

内存数据库与NoSql数据库

传统RDBMS 关系型数据库

- 表结构（列，行）
- 关联（OneToMany, ManyToOne, OneToOne）
- ACID:
 - 原子性(Atomicity) 整个事务中的所有操作，要么全部完成，要么全部不完成，不可能停滞在中间某个环节。
 - 一致性（Consistency） 在事务开始之前和事务结束以后，数据库的完整性约束没有被破坏。
 - 隔离性（Isolation） 两个事务的执行是互不干扰的，一个事务不可能看到其他事务运行时中间某一时刻的数据。
 - 持久性（Durability） 在事务完成以后，该事务所以对数据库所作的更改便持久的保存在数据库之中，并不会被回滚。

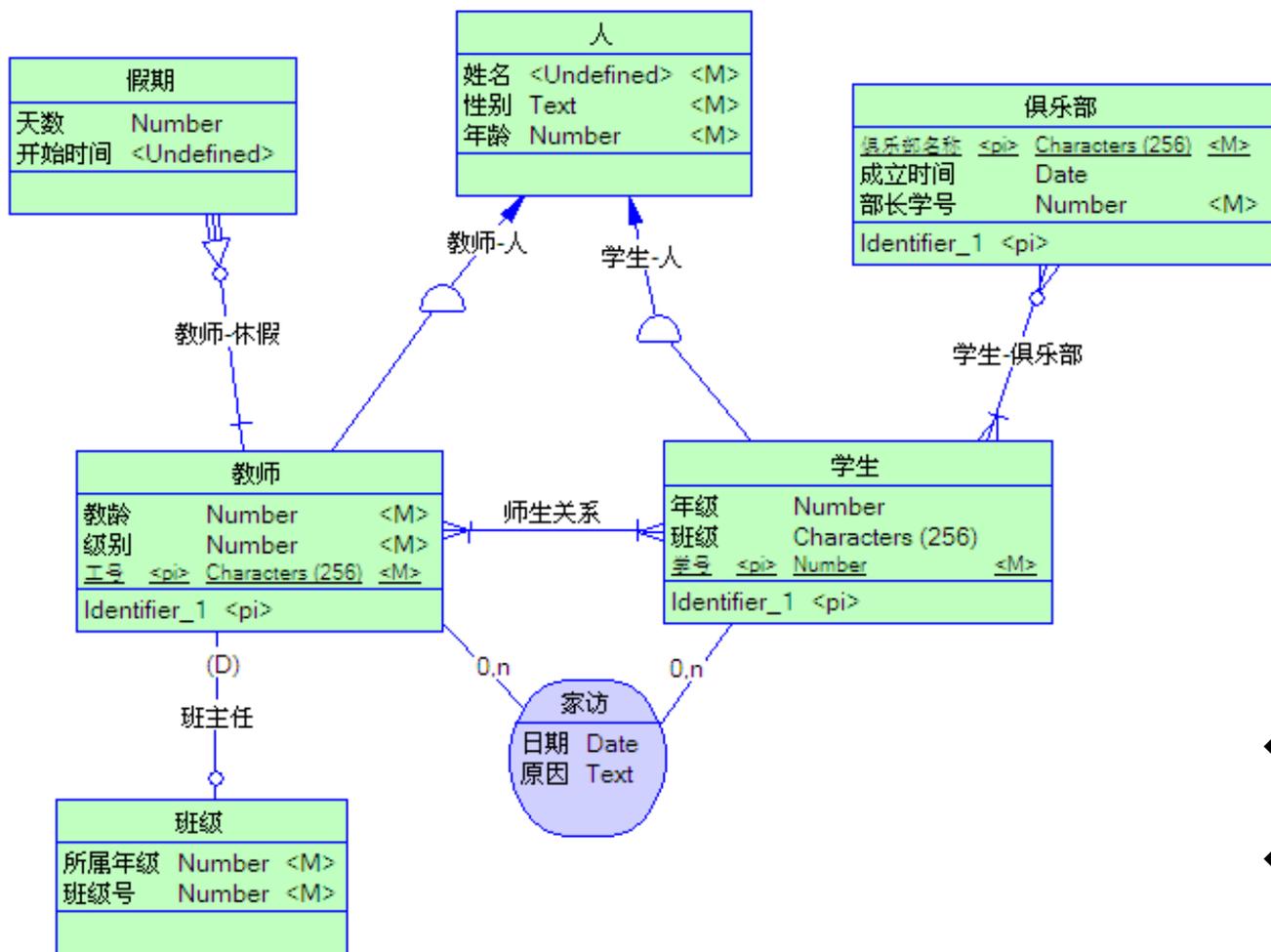
传统RDBMS表结构

ID	姓名	年龄	特长	爱好	好友	邻居	...
1	张三						
2	李四			喜欢打麻将			
3	王五			钓鱼			
4	甲						
5	乙						
6	丙						
7	丁						
8	戊						
9	己						
10	庚						

缺点:

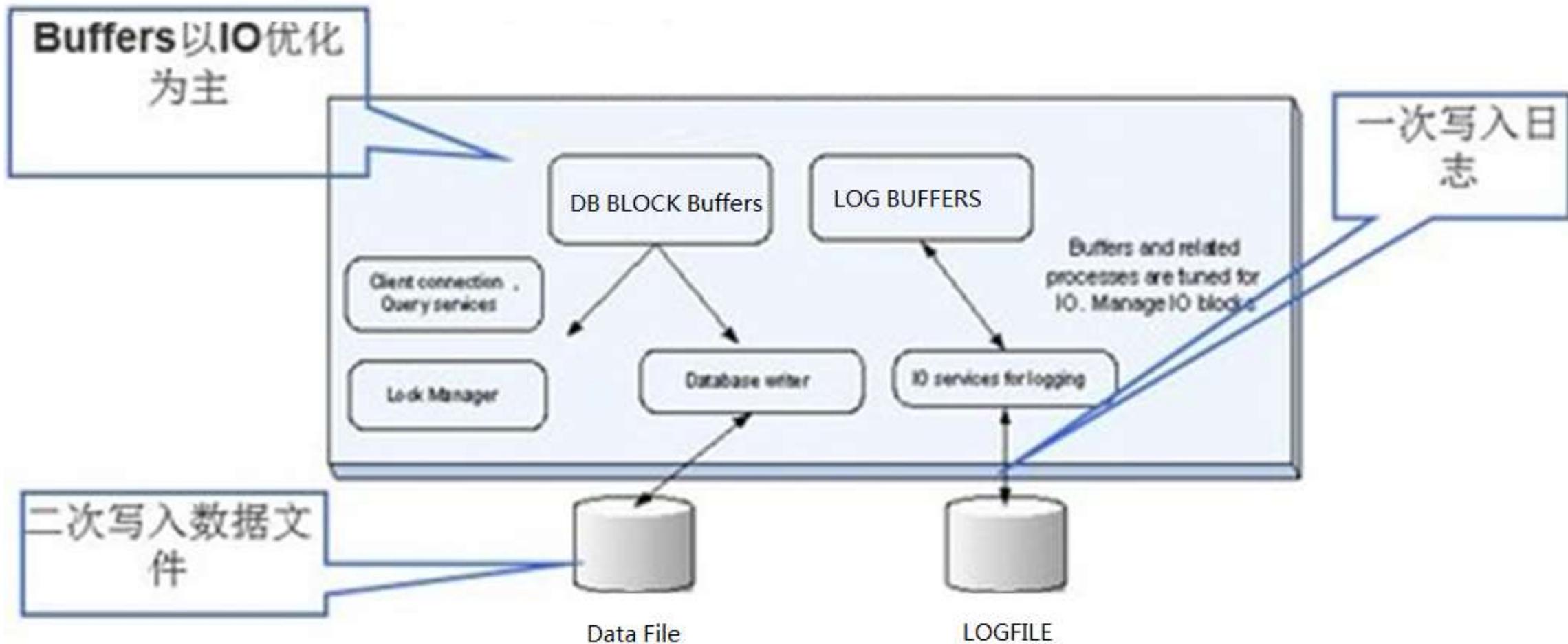
- ◆ 有限列
- ◆ 受限于字段类型、长度
- ◆ 空字段占用空间问题
- ◆ 全表扫描问题

传统RDBMS表关联



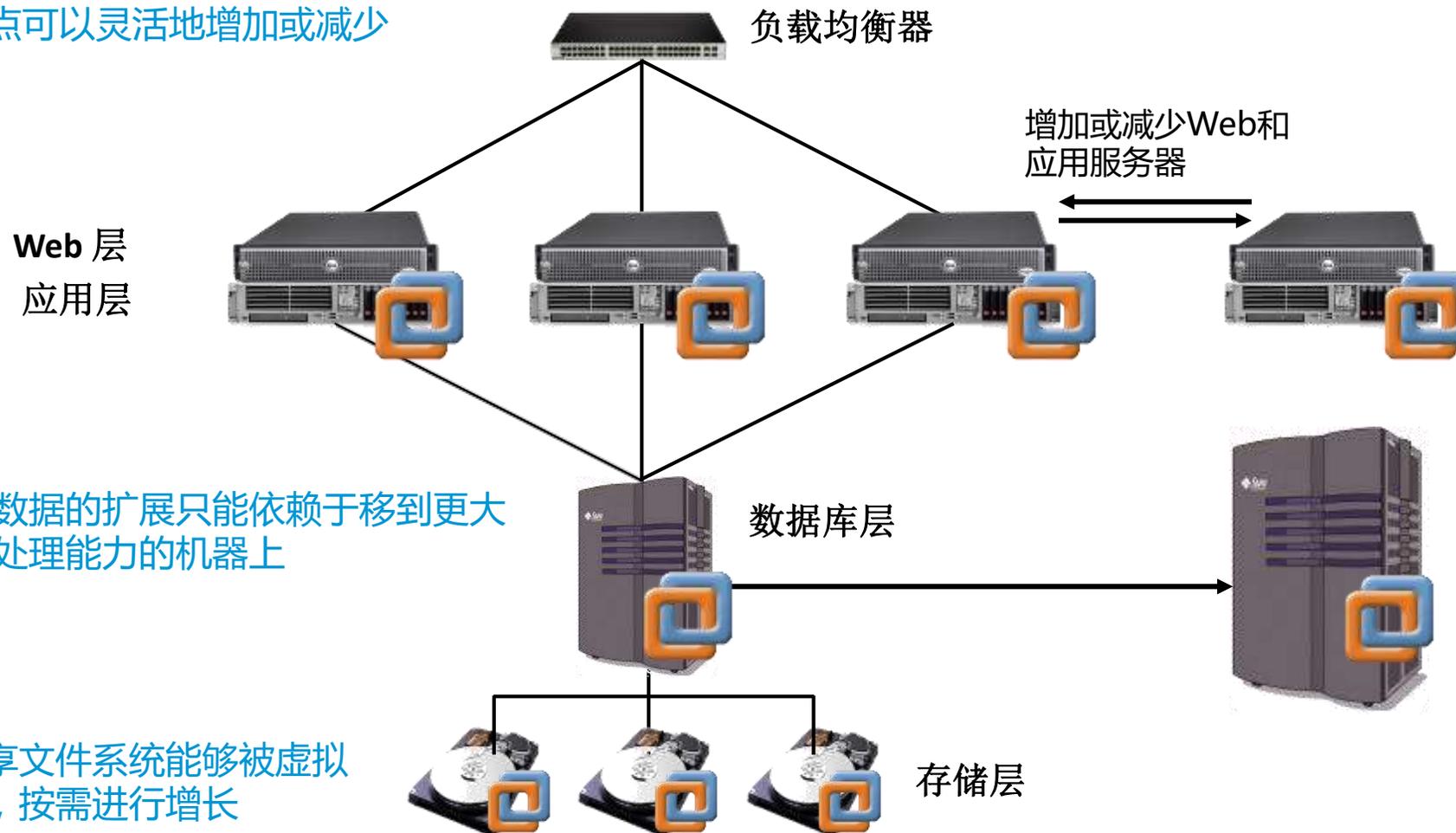
- ◆ 有限关联;
- ◆ 性能问题;
- ◆ 一系列限制;

