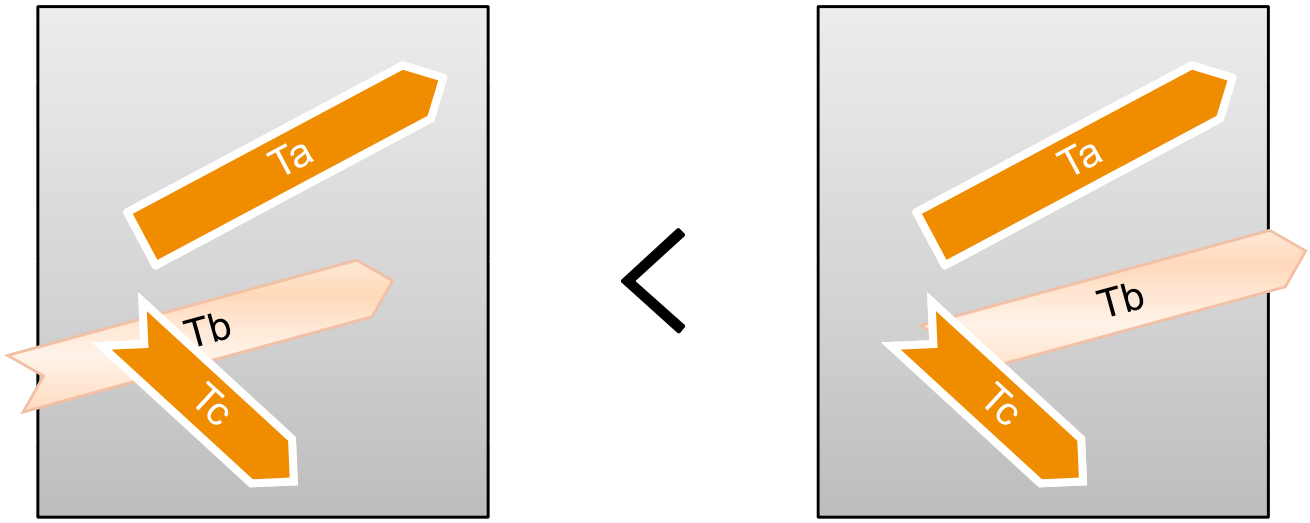
- 将2^32的Hash空间等分为若干分片，每个分片即是一个**虚节点**
- 根据各物理节点性能差异配置处理不同数量的虚节点，这些虚节点在物理节点上的部署关系即形成虚节点的**路由**

