DTCC 2014中国数据库技术大会
DATABASE TECHNOLOGY CONFERENCE CHINA 2014
大数据技术探索和价值发现

百分点内存数据库架构演变

武毅

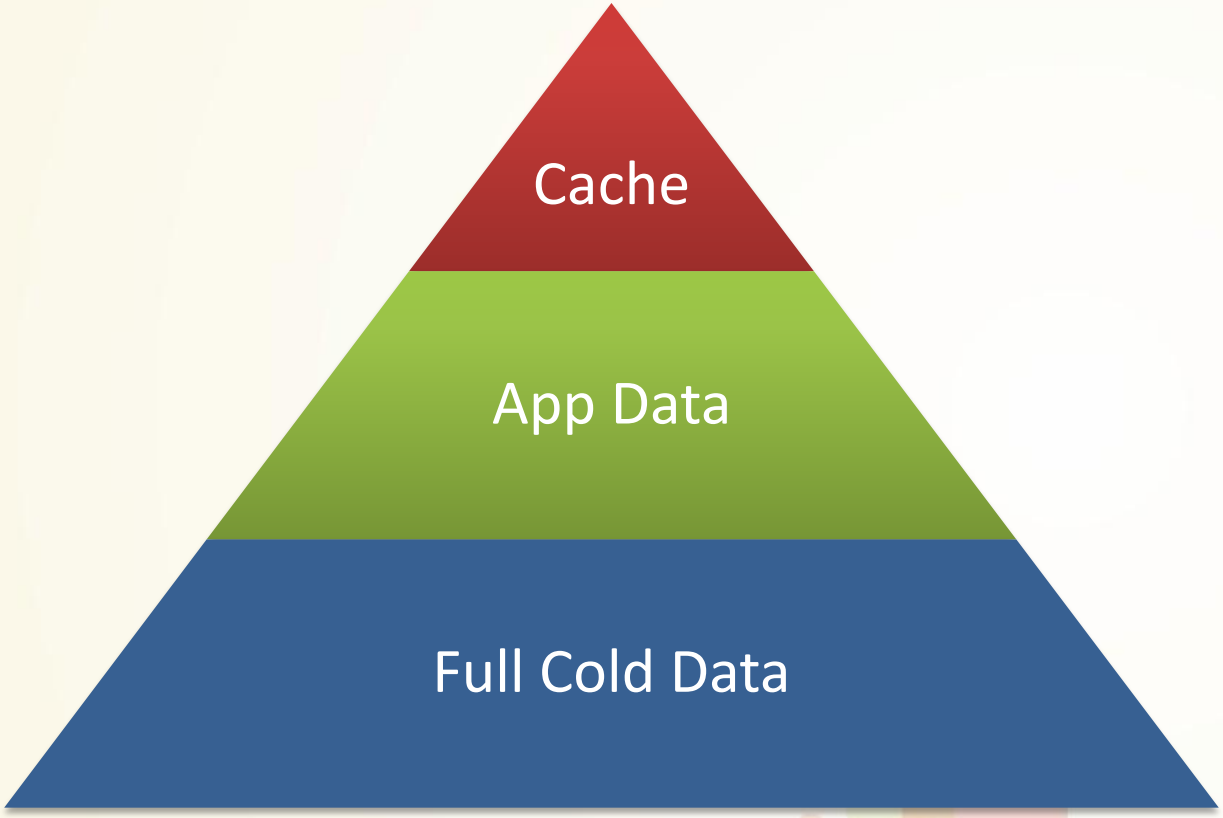
摘要

- **内存计算的趋势**
- 百分点推荐引擎BRE的内存计算
- 百分点内存数据库演变

内存计算的趋势

- 数据展现上，时间就是金钱
- 面临着处理海量的数据
- 磁盘IO成为并行计算的瓶颈
- 针对不同行业，应对各种业务需求，需要从不同的维度去处理，分析数据
- 对内存数据库的需求