传统关系型数据库IO面临的挑战



核心业务正在面临数据计算扩展能力的挑战

✓ Web和应用层很容易扩展及虚拟化，
节点可以灵活地增加或减少



✗ 数据的扩展只能依赖于移到更大
处理能力的机器上

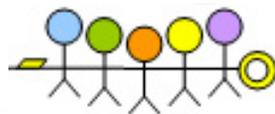
✓ 共享文件系统能够被虚拟
化，按需进行增长

内存数据库

- 内存数据库，顾名思义就是将数据放在内存中直接操作的数据库。相对于磁盘，内存的数据读写速度要高出几个数量级，将数据保存在内存中相比从磁盘上访问能够极大地提高应用的性能。
- 同时，内存数据库抛弃了磁盘数据管理的传统方式，基于全部数据都在内存中重新设计了体系结构，并且在数据缓存、快速算法、并行操作方面也进行了相应的改进，所以数据处理速度比传统数据库的数据处理速度要快很多。
(双通道DDR3-1333可以达到9300 MB/s，一般磁盘约150 MB/s)，
随机访问时间更是可以纳秒级（一般磁盘约10 ms，双通道DDR3-1333可以达到0.05 ms）。
- 内存数据库的最大特点是其“主拷贝”或“工作版本”常驻内存，即活动事务只与实时内存数据库的内存拷贝打交道。显然，它要求较大的内存量，但并非任何时刻整个数据库都存放在内存，即内存数据库系统还是要处理I/O

NoSQL概念

- NoSQL（NoSQL = Not Only SQL），指的是非关系型的数据库。
- 关系型数据库中的表都是存储一些格式化的数据结构，每个元组字段的组成都一样，即使不是每个元组都需要所有的字段，但数据库会为每个元组分配所有的字段，这样的结构可以便于表与表之间进行连接等操作，但从另一个角度来说它也是关系型数据库性能瓶颈的一个因素。
- 非关系型数据库以键值对存储，它的结构不固定，每一个元组可以有不一样的字段，每个元组可以根据需要增加一些自己的键值对，这样就不会局限于固定的结构，可以减少一些时间和空间的开销。



nosql运动背景

- ▶ web2.0的异军突起，传统的关系数据库为了保证“通用性”的设计而带来的功能复杂、性能开销大、价格昂贵的问题
- ▶ 避免不需要的复杂性
- ▶ 更高的吞吐量、高并发
- ▶ 在商用硬件上的水平扩展能力
- ▶ NOSQL实现了大表的自动分割（sharding）功能，更好的支持分布式处理
- ▶ 在性能和可靠性之间的折中
- ▶ 云计算的需求（从中心模式转到分布模式）

NoSQL特点

➤它们可以处理超大量的数据。

➤它们运行在便宜的PC服务器集群上。

PC集群扩充起来非常方便并且成本很低，避免了“sharding”操作的复杂性和成本。

➤它们击碎了性能瓶颈。

通过NoSQL架构可以省去将Web或Java应用和数据转换成SQL格式的时间，执行速度变得更快。

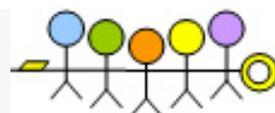
➤没有过多的操作。

关系数据库提供了无可比拟的功能集合，而且在数据完整性上也发挥绝对稳定，但对于某些企业的具体需求可能不需要那么多。

NoSQL产品

- Redis 内存数据库
- MongoDB 文档数据库
- Neo4j 图数据库
- Hbase 列数据库
- Hadoop 分布式计算模型
- GemFire 商用内存数据库
- timesTen Oracle商用内存数据库

各种nosql数据库介绍-redis



- ▶ 一个key-value存储系统,和Memcached类似
- ▶ 运行异常快
- ▶ 数据都是缓存在内存中,有硬盘存储支持的内存数据库
- ▶ Master-slave复制
- ▶ value数据类型丰富, string(字符串)、list(链表)、set(集合)和zset(有序集合)
- ▶ 支持push/pop, 允许用户实现消息机制

Redis与Memcached

- 共同特征
- 都是 key-value 形式的内存数据库
- 都是NoSQL家族的数据管理解决方案
- 都基于同样的key-value 数据模型
- 所有数据全部放在内存中(这也是适用于缓存的原因)
- 性能得分不分伯仲,包括数据吞吐量和延迟等指标
- 都是成熟的、广受开源项目欢迎的 key-value存储系统

Redis

- 有硬盘存储支持的内存数据库，
- 支持Master-slave复制
- 虽然采用简单数据或以键值索引的哈希表，但也支持复杂操作，例如 ZREVRANGEBYSCORE。
- INCR & co（适合计算极限值或统计数据）
- 支持 sets（同时也支持 union/diff/inter）
- 支持列表（同时也支持队列；阻塞式 pop操作）
- 支持哈希表（带有多个域的对象）
- 支持排序 sets（高得分表，适用于范围查询）
- Redis支持事务
- 支持将数据设置成过期数据（类似快速缓冲区设计）
- Pub/Sub允许用户实现消息机制

- **最佳应用场景：**适用于数据变化快且数据库大小可遇见（适合内存容量）的应用程序。
- **例如：** 股票价格、数据分析、实时数据搜集、实时通讯。

Redis

- 内存数据库
- 单核
- value限制512M
- 多种value类型, 游戏用途使用私有的序列化协议(例如protobuf)
- 支持落地(bgsave)
- 用户: 新浪, 淘宝, Flickr, Github
- 应用场景: 适合读写都很高, 数据处理复杂等

Redis速度

- SET操作每秒钟 110000 次，GET操作每秒钟 81000 次，
- 服务器配置如下：

Linux 2.6, Xeon X3320 2.5Ghz.

Redis 用武之地

- 使用Redis作为缓存,通过调优缓存内容,系统效率能获得极大提升。
- 很明显Redis的优势在于缓存管理。缓存通过某种数据回收机制(data eviction mechanism)在必要时将旧数据从内存中删除,为新数据腾出空间。Redis允许细粒度控制过期缓存,有6种不同的策略可供选择。Redis还采用了一些更复杂的内存管理方法和回收策略。
- Redis对缓存的对象提供更大的灵活性。是二进制安全的【即不丢数据,与编码无关】。Redis提供6种数据类型,使缓存以及管理缓存变得更加智能和方便。
- Redis 通过 Hash来存储一个对象的字段和值,并可以通过单个key来管理它们。
- 而使用Redis Hash的方式,可以大幅度降低资源消耗并提高性能。Redis的其他数据类型,如List 或者 Set,可用来实现更复杂的缓存管理模式。
- 160多个命令中的大部分都可以用来进行数据操作,所以通过服务端脚本调用进行数据处理成为现实。这些内置命令和用户脚本可以让你直接灵活地处理数据任务,而无需通过网络将数据传输给另一个系统进行处理。
- Redis提供了可选/可调整的数据持久化,目的是为了在崩溃/重启后可以快速加载缓存。
- 最后,Redis提供主从复制(replication)。Replication 可用于实现高可用的cache系统,允许某些服务器宕机的情况下也能提供不间断的服务。
- 需要使用缓存来提高应用系统性能时,Redis和Memcached是最佳的产品级解决方案。但考虑到其丰富的功能和先进的设计,绝大多数时候Redis都应该是你的第一选择。

Hbase特点

- 特点：支持数十亿行X上百万列
- 在 BigTable之后建模
- 采用分布式架构 Map/reduce
- 对实时查询进行优化
- 高性能 Thrift网关
- 通过在server端扫描及过滤实现对查询操作预判
- 支持 XML, Protobuf, 和binary的HTTP
- Cascading, hive, and pig source and sink modules
- 基于 Jruby (JIRB) 的shell
- 对配置改变和较小的升级都会重新回滚
- 不会出现单点故障
- 堪比MySQL的随机访问性能

Hbase存储特点

- 假如系统中有一个User表，如果按照传统的RDBMS的话，User表中的列是固定的，比如schema定义了name,age,sex等属性，User的属性是不能动态增加的。
- 但是如果采用列存储系统，比如Hbase，那么我们可以定义User表，然后定义info列族，User的数据可以分为：info:name = zhangsan,info:age=30,info:sex=male等，如果后来你又想增加另外的属性，这样很方便只需要info:newProperty就可以了。

MongoDB

- **DB-Engines**公布了**2014年年度最受欢迎数据库管理系统**
- 自从公布此榜单两年以来，MongoDB已经二度成为最受欢迎的NoSQL数据库。MongoDB最大的特点是它支持的查询语言非常大，其语法有点类似于面向对象的查询语言，几乎可以实现类似关系数据库单表查询的绝大部分功能，而且还支持对数据建立索引，这也是其如此受欢迎的原因之一
- 在开源的、面向文档的数据库中，MongoDB 经常被誉为具有RDBMS 功能的 NoSQL 数据库。
- MongoDB 还带有交互式 shell，这使得访问其数据存储变得简单，且其对于分块的即装即用的支持能够使高可伸缩性跨多个节点。

MongoDB

➤ i7四核 8G，insert 一次20万，循环10次，总共200万数据用时65秒。

➤ Insert语句及存储示例：

```
>db.person.insert({name:"test",age:40});
```

```
>db.person.insert({name:"tianmj",desc:{height:170,weight:80}});
```

```
> db.person.find();
```

```
{ "_id" : ObjectId("559388bf52a9ffe7970065bf"), "name" : "test", "age" : 40 }
```

```
{ "_id" : ObjectId("559388bf52a9ffe7970065c0"), "name" : "tianmj", "desc" : { "height" : 170, "weight" : 80 } }
```

```
>db.person.find({age:{$gt:30}});
```

```
"_id" : ObjectId("559388bf52a9ffe7970065bf"), "name" : "test", "age" : 40 }
```

```
> db.person.find({"desc.height":{$gt:130}});
```

```
{ "_id" : ObjectId("559388bf52a9ffe7970065c0"), "name" : "tianmj", "desc" : { "height" : 170, "weight" : 80 } }
```