通过一致性Hash和虚节点相结合的方式，实现了数据在集群的均匀分布以及数据服务器节点热点的消除



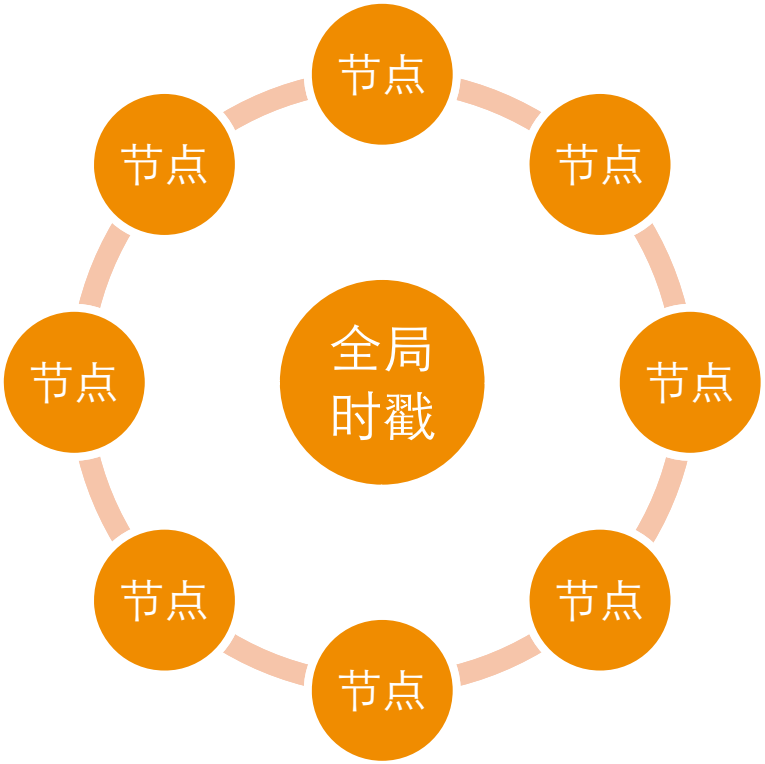
一致性：版本向量瞬时一致性保障

- 版本向量
  - 版本向量和数据变更相关，向量采用数组结构 (Ta, Tb, Tc)，维数和副本数保持一致
  - Key的每次变更递增对应的向量
  - 同一个Key的多个副本不一致时，可通过版本向量之间的大小比较来判断数据版本的新旧
  - 当版本向量不可比较时，参考全局向量进行辅助判断



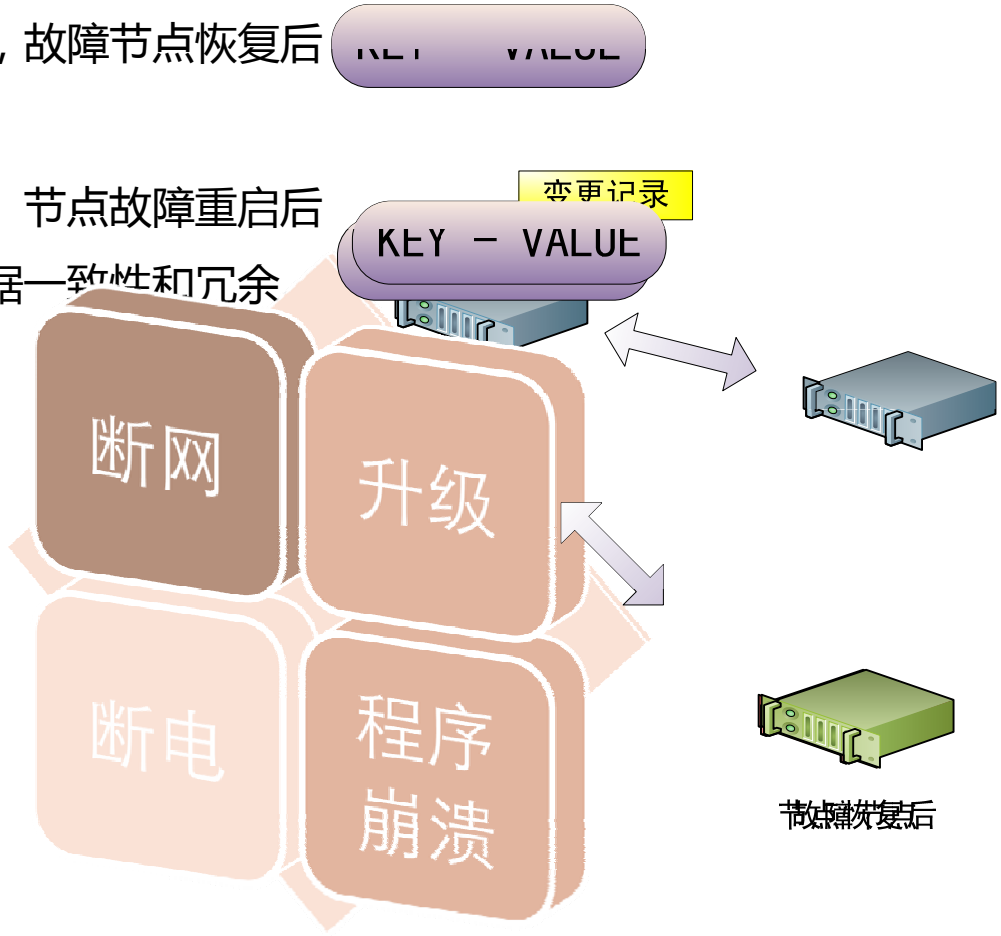
一致性：分布式全局时间向量持久一致性保障

- 全局向量
  - 全局向量和时间相关
  - 每台节点上初始启动时，全局向量都从0开始，按固定周期递增
  - 集群多个节点间通过Gossip对全局向量进行协商，按照向量的大小校正本机看到的全局向量
  - 全局向量节点持久化存储
  - 全局向量辅助版本向量，参与两份数据新旧关系的判断