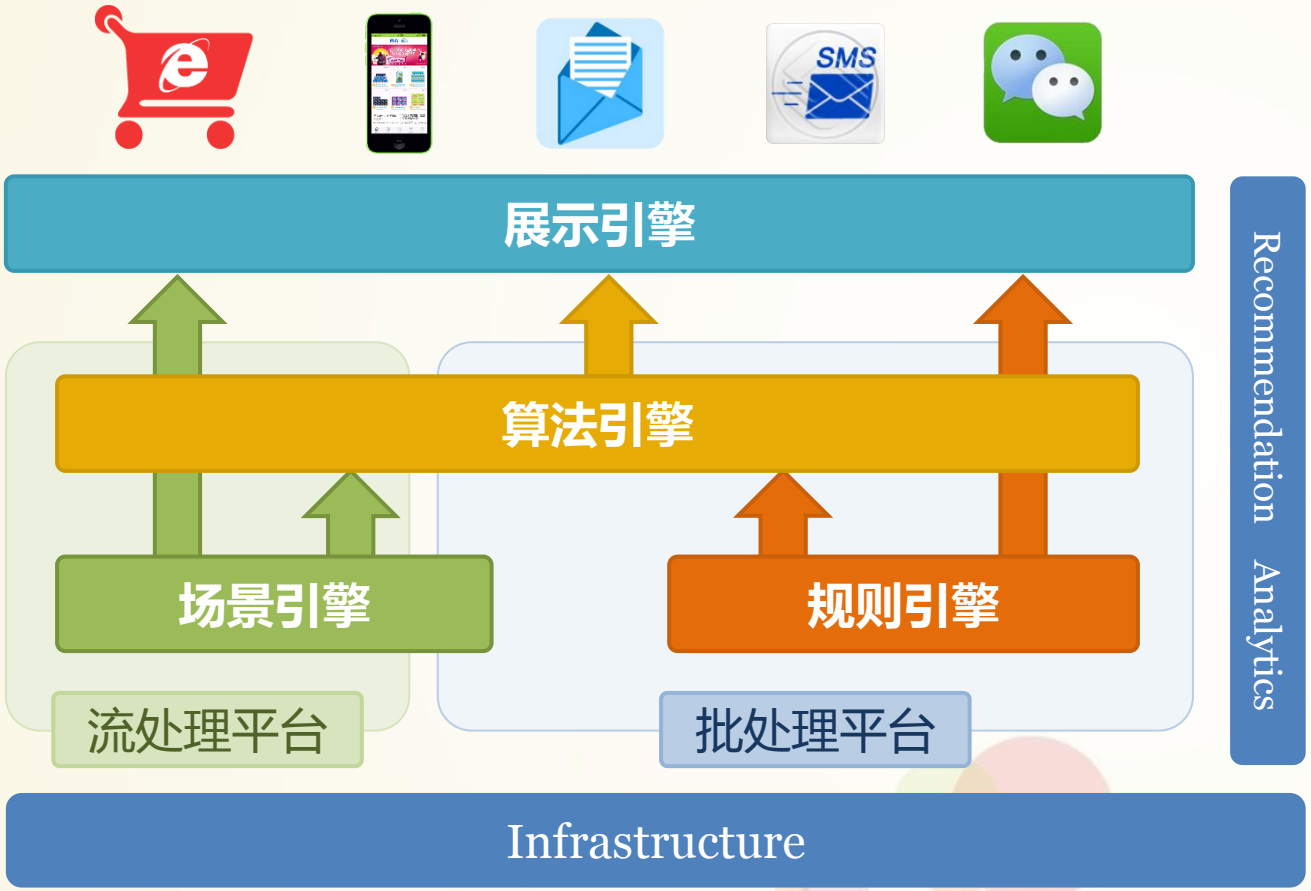
互联网公司数据金字塔



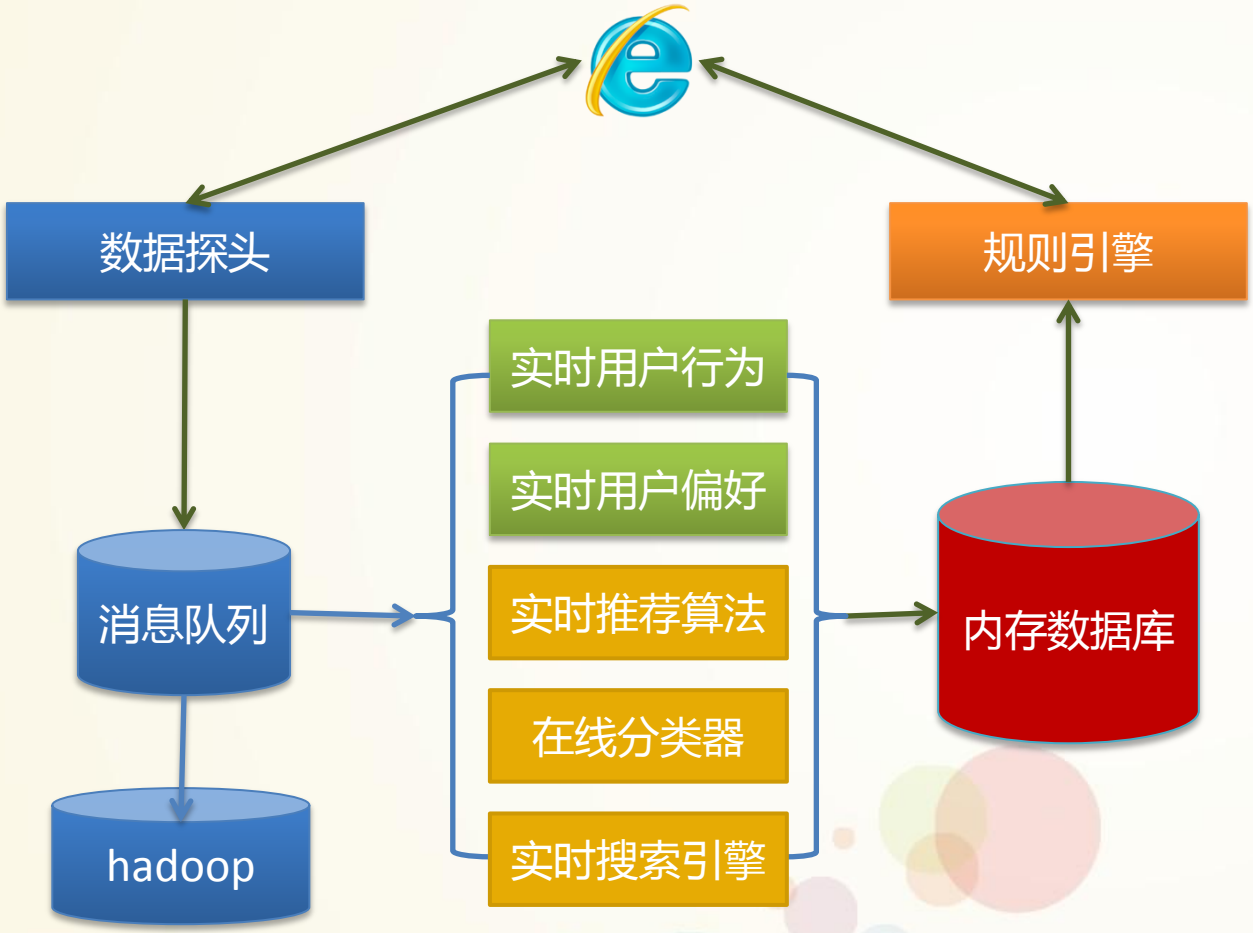
摘要

- 内存计算的趋势
- **百分点推荐引擎BRE与内存计算**
- 百分点内存数据库演变

BRE原理


SMS
展示引擎
算法引擎
场景引擎
规则引擎
流处理平台
批处理平台
Recommendation Analytics
Infrastructure

BRE实时计算：lambda架构示意



BRE基于内存数据库的实时计算

实时用户偏好	实时推荐算法	在线分类器	实时规则引擎	实时搜索引擎	流计算
用户购物状态	大规模矩阵计算	商品自动分类	最优化算法	商品索引	带有时间窗口的存储和计算
用户购物目标	大规模图计算	舆情评估	推荐结果混合	资讯索引	推荐效果统计分析
用户购物周期	统计类算法		推荐结果排序		监控信息分析
			A/B Test		

内存数据库是BRE的主存

- 数据实时更新
  - 用户行为、用户偏好、商品资讯信息、推荐算法结果、集群监控数据…
- 海量数据
  - 十几种类别、十亿量级条目数、TB量级存储量
- 高并发、高吞吐量
  - 每秒十万量级读写次数、GB量级数据量
- 高可靠和高可用
  - 数据固化、容灾、备份