MongoDB特点

- 特点：保留了SQL一些友好的特性（查询，索引）。
- Master/slave复制（支持自动错误恢复，使用 sets 复制）
- 内建分片机制
- 支持 javascript表达式查询
- 可在服务器端执行任意的 javascript函数
- update-in-place支持比CouchDB更好
- 在数据存储时采用内存到文件映射
- 对性能的关注超过对功能的要求
- 采用 GridFS存储大数据或元数据（不是真正的文件系统）

- **最佳应用场景：**适用于需要动态查询支持；需要使用索引而不是 map/reduce功能；需要对大数据库有性能要求；

MongoDB 与 Hbase

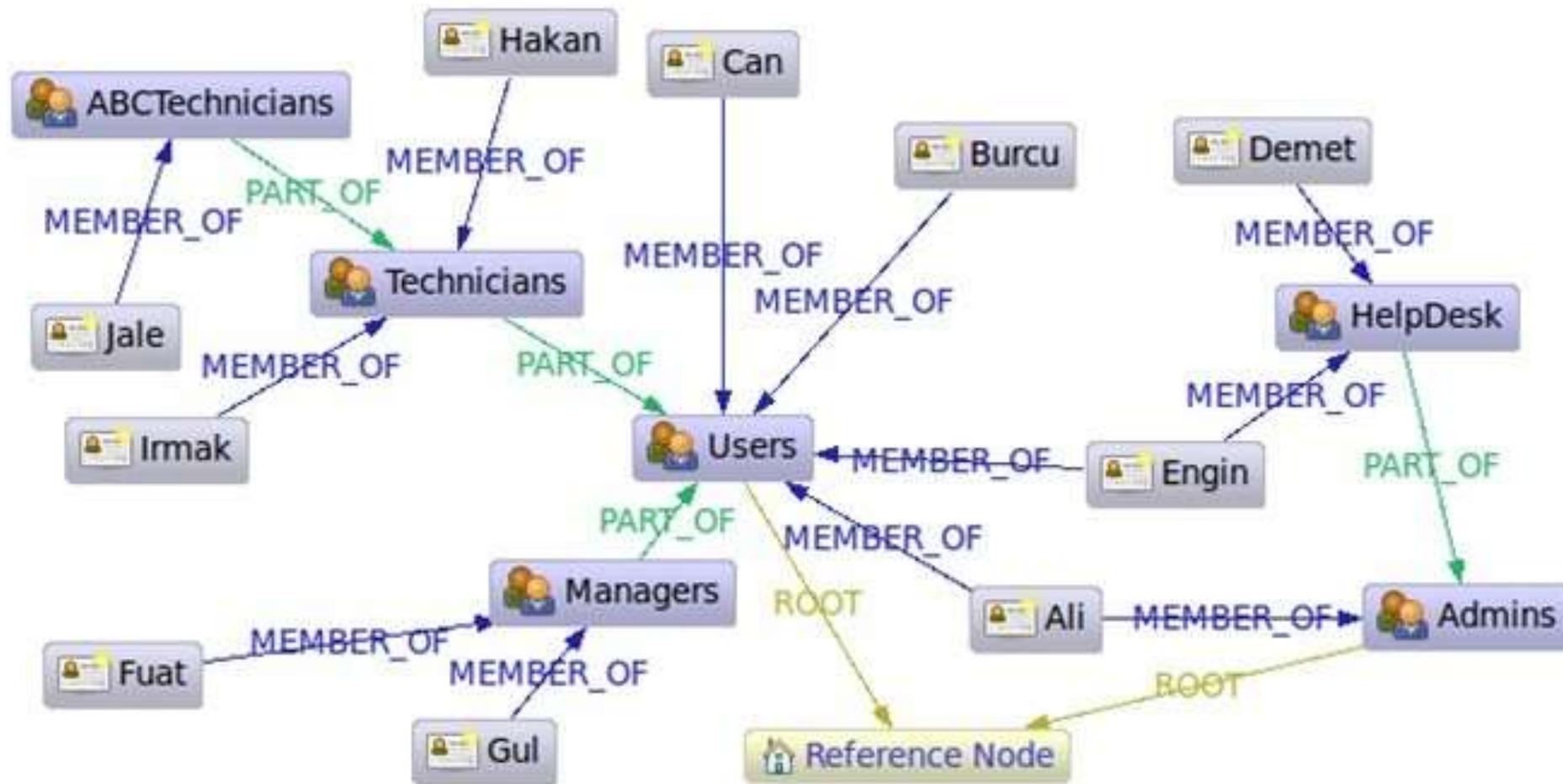
- 1.Mongodb bson文档型数据库，整个数据都存在磁盘中，hbase是列式数据库，集群部署时每个familycolumn保存在单独的hdfs文件中。
- 2.Mongodb 主键是“_id”，主键上面可以不建索引，记录插入的顺序和存放的顺序一样，hbase的主键就是row key，可以是任意字符串(最大长度是 64KB，实际应用中长度一般为 10-100bytes)，在hbase内部，row key保存为字节数组。存储时，数据按照Row key的字典序(byte order)排序存储。
- 3.Mongodb支持二级索引，而hbase本身不支持二级索引
- 4.Mongodb支持集合查找，正则查找，范围查找，支持skip和limit等等，是最像mysql的nosql数据库，而hbase只支持三种查找：通过单个row key访问，通过row key的range，全表扫描
- 5.mongodb的update是update-in-place，也就是原地更新，除非原地容纳不下更新后的数据记录。而hbase的修改和添加都是同一个命令：put，如果put传入的row key已经存在就更新原记录，实际上hbase内部也不是更新，它只是将这一份数据已不同的版本保存下来而已，hbase默认的保存版本的历史数量是3。
- 6.mongodb的delete会将该行的数据标示为已删除，将该删除记录数据置空，这种方法会提升性能。

MongoDB 与 Hbase

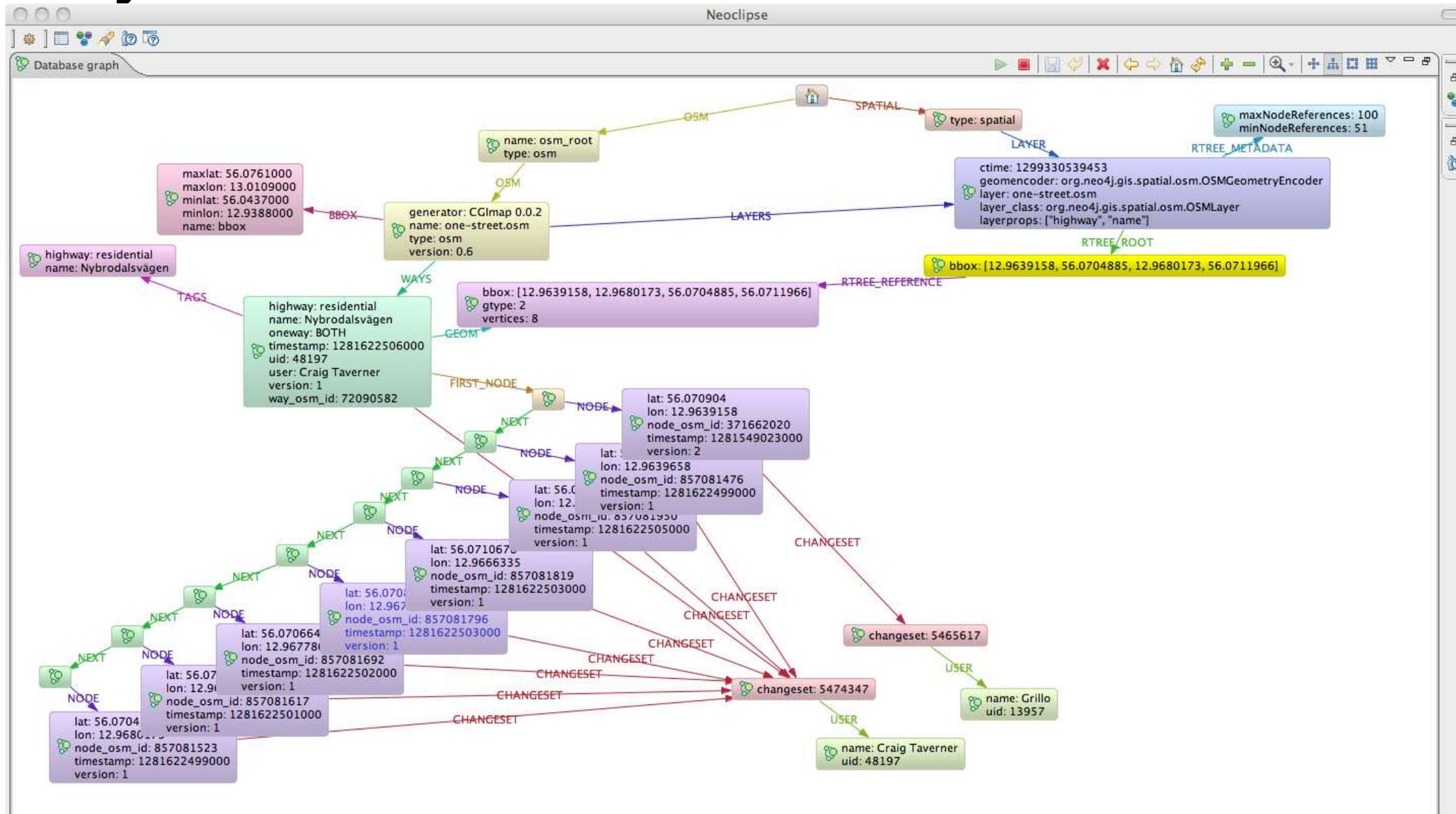
- 7.mongodb和hbase都支持mapreduce，不过mongodb的mapreduce支持不够强大，如果没有使用mongodb分片，mapreduce实际上不是并行执行的
- 8.mongodb支持shard分片，hbase根据row key自动负载均衡，这里shard key和row key的选取尽量用非递增的字段，尽量用分布均衡的字段，因为分片都是根据范围来选择对应的存取server的，如果用递增字段很容易热点server的产生，由于是根据key的范围来自动分片的，如果key分布不均衡就会导致有些key根本就没法切分，从而产生负载不均衡。
- 9.mongodb的读效率比写高，hbase默认适合写多读少的情况。
- 10.hbase采用的LSM思想(Log-Structured Merge-Tree)，就是将对数据的更改hold在内存中，达到指定的threadhold后将该批更改merge后批量写入到磁盘，这样将单个写变成了批量写，大大提高了写入速度，但降低了读的性能。

mongodb采用的是mapfile+Journal思想，如果记录不在内存，先加载到内存，然后在内存中更改后记录日志，然后隔一段时间批量的写入data文件，这样对内存的要求较高，至少需要容纳下热点数据和索引。