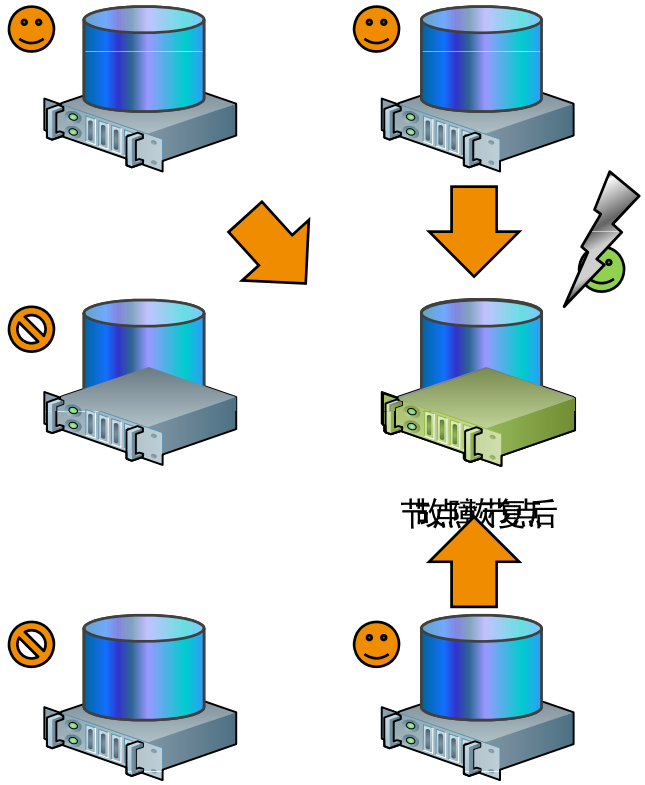


数据可靠性：临时故障后的数据修复

- 当物理节点发生临时性的可恢复故障时，由数据处理节点缓存故障期间所有的数据变更记录，故障节点恢复后根据缓存的变更记录对数据进行修复。
- 临时故障修复机制使节点间断链重连、节点故障重启后能自动的修复数据，保证了节点间数据一致性和冗余



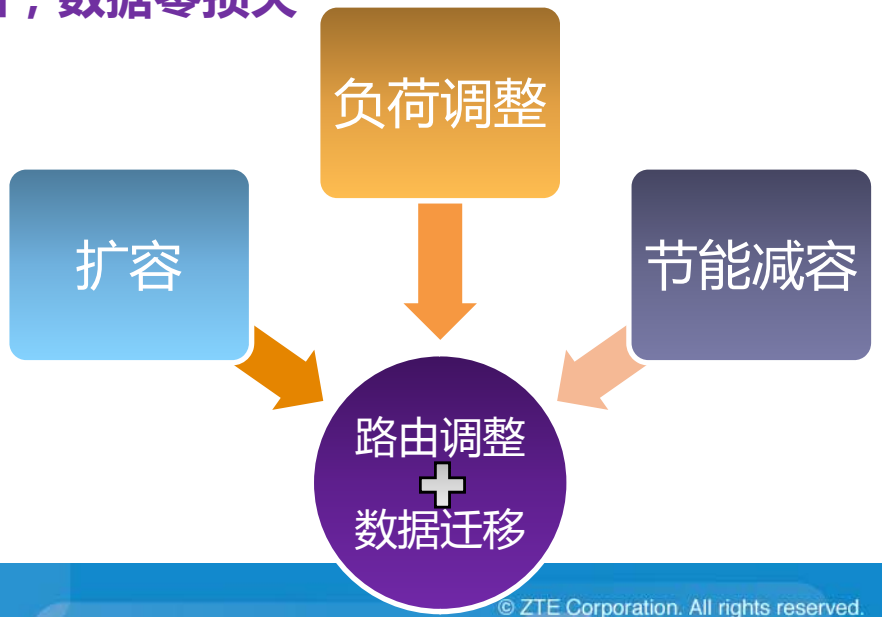
数据可靠性：数据丢失后多机冗余智能修复



利用集群中数据多个副本，对故障节点进行数据重建，以保持整体的Quorum算法的完整性

成员路由：数据分布策略变更和数据迁移

- 在各物理节点负荷不均匀，或集群扩容、减容时，需要对集群的路由信息进行调整，数据随着路由的调整，数据也会在节点间进行迁移
- 手工和自动两种模式
- 路由变更和数据迁移后，负荷仍能够在节点间均衡分配
- 路由变更和数据迁移的过程是原子的，中途可回退，最重要的一点是执行变更的过程对应用访问透明，做到**访问不中断，数据零损失**