摘要

- 内存计算的趋势
- 百分点推荐引擎BRE与内存计算
- **百分点内存数据库演变**

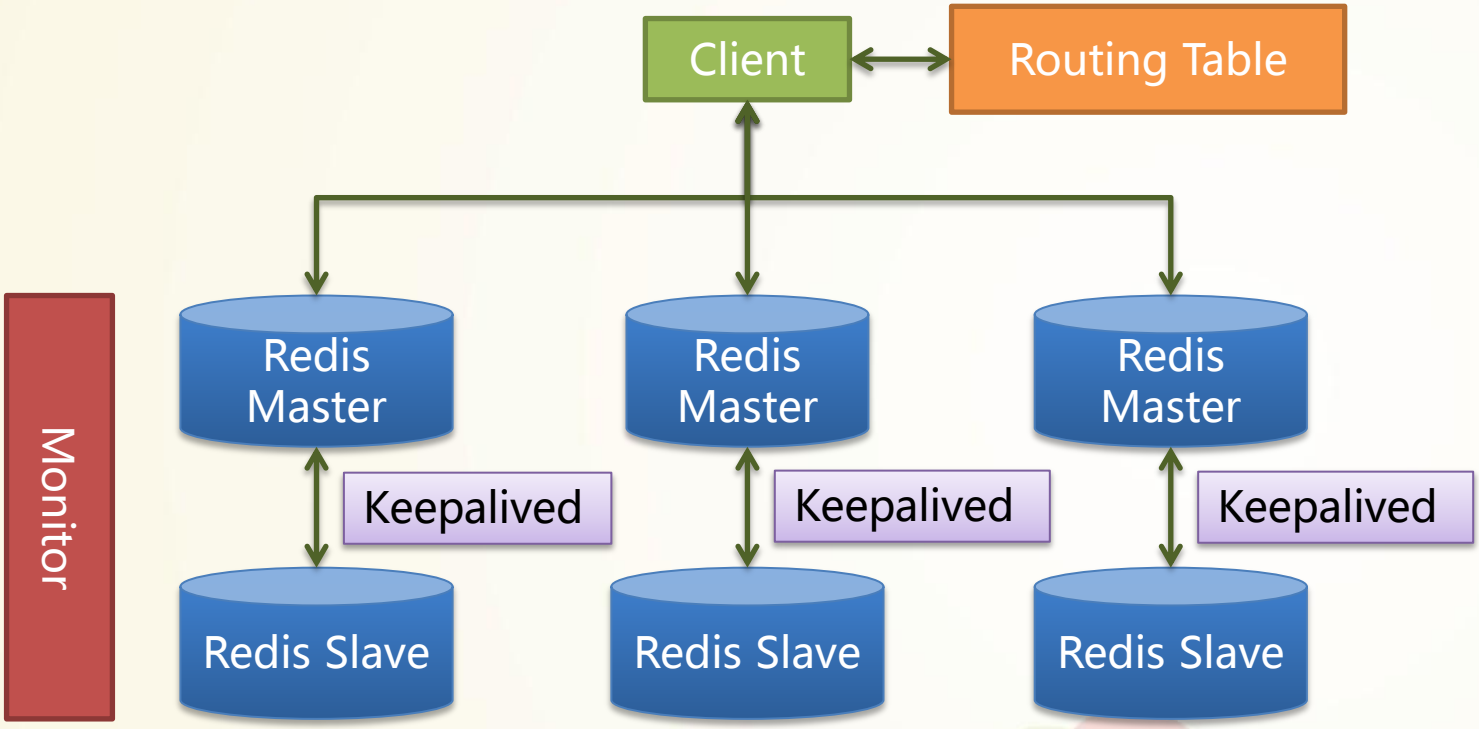
百分点内存数据库演变

BRE 0.x	Redis + Route-Table
BRE 1.x	Memcached + Mysql + Zookeeper
	Memcached + Redis + Zookeeper
BDM、BRE 2.x	Redis+ MongoDB + Zookeeper

BRE 0.x的内存数据库：需求

- 2010.1 - 2012.3
- 数据量
  - < 100G
- 并发和吞吐量
  - < 1万RW/秒，< 2M/秒
- 承载的应用数目不多

BRE 0.x的内存数据库：架构



BRE 0.x的内存数据库：说明

- 分布式
  - 数据按客户分组
  - 每个数据分组由唯一的Redis实例存储
  - 用路由表记录分组和Redis实例的对应关系
- 高可用
  - Redis主从 + Keepalived
  - 通过info和dbsize命令监控Redis状态
- 数据固化
  - 主无持久化，从定期持久化