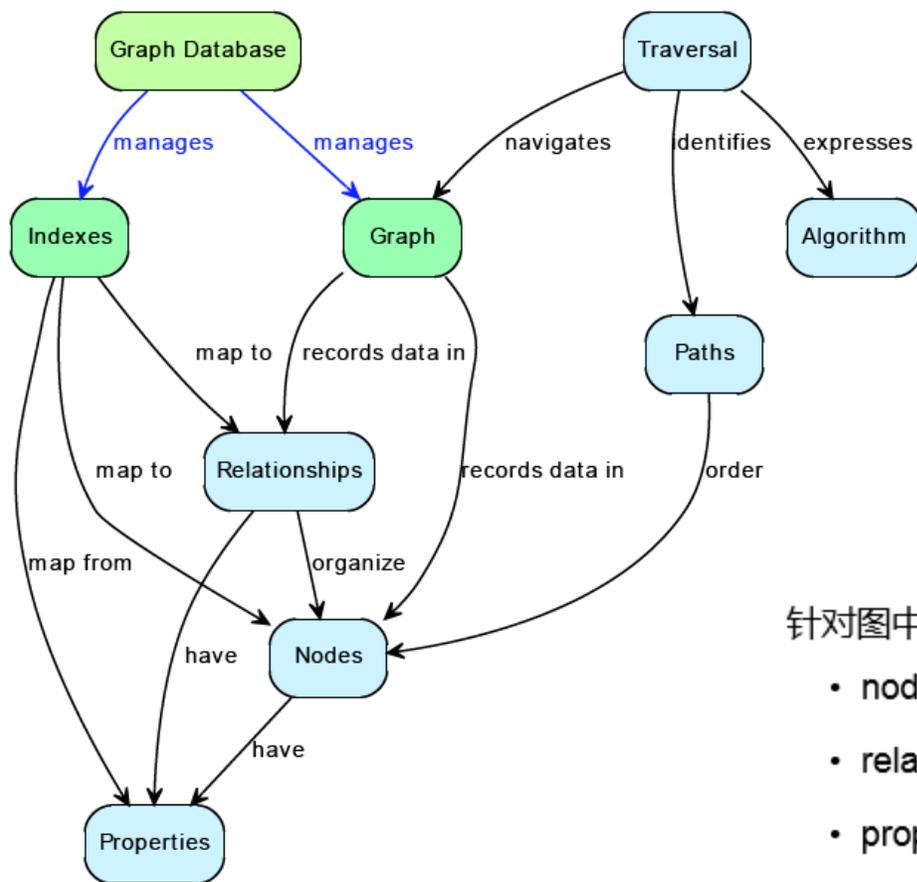
Neo4j 图数据库



Neo4j 图数据库



Neo4j 图数据库



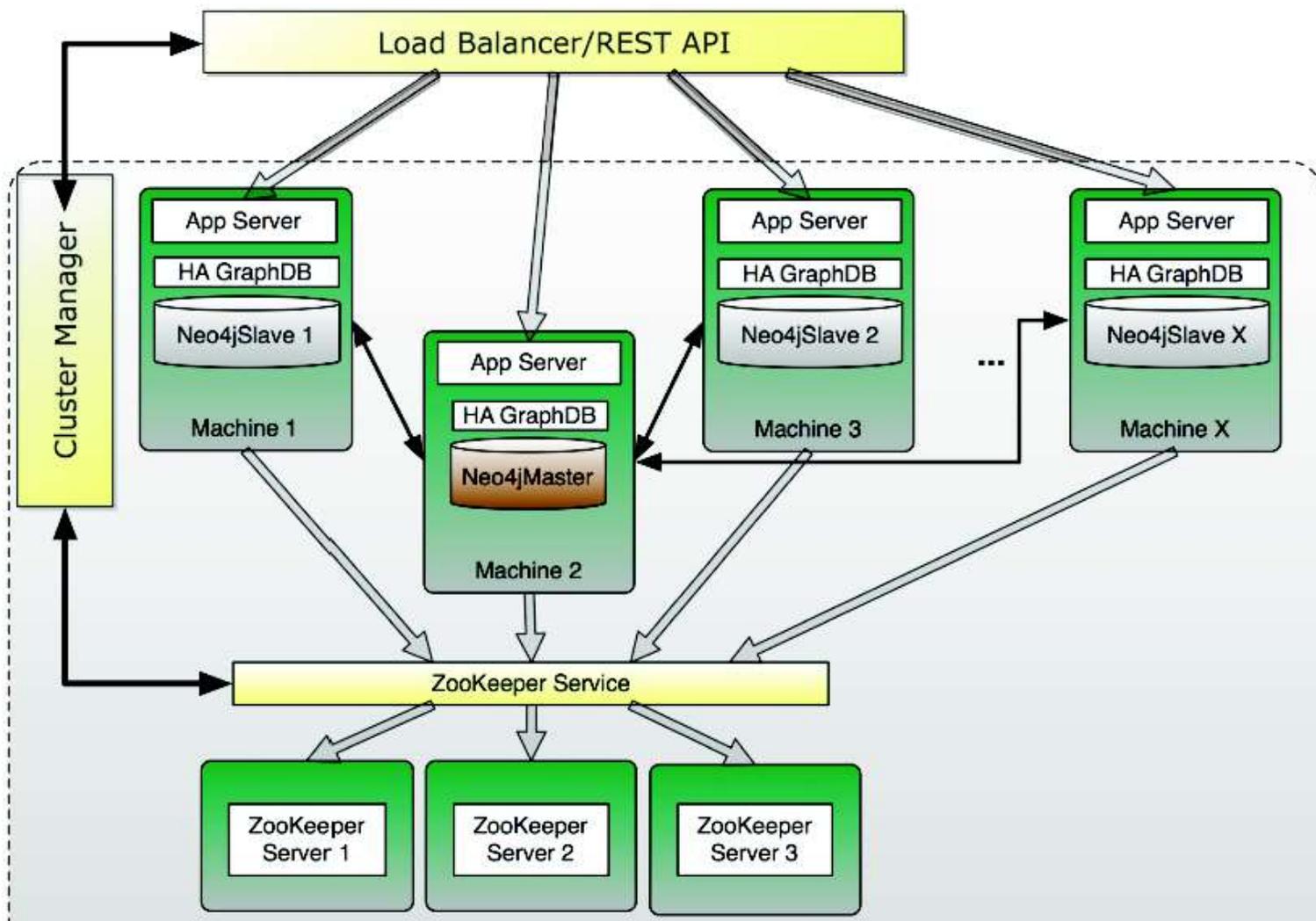
针对图中的一些基本概念：

- node：节点
- relationships：关系，也就是图中的边，注意是有向边
- properties：属性，针对node/relationship都可以设置property
- Traversal：图遍历工具
- Indexes：索引

Neo4j 图数据库

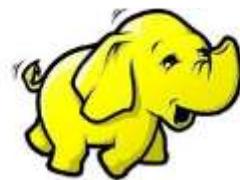
- **node(节点)**
 - 每个节点可以和多个节点之间建立多个关系(relationship)
 - 单个节点可以设置多个(Key,Value)的properties属性的键值对
- **relationships(关系)**
 - 每个关系都会包含一个startNode和endNode
 - 每个关系可以设置多个(Key,Value)的properties属性的键值对
 - 可以为关系定义对应的关系类型(RelationshipType)
 - * DynamicRelationshipType 动态关系类型
 - * XXXRelationshipType 静态关系类型(实现了RelationshipType接口)

Neo4j 图数据库



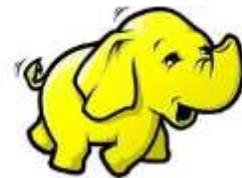
Neo4j特点

- 可独立使用或嵌入到 Java应用程序
- 图形的节点和边都可以带有元数据
- 很好的自带web管理功能
- 使用多种算法支持路径搜索
- 使用键值和关系进行索引
- 为读操作进行优化
- 支持事务（用 Java api）
- 使用 Gremlin图形遍历语言
- 支持 Groovy脚本
- 支持在线备份，高级监控及高可靠性支持使用 AGPL/商业许可
-
- **最佳应用场景：**适用于图形一类数据。这是 Neo4j与其他nosql数据库的最显著区别
- 例如：社会关系，公共交通网络，地图及网络拓谱