存储管理：内存-SSD-硬盘 各自优势结合

- 支持内存/SSD/硬盘，三级存储。SSD用来做内存的扩展，硬盘作为持久化存储介质，根据需要任意组合

存储管理：内存缓存的有效性

- 内存缓存属于昂贵资源，如何有效利用?

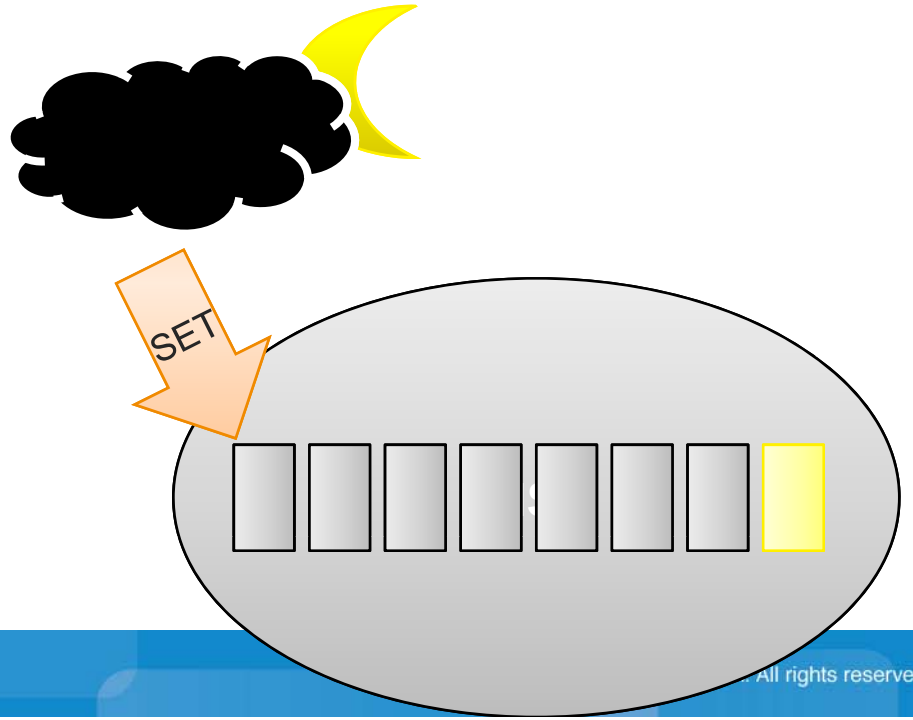
碎片消除，采用定长内存片分级管理的方法，有效的消除内存碎片，提高内存使用效率。

热点交换，对数据访问进行统计，内存中优先保存访问频率高的数据，提高数据在内存中的命中率。

智能分配，基于各应用空间对内存使用的统计，访问频繁的应用得到更多的内存使用，访问较少的应用分配的内存也少，智能地动态调整，提高内存使用的有效性

存储管理：可插拔的持久化引擎

- 适用场景，海量小文件存储，配合分布式文件系统实现海量云存储
- 可插拔、存储接口标准化，支持BDB、高速顺序持久化存储引擎
- 高速顺序持久化存储引擎
  - ✓ 顺序写
  - ✓ 索引
  - ✓ 数据断点整理
  - ✓ 智能索引恢复