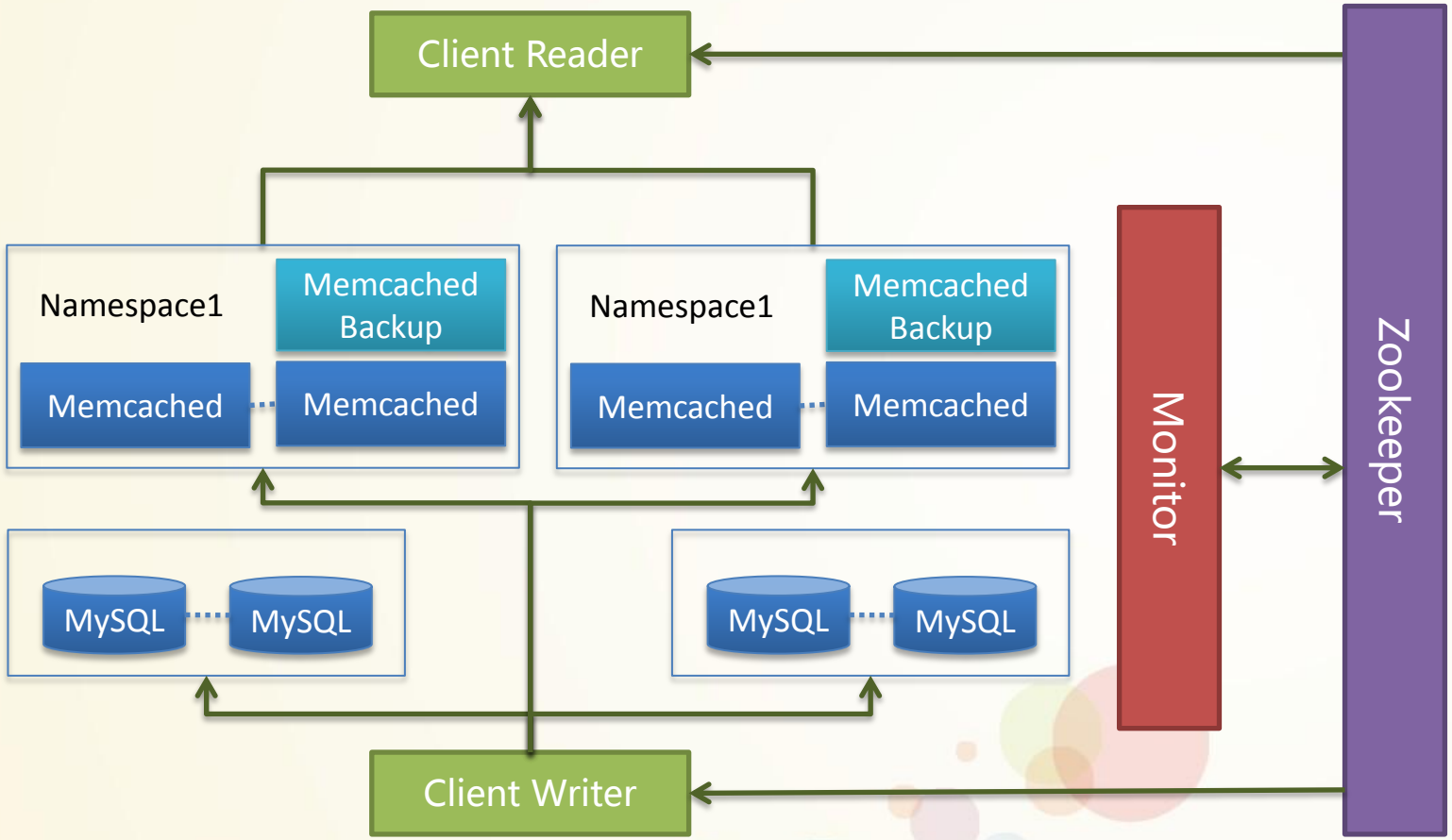
BRE 0.x的内存数据库：局限

- 需要手工维护路由表
  - 容易导致负载不均衡
  - 人工成本高
  - 扩展性较差

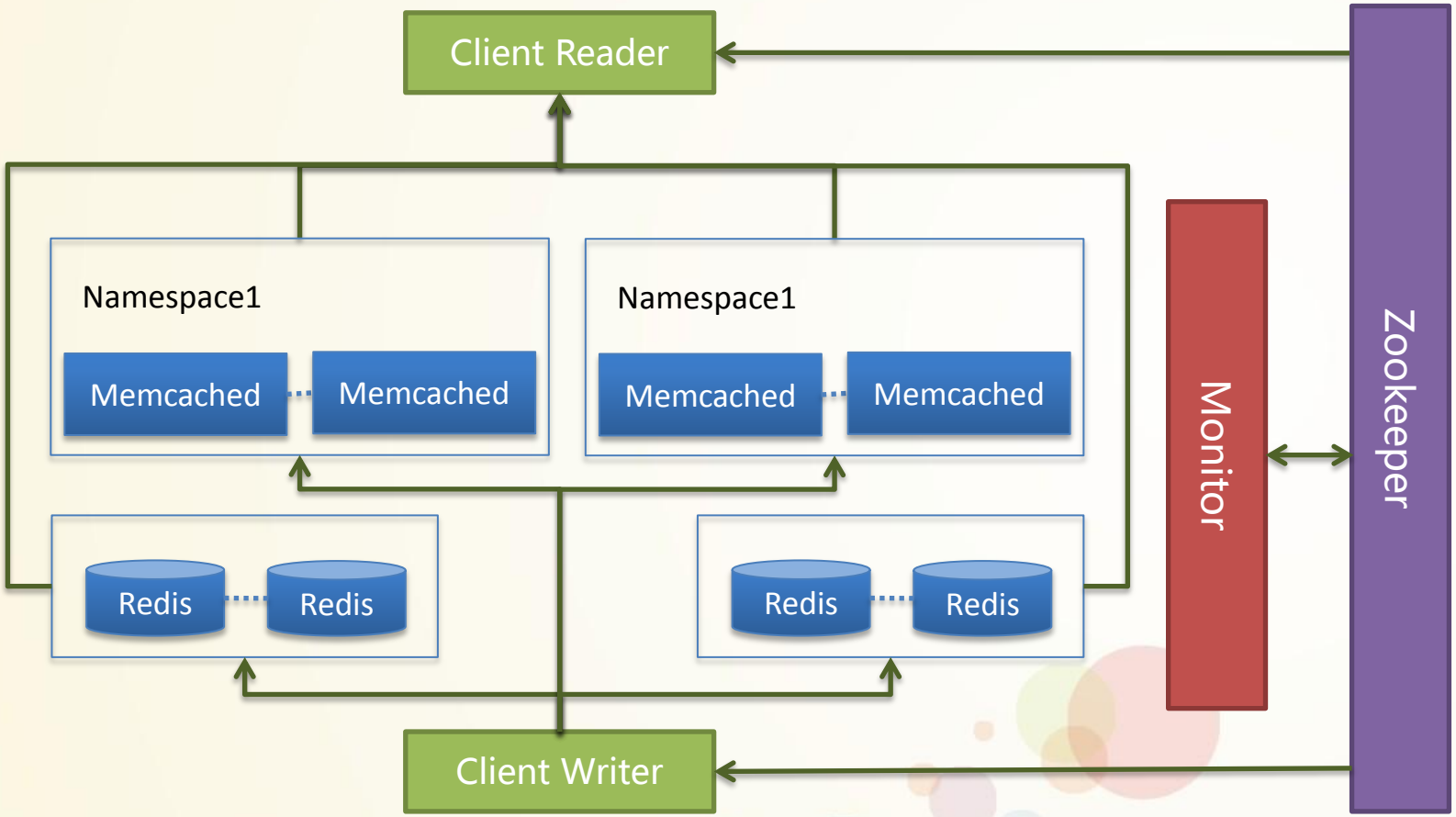
BRE 1.x的内存数据库：需求

- 2011.7 – 今
- 数据量
  - ~ 700G
- 并发和吞吐量
  - ~200万RW/秒，~400M/秒
- 承载很多应用
  - 应用间数据需要隔离
- 良好的负载均衡和水平扩展性