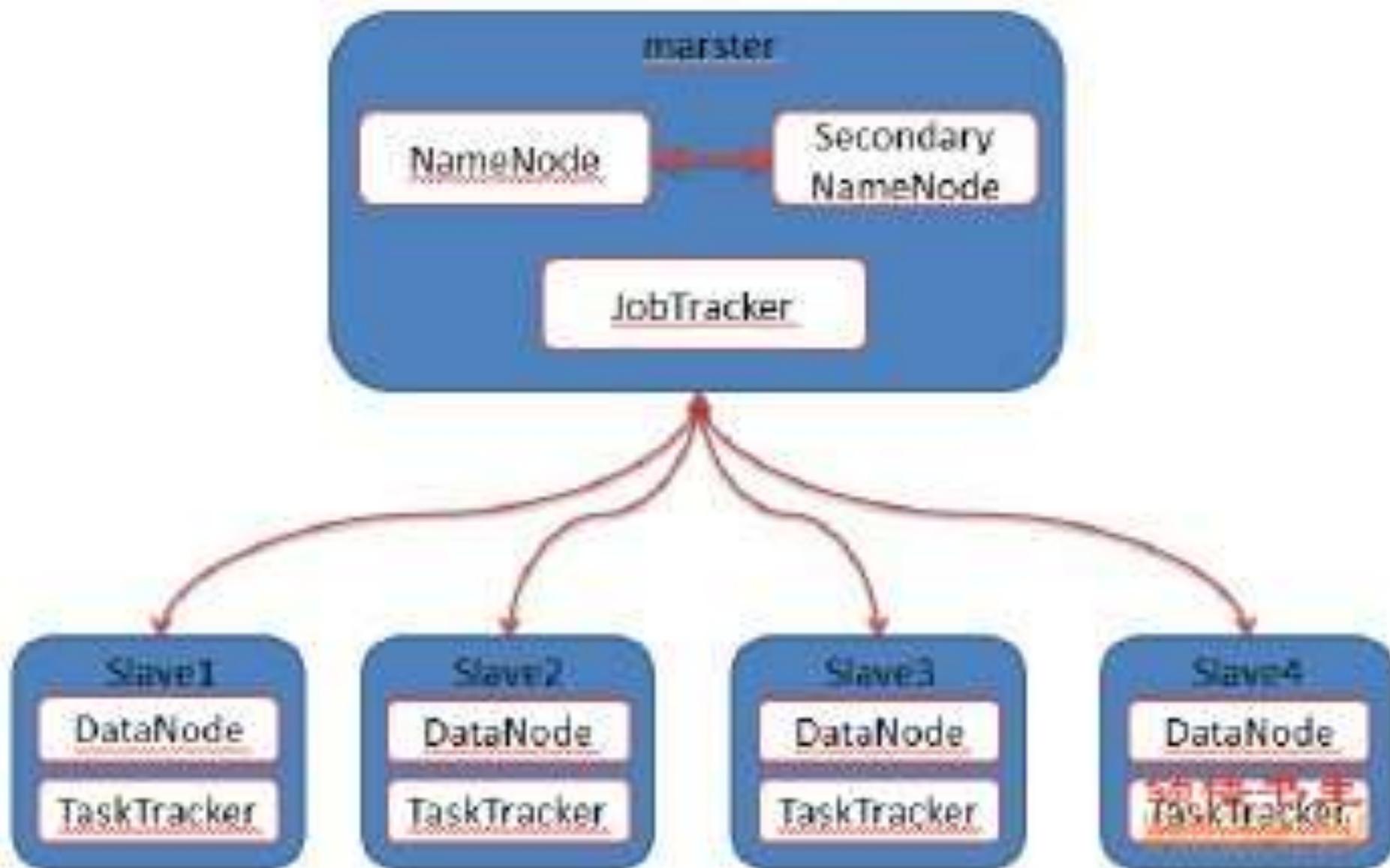


HADOOP





HADOOP



GemFire

- GemFire的第一个版本发布于2002年3月份，当时它还属于一家独立的公司GemStone Systems.后来GemStone System这家公司被VMware给收购了，GemFire也被整合到了VMware Vfabric产品线。请注意，VMWare当时也收购了Redis项目。
- 在2013年4月EMC与VMware/GE合资成立一家新公司Pivotal，VMware慷慨的贡献出了它的vfabric产品线，以及它收购的一些开源项目。
- 目前，GemFire的商业版权已经属于Pivotal了。顺便说一句，Redis的创始人Salvatore Sanfilippo 现在也供职于Pivotal.

GemFire有什么特点？

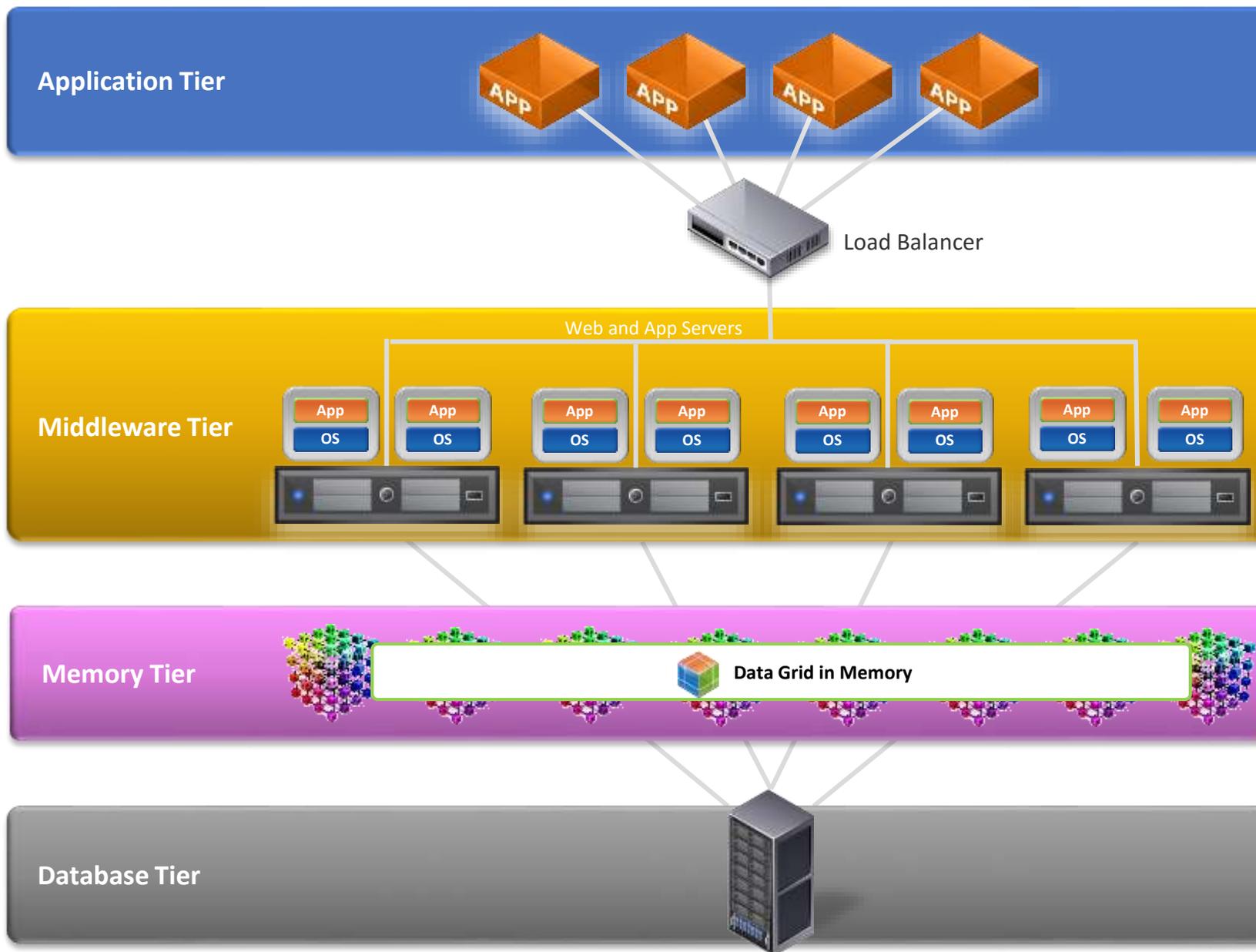
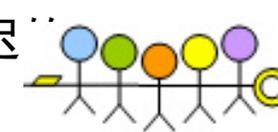
1. 分布式数据存储

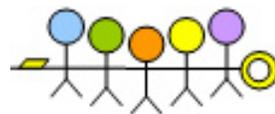
- 稳定而高性能的基于内存的数据数据存储
- 灵活的Cache部署策略：点对点（peer to peer）；客户端 / 服务端（client server）；多集群（multiple clusters）的本地或远程数据同步，支持数据高性能灾备和双活
- 灵活的Region（数据对象集或者可理解为表）分布式处理：同一集合数据（可理解为一个表的数据）可以整集多点同步或切割后不同点保存，并支持数据实时再平衡（rebalance）既数据分隔保存后若加入新的空闲服务器，数据可以在不重启服务的情况下重新切割和平衡数据，从而达到真正的数据在线动态延展
- 具有持续性的数据高可用性和容错性：各个分散的数据点可以配置一个或多个基于内存的热备数据点，当主数据点宕机的情况下，其中一个热备点就会提升称为主数据点，同时可以继续的空闲机器上创建备份点，从而达到数据的持续的可用性。同时数据可以通过配置同步或异步地持续化到本地硬盘，或者到指定的数据库或文件中。
- 数据地客户端缓存：客户端可以将最常用数据缓存一个备份与本地，进一步加快效能
- 在线数据备份
- 数据全内存和部分内存策略：通过配置可以将数据全部存入内存，或者通过将非频繁使用数据挤出策略（LRU）来将部分频繁适用数据保存于内存中达到成本效益最大化
- 内置资源优化器用以降低JAVA GC所带来的延迟，支持单个大容量Cache点（一般服务器可配置超过40GB内存的Java heap size）

GemFire有什么特点？

- 安全支持：基于用户和角色的数据访问，数据传输渠道加密（SSL）
- 数据存取除key-value简单cache支持外，支持复杂数据对象和关系存储
- 丰富的OQL（类SQL）的查询语言支持
- 支持数据单记录或批处理
- 本地或分布式事务处理
- Map-Reduce并行查询：同一查询命令可并行发送到各Cache点（Map），结果集自动在客户端汇合（Reduce）
- 智能定点查询：查询命令在包含数据特征如主键值时，查询命令会自动命中数据点

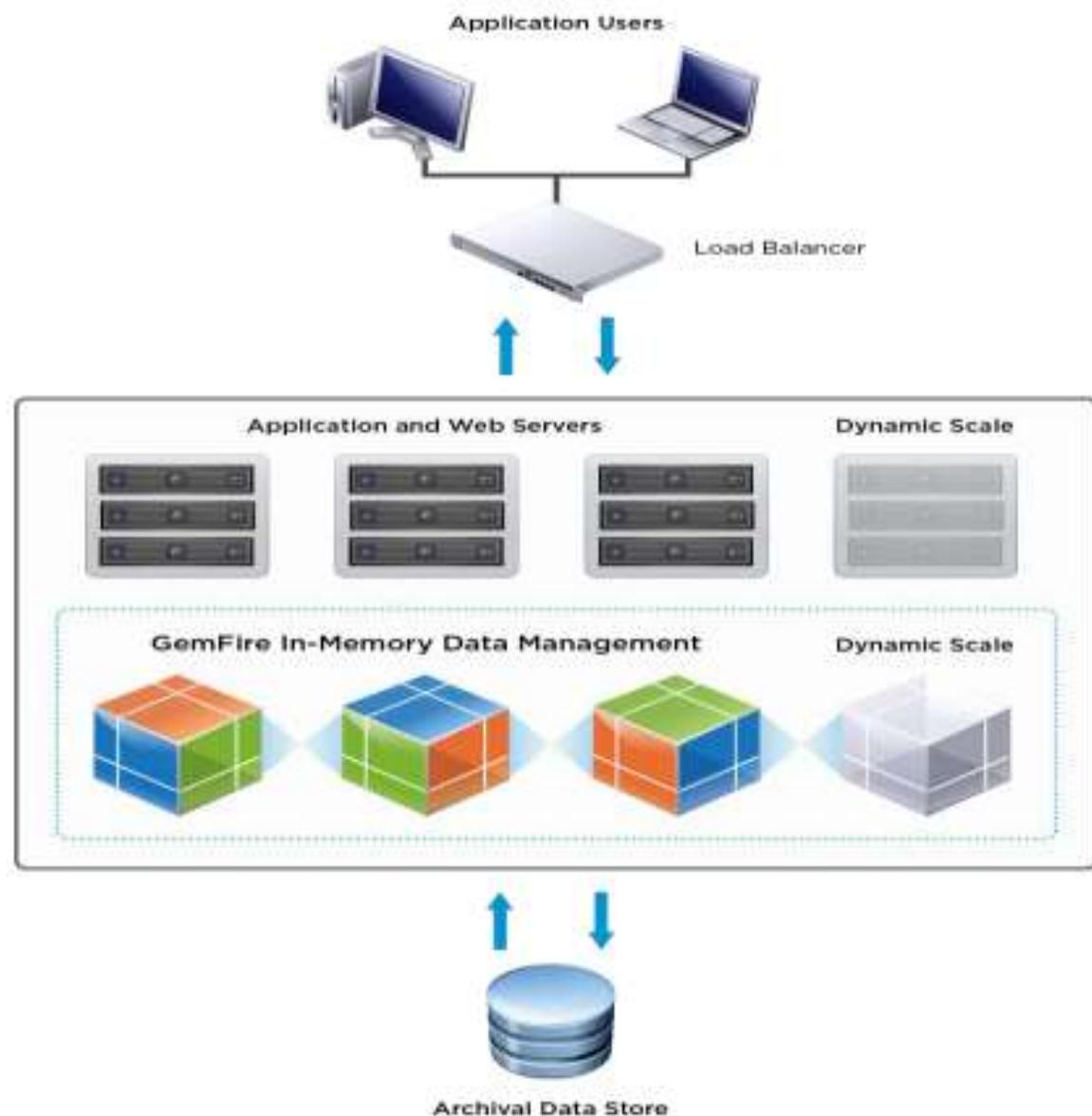
在NoSQL领域，利用弹性的数据网格计算架构，为业务系统提供高并发、低延迟数据处理能力，同时保障数据的最终一致性和完整性