不能忽视的可服务性

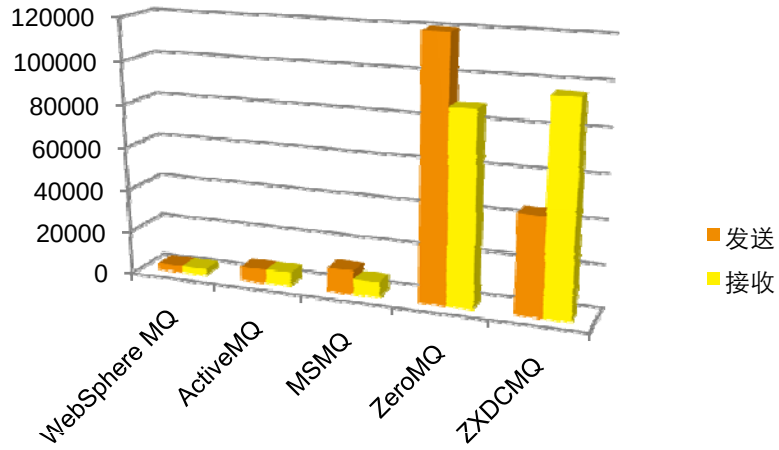
- 提供Web方式和telnet方式的控制维护台,提供了自动管理和监控系统拓扑结构，负载均衡，性能，事件和警报
- 一键安装、升级、巡检
- 使用工具以“复制粘贴”的方式部署集群
- 提供Dashboard，方便查看系统运行状态
- 集群出现故障时短信或邮件方式告警
- 在线的数据存取流程跟踪

高性能分布式自增计数器

- 分布式存储产品提供分布式架构下的 计数器，用来实现高并发场景下的计数功能
- 使用场景：
  - ✓ 数值加减运算
  - ✓ 全局访问计数器
  - ✓ 全局UUID生成
  - ✓ 全局流控
- 特点是高性能、高可用、高准确性
  同时支持海量计数器

分布式消息队列

- 基于分布式计数器技术，将消息队列和消息本身都抽象为KV形式，利用计数器生成消息的索引，实现了轻量级分布式消息队列功能
  - ✓ 支持点对点的消息发送接收
  - ✓ 支持多对多的消息发送接收
  - ✓ 支持同步和异步的消息收发方式
  - ✓ 支持关注方式和主动获取方式的消息接收
  - ✓ 消息队列访问负荷在集群各节点中均衡



分布式消息队列特征和优势

关键特性

- 高性能，单队列IOPS大于8万
- 大容量，支持几千万消息队列
- 高可靠，不丢消息，不重复
- 可扩展，轻松扩展
- 有序性，先进先出
- 易用性，应用不需要关注通讯和分布式细节

技术优势

- 把效率较低的同步应用变成高效的异步应用，把复杂的紧耦合系统变为简单的松耦合系统
- 实现不同系统和不同应用间的高效的消息交互
- 社区消息的广播和订阅架构进行平台化，应用开发更easy

目录

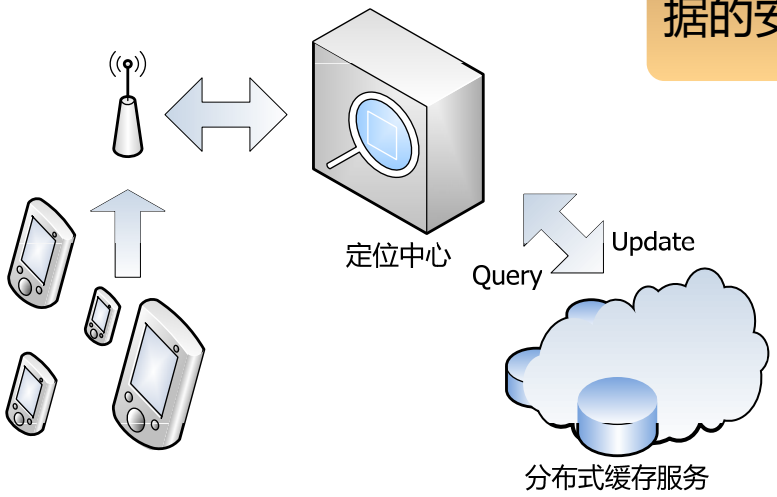
- NoSQL的前世今生
- NoSQL的理论和关键技术
- NoSQL典型产品与分析
- NoSQL之分布式存储产品
- 电信业务应用实践