BRE 1.x的内存数据库：架构I



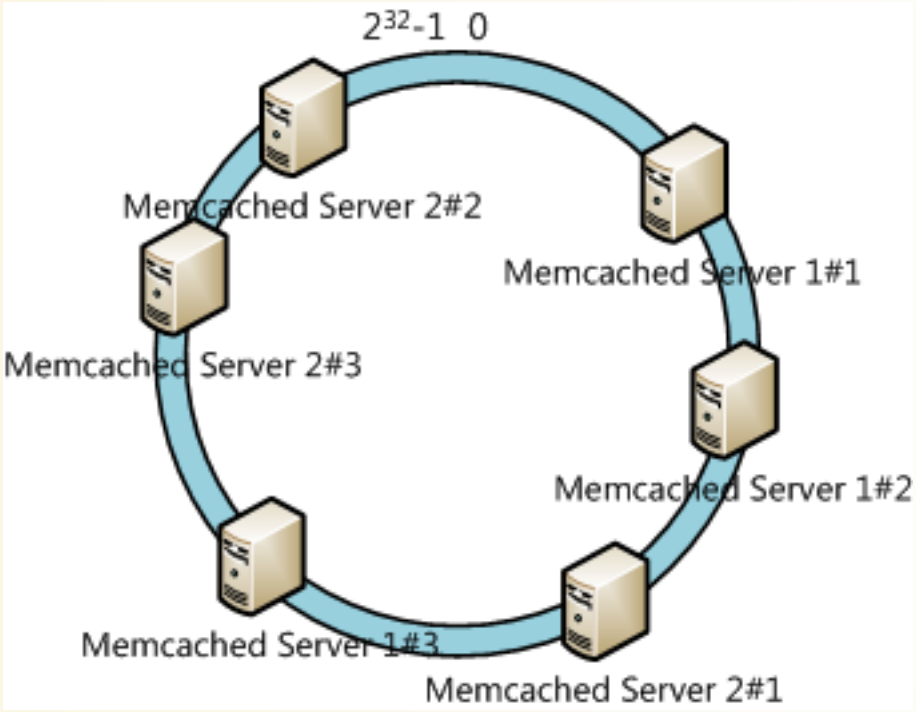
BRE 1.x的内存数据库：架构II



BRE 1.x的内存数据库：说明

- 分布式
  - 集群分为多个Namespace，物理上隔离
  - 同一Namespace内使用一致性Hash（Ketama）及虚结点机制均匀分布数据
  - 写入的数据通过key值做逻辑隔离
- 高可用
  - 利用组件内建工具（Memcached stats、Redis info）实时监控集群信息
  - 利用Zookeeper实时更新集群状态，并通知客户端
  - 自动故障迁移，一开始采用备份机，后期采用Redis
- 数据固化
  - 写入数据时异步固化
  - MySQL固化速度慢，最终采用Redis