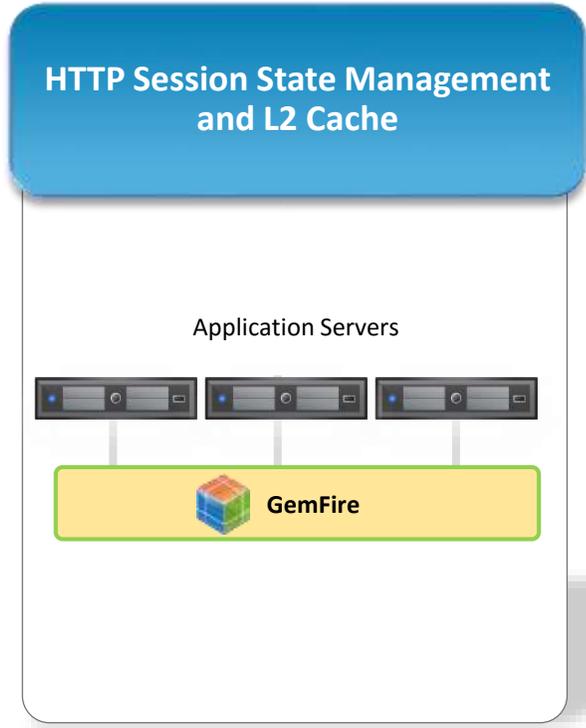


VMware GemFire是什么?

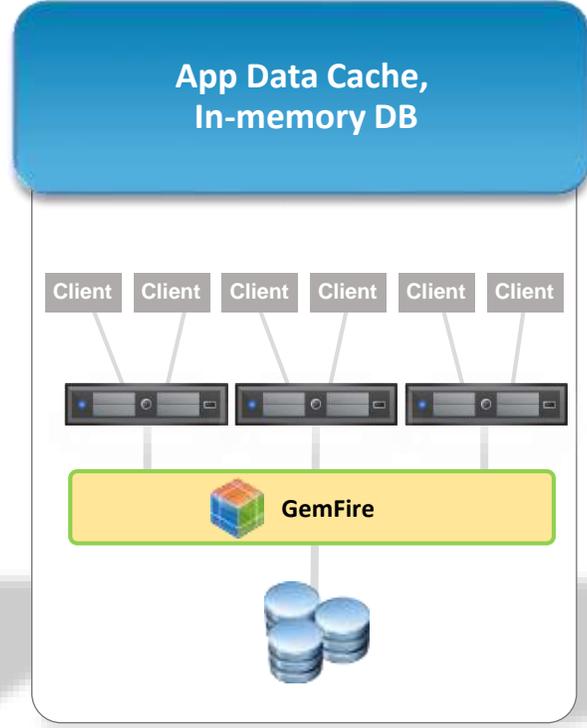
- 把数据移动中间层
 - 更靠近需要它的地方
- 伸缩性
 - 易于适应更多的应用用户
- 高性能
 - 所以计算操作都在内存, 极大提高性能
- 数据可靠性
 - 多个数据备份
 - 数据自动同步
 - 数据可以写入磁盘(数据库)直写或是后写
- 跨地域分布
 - 通过WAN连接



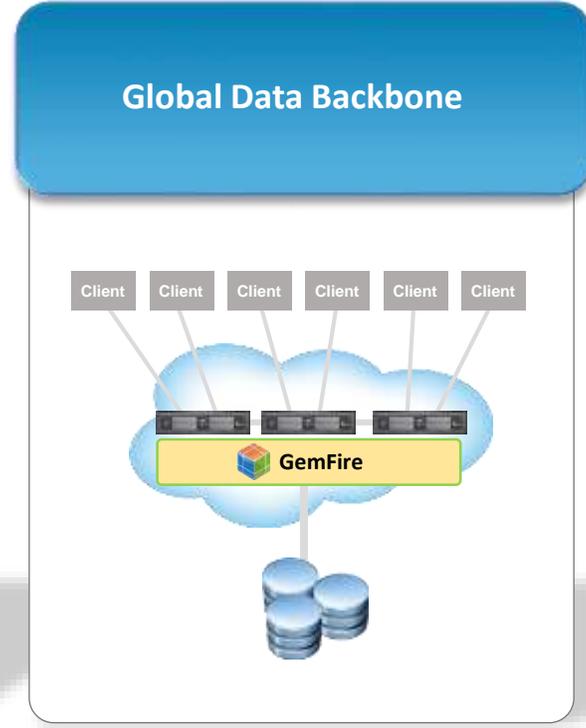
GemFire 典型的几种部署方式



Shopping cart state management

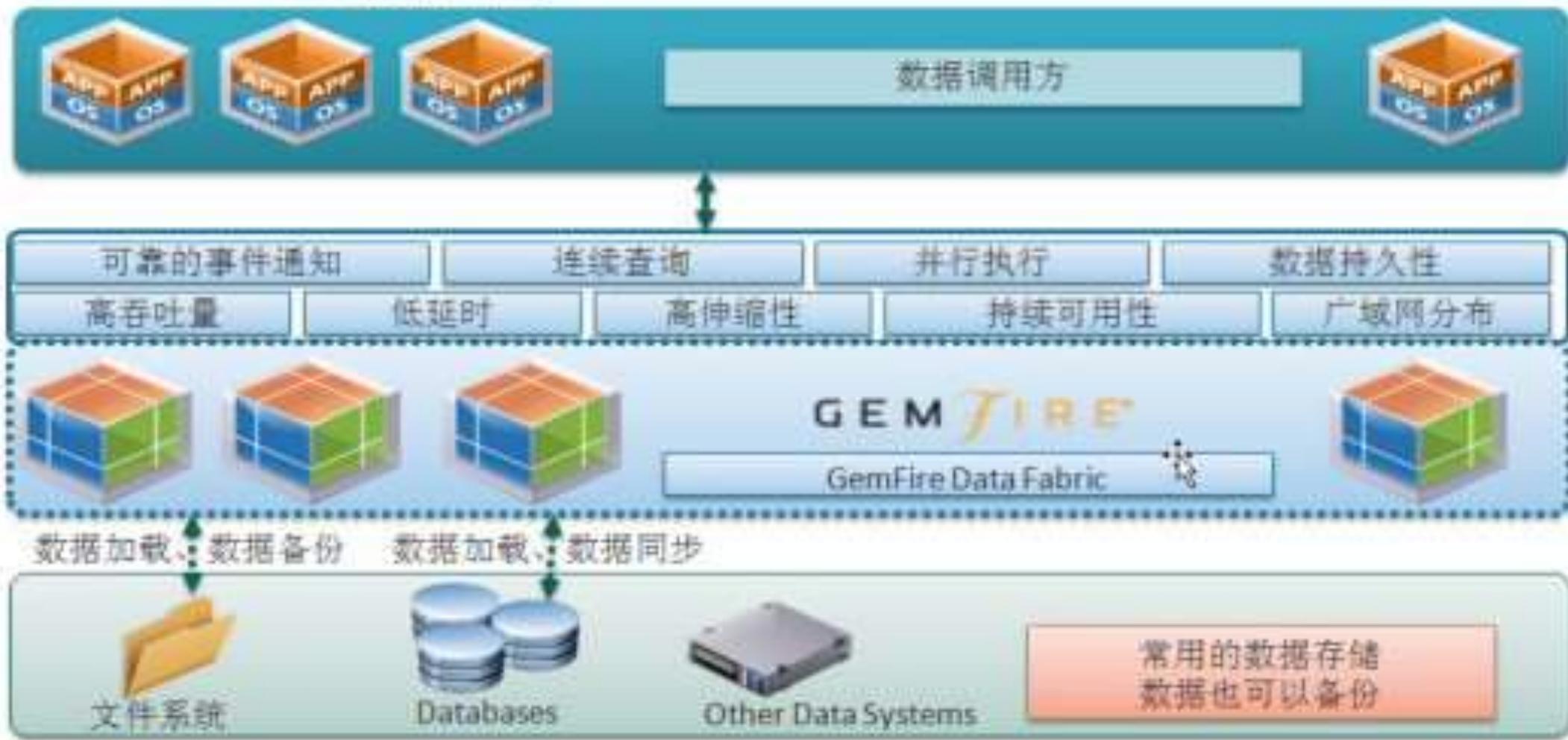


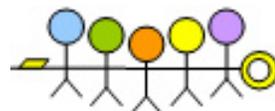
High performance OLTP



- Shared data grid populated by multiple data feeds and serving many clients
- Ability to execute app logic in clients and/or within the data grid itself

GemFire功能架构





Pivotal 内存计算解决方案



GemFire Enterprise

关键值随 OQL 查询存储

存储对象 (Java、C++、C#、.NET) 或非结构化数据

允许来自多种语言的并发客户端同时访问

2 级缓存用于 Hibernate

HTTP 会话状态管理

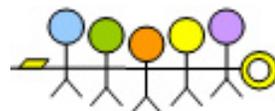
GemFire SQLFire

使用 SQL 界面存储关系数据

支持 JDBC、ODBC、Java 和 .NET 界面

使用现有的关系工具

Item Name	Thread price (unit)	Weight kilograms	Volume cubic meters	Lead time (days)	Price per 100 units	Available at factory stock?	Number in stock	Fill in policy (unit)
M1	0.7	4g	4	Fast	\$1008	Yes	235	Full
M2	0.3	4g	3	Normal	\$11.86	Yes	182	Both
M3	1	3g	4	Subst.	\$50.02	Yes	140	Full
M4	1.25	5g	4	Fast	\$11.98	No	298	Partial
M5	1.5	5g	10	Normal	\$16.75	Yes	400	Partial
M6	1.75	7g	12	Fast	\$18.28	No	388	Full
M7	2	7g	14	Normal	\$21.12	No	222	Partial
M8	2	4g	14	Subst.	\$50.02	Yes	200	Both