XX运营商LBS产品应用

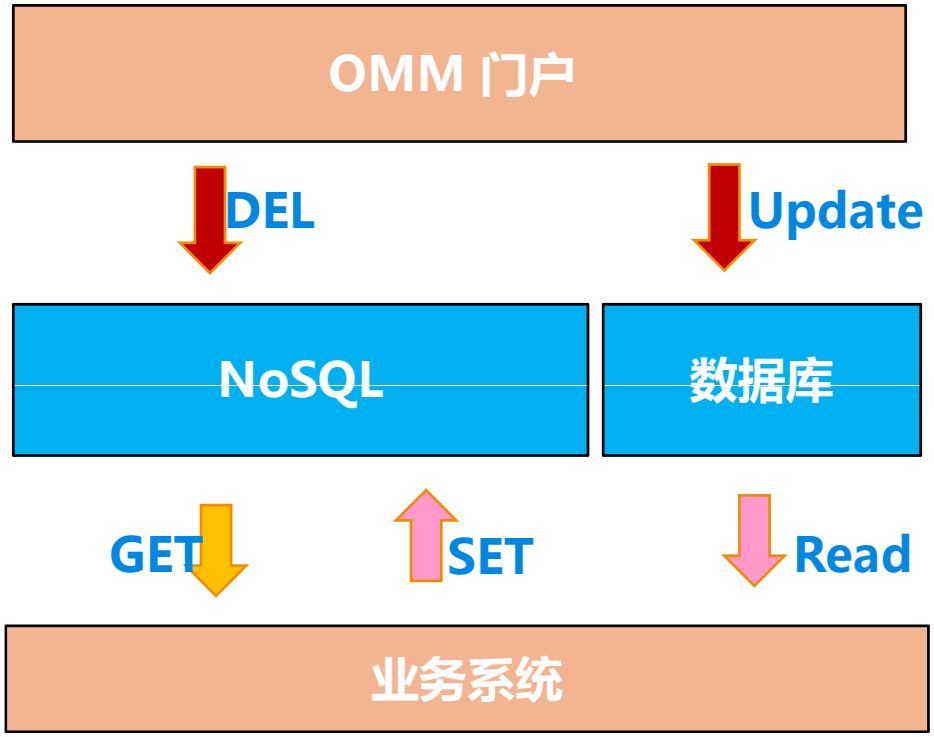
每秒超过20万次的位置信息的存储和读取，实时性要求非常高，普通的数据库根本无法承受