BRE 1.x的内存数据库：数据重分布



- 虚拟结点导致物理机器数目改变时数据必须重新分布
  - 每台物理机器上有多个虚结点
  - 虚结点无法有效区分

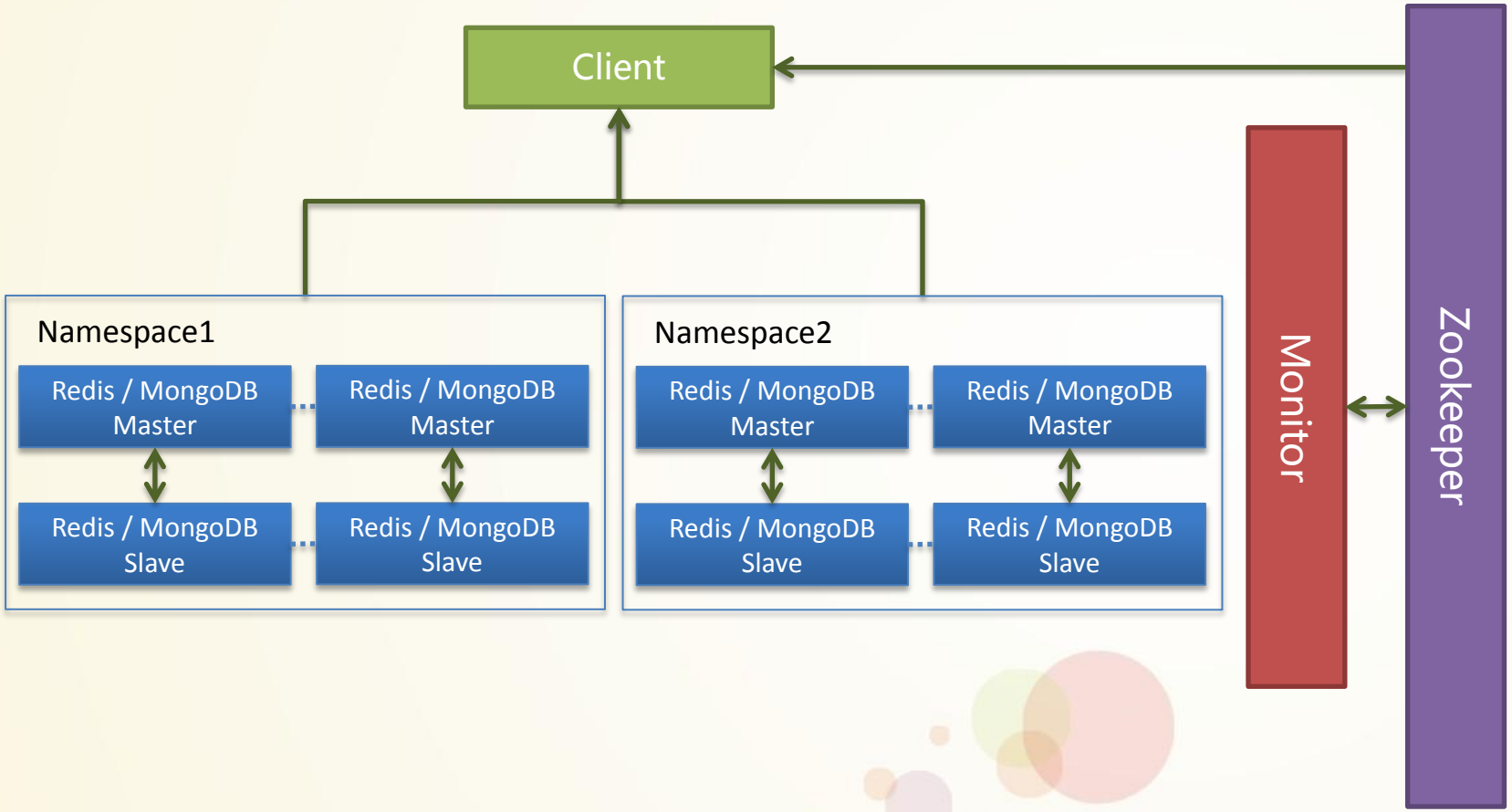
BRE 1.x的内存数据库：局限

- Memcached不能作为数据库
  - 无法固化数据
  - 无法枚举数据
  - 无法很好的控制数据过期
- 读写分离导致系统复杂
- 简单的KV不能满足需求
  - 大Value导致网卡瓶颈
  - 数据序列化/反序列化消耗系统资源
- 扩容不易
  - 虚结点的使用导致需要重新计算所有数据分布

BDM的内存数据库：需求

- 2013.7 – 今
- 更多的应用
  - BRE的底层架构演变为百分点大数据平台（BDM）
  - BRE（2.x）成为BDM上的一个应用
- 数据量
  - ~ 1.5T
- 并发和吞吐量
  - ~500万RW/秒，~1G/秒
- 更多的数据结构支持
- 更简便的的扩容

BDM的内存数据库：架构



BDM的内存数据库：说明

- 多种数据结构
  - Redis：KV、List、HashMap、Set …
  - MongoDB：JSON文档
- 分布式
  - 集群分为多个Namespace
  - 同一Namespace内使用一致性Hash及虚结点机制均匀分布数据
  - 利用Redis和MongoDB中的数据库作为（半）虚结点，扩容时只需重分布某些数据库中的数据
  - Small instance, more instance

BDM的内存数据库：说明

- 高可用
  - 利用Redis Sentinal 、MongoDB mongostat实时监控集群状态
  - Redis Sentinal
    - 记录集群状态、状态变化通知、控制Redis故障时切换主从
    - 多个Sentinal冗余、高可用，可用性投票
- 数据固化
  - 数据分层
    - 不可舍弃无法再生的数据
    - 可再生数据
    - 缓存数据
  - Redis主不持久化，从定期持久化

BDM的内存数据库：现状

- 集群
  - Redis
    - 24台服务器，单机144G内存
    - 每台服务器4个Redis实例
    - 3个Sentinal实例
  - MongoDB
    - 4台服务器，单机144G内存
    - 每台服务器4个MongoDB实例
- 数据
  - 42亿，1.5T
  - 单机读峰值50K/s，写2K/s
  - 单机出口流量70MB/s, 入口流量峰值：20M/s

BDM的内存数据库：最终一致性

- 读写异步模型（lambda架构）
- Master 挂掉，此时还未同步到Slave的数据
  - 从消息队列中回放数据恢复
  - 算法数据再生，持续输出
  - Slave升级为Master，原Master恢复后作为Slave
- Slave挂掉，恢复后数据重新同步

百分点内存数据库：其他

- In-Memory-Index内存索引
  - 轻量级索引
  - 异步准实时建立
- Half-Memory-Sparse-Matrix(TC、KC)
  - 使用嵌入式kc作为算法引擎内存数据库

Q&A

THANKS

We are always hiring!
Email to: yi.wu@baifendian.com