采用 GemFire 的场景和切入点

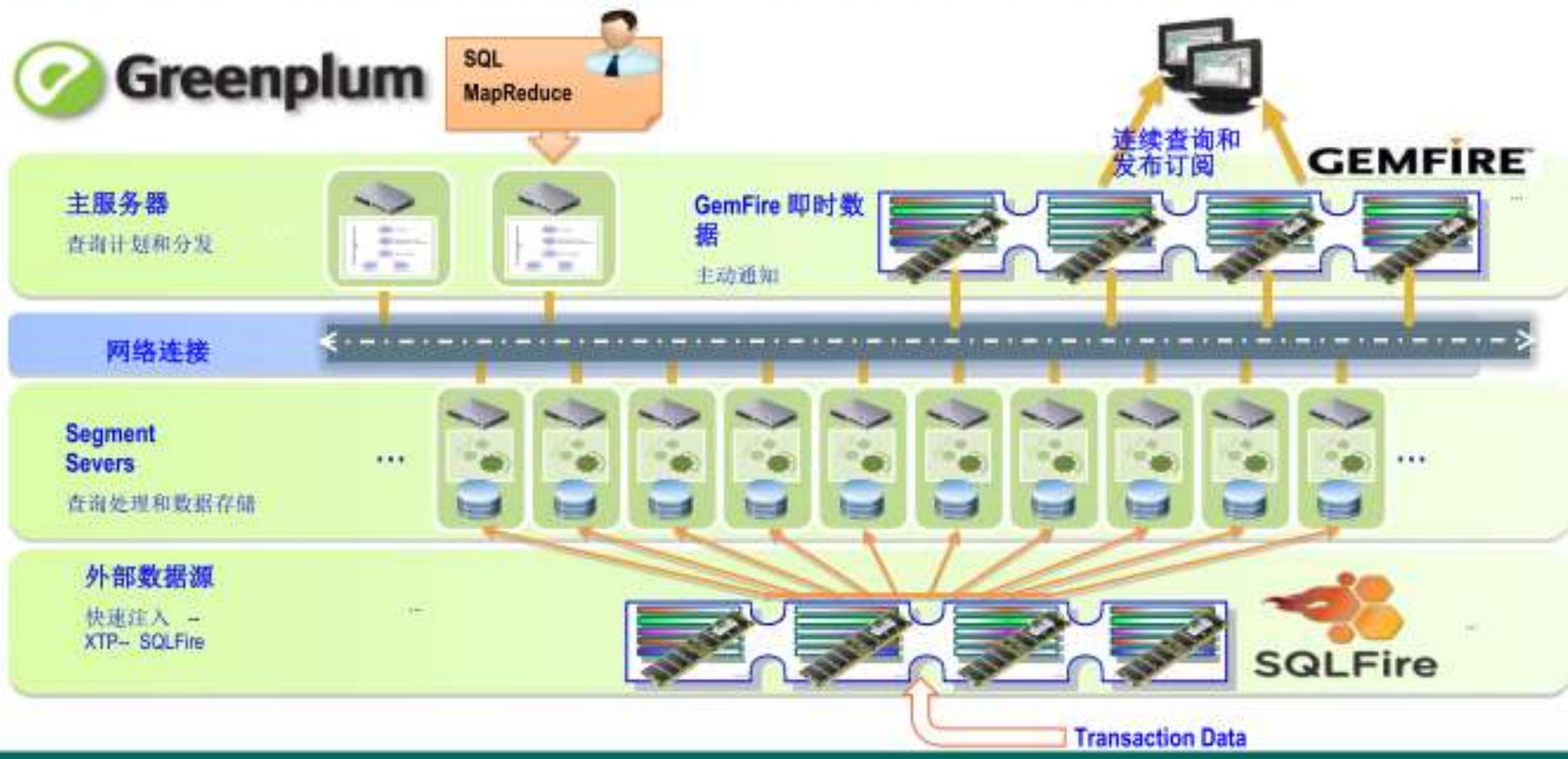
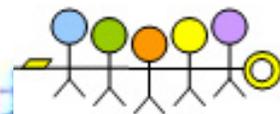
应用场景

- ▶ 解决应用性能问题
 - ▶ OLAP数据处理不够快
 - ▶ 报表生成
 - ▶ ETL
 - ▶ CEP
 - ▶ OLTP业务处理不够快
 - ▶ 网上银行
 - ▶ 录入系统
 - ▶ CRM
 - ▶ 网上业务
 - ▶ 综合支付平台
- ▶ 数据总线
 - ▶ 花旗银行
- ▶ 双活数据中心(全局数据分布)
 - ▶ 上海教委

解决方案切入点

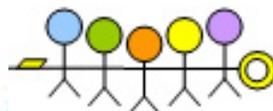
- ▶ PaaS平台
- ▶ 快应用
- ▶ 快数据
- ▶ 数据总线
- ▶ 大数据Installbase需要快数据

Greenplum和Gem/SQLFire的集成=大数据+快数据



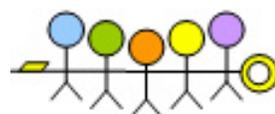
使用场景

- 商业银行信用卡用户行为实时分析及附加服务的提供
- 商业银行信用卡反欺诈实时分析和监控
- 商业银行客户关系管理系统报表查询及分析
- 商业银行主机系统日间/夜间批量对账
- 投资银行外汇交易
- 投资银行金融衍生品风险评估和定价
- 大型在线订票网站火车票余额查询
- 大型航空信息提供商机票信息查询和产品定价
- 交通运输行业实时道路车辆监控
- 娱乐网站实时在线广告推送
- 等等



典型海量网上并发业务—12306采用GemFire改造

- 2012年春节，12306网上订票上线，高峰时间无法登陆网站，登陆了网站也无法订票
- 2012年3月开始，铁路总公司(原铁道部)开始调研、改造12306
- 2012年6月选择GemFire改造12306，一期是改造余票查询，9月份完成代码改造，系统上线
- 2012年国庆，又是网上订票高峰期间，大家可以显著发现，可以登录12306,虽然还是很难订票，但是查询余票很快。
- 2012年10月份，二期用GemFire改造订票查询
- 2013年春节，又是网上订票高峰期间，大家可以显著发现，可以登录12306,虽然还是很难订票，但是查询余票很快，而且查询自己的订票记录也很快。



网站12306改造--从余票查询开始

- 1、要订票，你先要查有没有余票可以订
- 2、余票查询对整体改造小、对整体影响不大，但是见效快，所以第一个选择余票查询模块用GemFire改造

余票查询

日期: 2012年 03月 03日

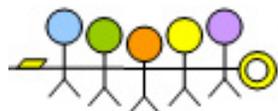
出发: 广州 到达: 北京 车次: G134

票种: 成人 儿童 学生 团体

票额: 成人 儿童 学生 团体

序号	车次	查询区间		区间运行时刻			余票情况										币种
		发站	到站	发时	到站	到站	硬座	特等座	一等座	二等座	商务座	软卧	硬卧	软卧	硬卧	硬座	
18	T130(广州-北京西)	广州	北京西	17:42	13:28	09:34	-	-	-	-	-	0	0	-	0	0	
20	G134(广州-北京西)	广州	北京西	08:27	12:46	05:45	-	-	0	20	-	-	-	-	-	-	
21	T140(广州-北京北)	广州	北京北	08:28	14:12	06:44	-	-	-	-	-	0	0	-	0	0	
22	T198(九龙-北京西)	广州	北京西	09:21	14:52	05:31	-	-	-	-	-	2	-	-	3	0	
24	D138(广州-北京西)	广州	北京西	08:48	08:38	05:47	-	-	24	40	-	-	-	-	-	-	
26	K440(广州-北京西)	广州	北京西	11:25	20:00	08:50	-	-	-	-	-	0	0	-	0	0	
28	D138(汉口-北京西)	广州	北京西	10:45	08:18	05:38	-	-	0	0	-	-	-	-	-	-	
29	K968(张家界-北京西)	广州	北京西	10:55	21:25	08:42	-	-	-	-	-	2	75	-	1	0	
27	K920(重庆北-北京西)	广州	北京西	10:24	22:10	08:36	-	-	-	-	-	0	0	-	0	0	
28	D134(汉口-北京西)	广州	北京西	12:42	08:18	05:27	-	-	0	1	-	-	-	-	-	-	
29	K918(成都-北京西)	广州	北京西	14:55	22:02	08:42	-	-	-	-	-	0	0	-	0	0	
30	K714(怀化-北京西)	广州	北京西	18:11	21:28	08:12	-	-	-	-	-	0	10	-	0	0	





改造后取得的效果

--来自网上订票系统实际运行数据

12306改造之前

- 单次查询耗时15秒左右
- 无法支持高流量并发查询，只能通过分库来实现，在极端高流量并发情况，系统无法支撑
- 高峰期无法访问，也无法动态增加机器来应当
- 运行在 UNIX小型机