通过分布式缓存的存储，6台服务器，3个副本，轻松的支持到每秒30万次的访问

定期的把内存数据备份到磁盘，保证了缓存数据的安全，避免系统异常带来的震荡



SDP产品应用



分布式缓存

◆热点数据的存储，预热后全部基于缓存提供数据访问服务

分布式消息队列

◆海量消息的缓存，消峰填谷
◆失败重发

分布式KV DB

◆实现海量小尺寸文件存储，提升IO效率

分布式计数器

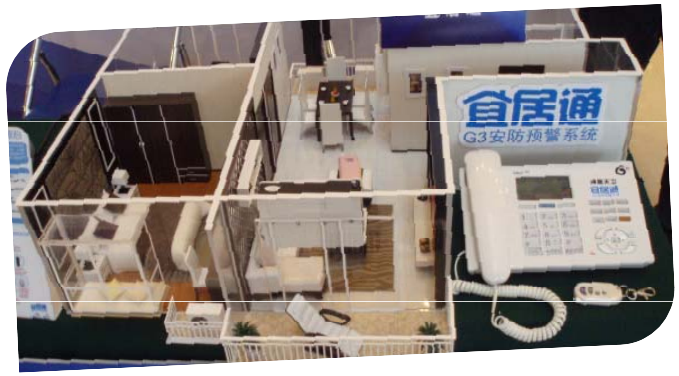
◆全局计数器可以实现分布式的多节点的统计，流控，和全局ID的生成

物联网应用

物联网数据控制信息和变更记录、状态信息等是海量的，普通数据库捉襟见肘

将物联网实体的状态信息存入分布式存储产品中，通过使用分布式存储就能够满足海量实时存储需求

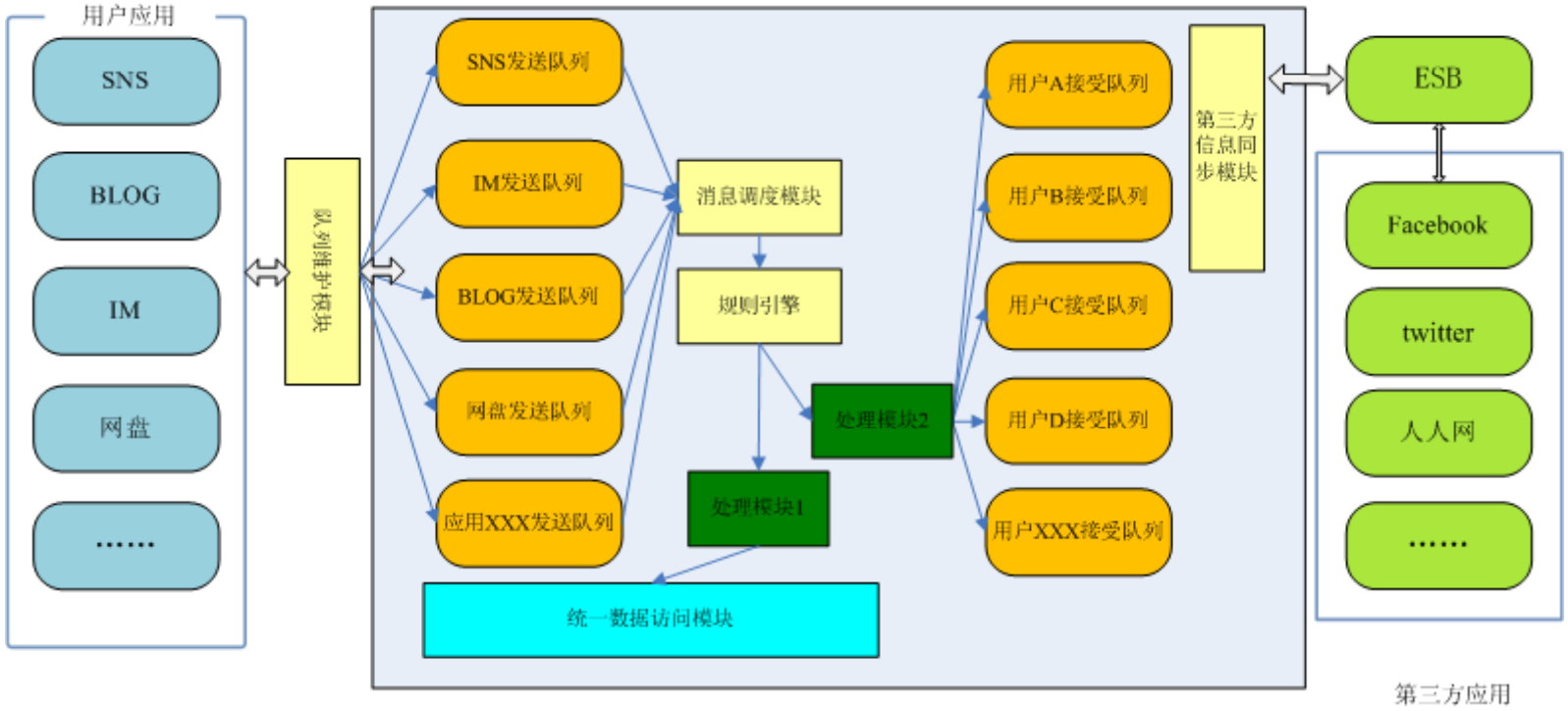
无论是在时延，并发量，还是容量，已经可扩展性上，分布式存储产品比传统存储都更具有优势



移动互联网应用

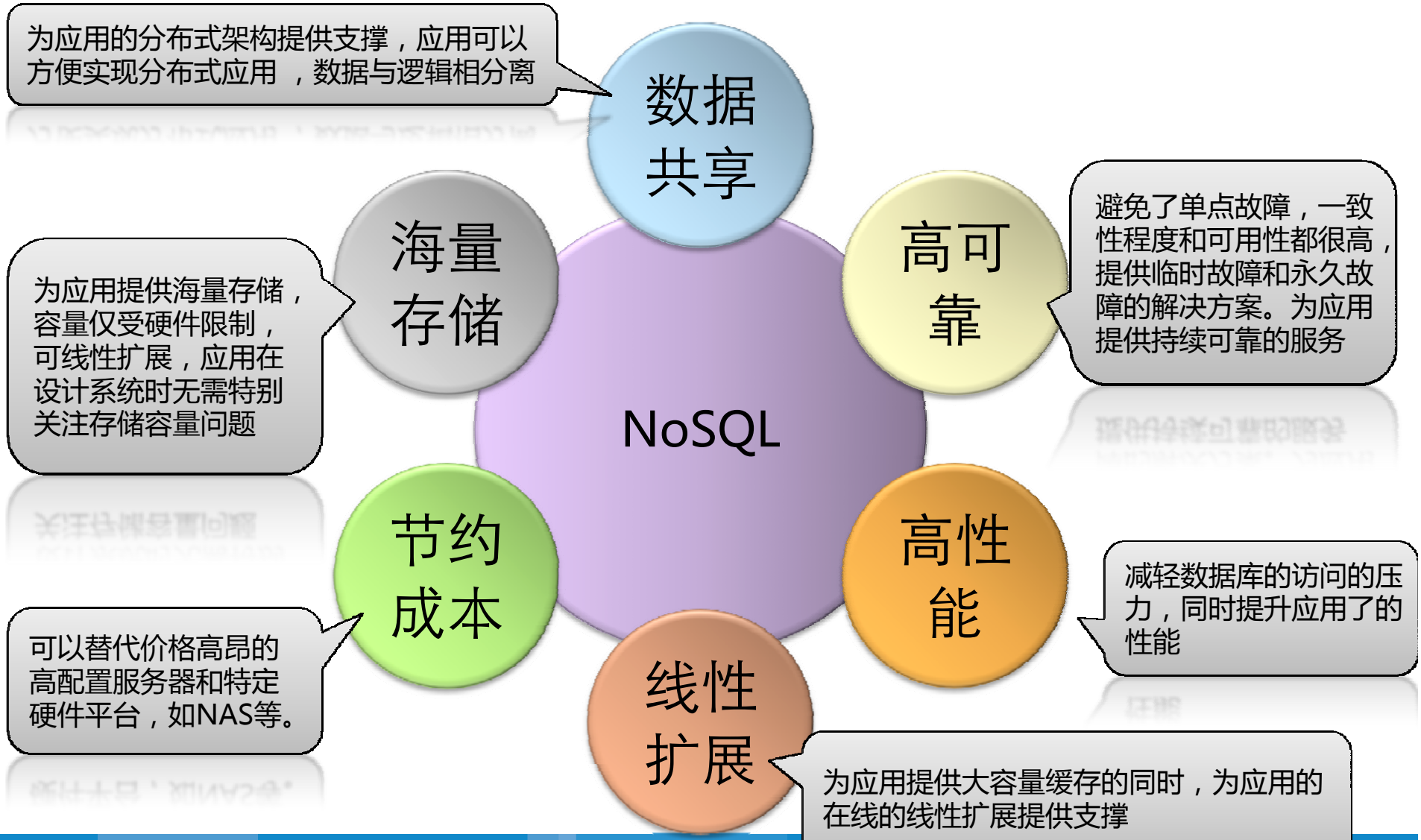
内存方式	持久化方式	计数器
• 网店商品搜索，详情，促销 • 社区开发资源，创意，资讯，终端，公告，合作伙伴 • WAP的好友推荐，品牌，促销，搜索；手机服务端的促销，详情，机型，图片等	• 浏览记录目前保留最近的20条 • 好友动态（收藏，浏览等8个	• 社区浏览次数，使用硬盘方式 • 渠道商，广告位注册、点击率统计 • 邀请好友、短信发送次数，使用内存方式

用户中心消息队列的使用



每个应用一个消息发送队列，每个应用的发送队列保存消息，消息有消息唯一id，用户id，消息类型，消息到期时间，消息title，消息摘要；消息调度模块，从发送队列有消息以后，消息调度模块接受消息，并根据消息类型，规则引擎做不同的处理。
通过消息队列降低了系统之间的耦合，优化了系统的架构，同时也简化了开发的难度。

总结



ZTE中兴

Thanks!

Bringing you closer