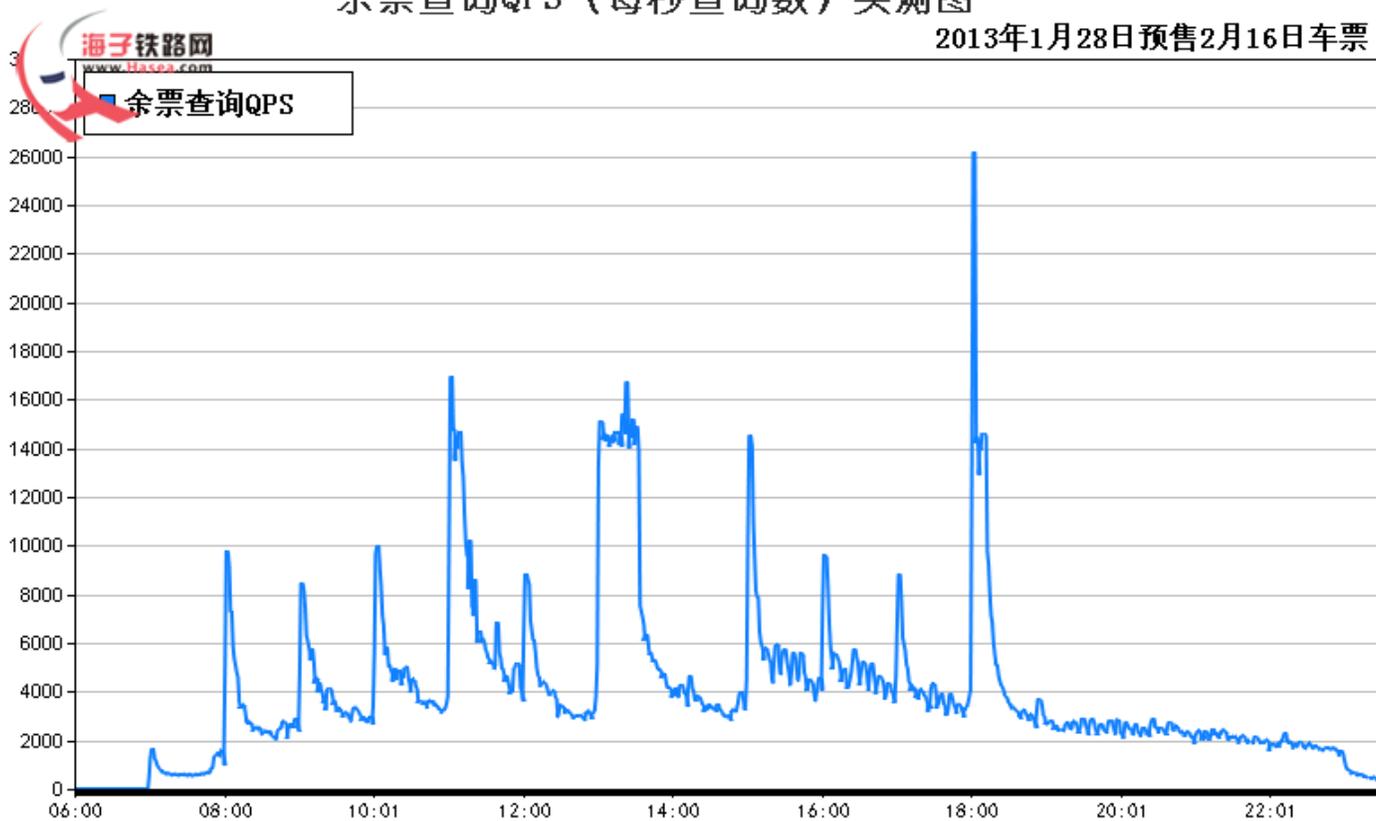
12306改造后

- 单次订票查询最长耗时150-200毫秒，单次查询最短耗时1-2毫秒。提高100倍-1000
- 支持每秒上万次的并发查询，高峰期间2.6个并发/秒，查询速度依然是平均10毫秒左右
- 按需弹性动态扩展，并发量增加还可以动态增加机器应对，同步实时变化的数据耗时秒级
- 运行在Linux X86服务器集群

12306

余票查询QPS（每秒查询数）实测图

2013年1月28日预售2月16日车票



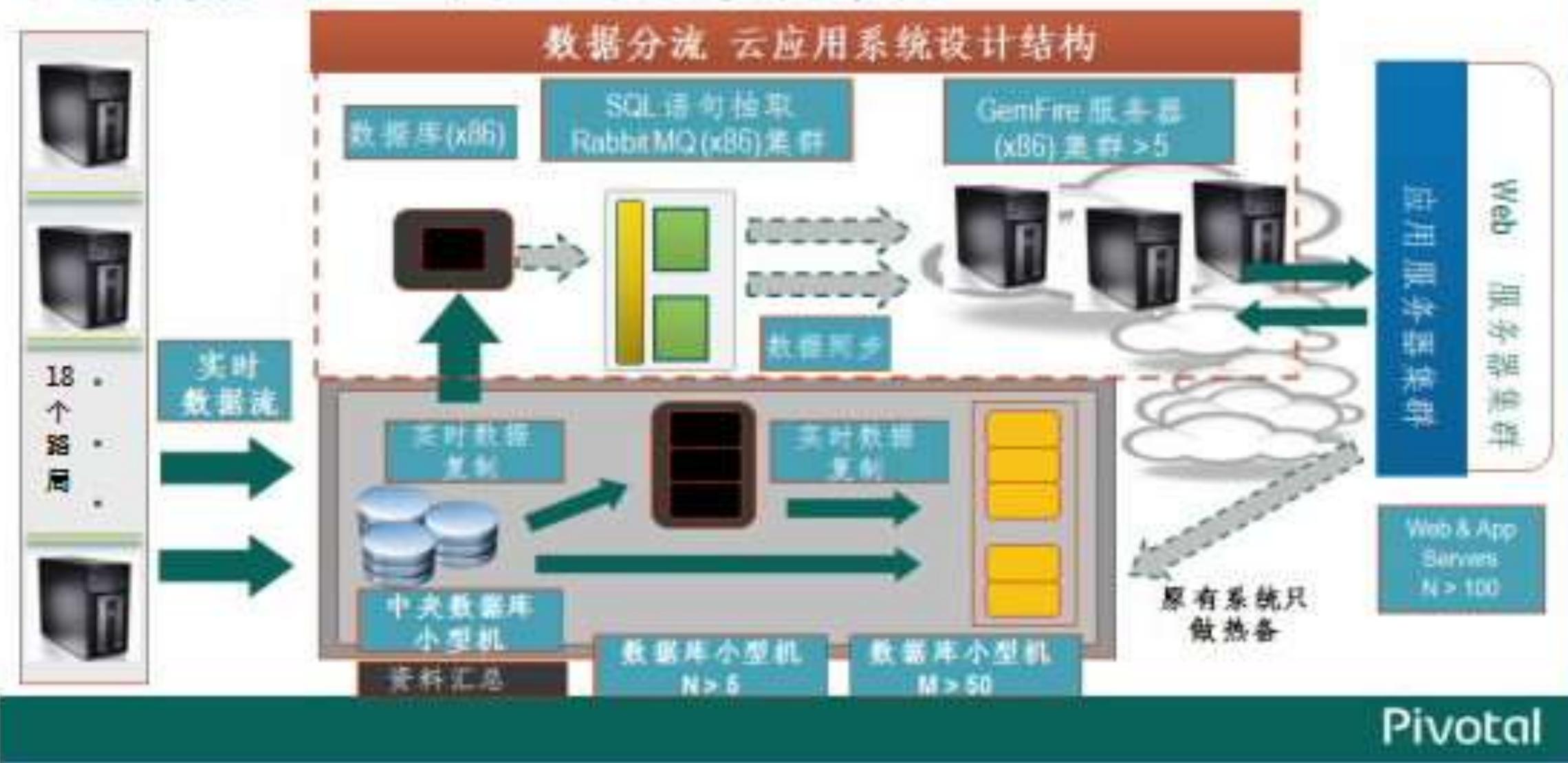
12306

- 12306从2012年3月开始改造，采用10几台X86服务器实现了以前数十台小型机的余票计算和查询能力，单次查询的最长时间从之前的15秒左右下降到0.2秒以下，缩短了75倍以上。2012年春运的极端高流量并发情况下，系统几近瘫痪。而在改造之后，支持每秒上万次的并发查询，高峰期间达到2.6万个查询/秒吞吐量

	尖峰日 PV 值	放票次数	网络带宽	尖峰日售票	同时在线人数限制	处理订单 (张/秒)
2012	10亿	4次	1.5G	110万	1万	200
2013	15亿	10次	3G	265万	20万	450
2014	144亿	16次	5G	501万		1000
2015	297亿	21次	12G	564万		1032

pv就是访问者打开了你的几个页面

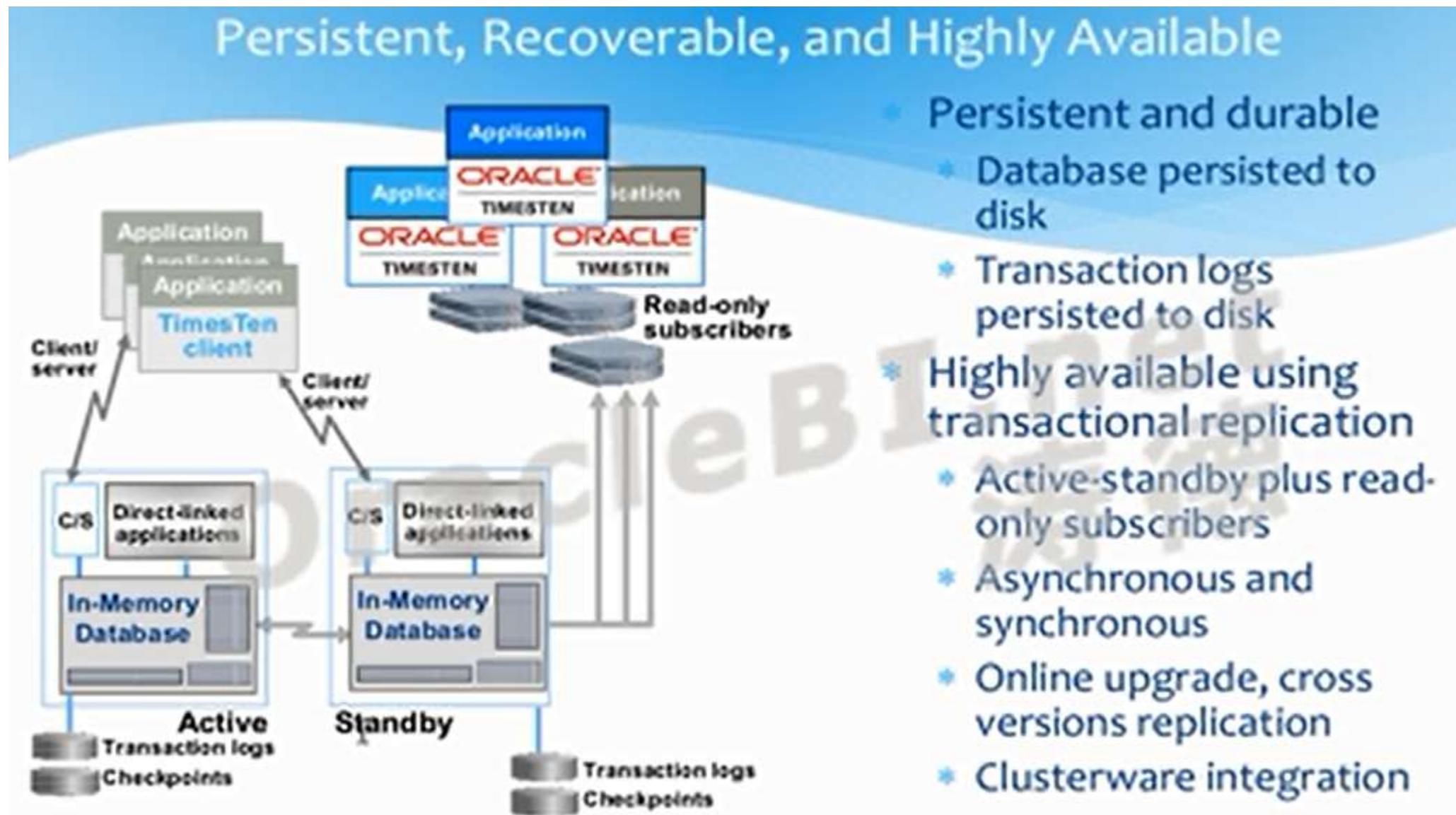
改造后的12306网上订票系统架构



12306

- 高流量方面，主要涉及低延迟的余票查询。Gemfire支持类Map-Reduce并行处理，能够按车次将余票、共用定义等数据拆分成多个独立的计算单元,对余票查询中最耗时的共用定义部分做预先处理,生成查询缓存。当余票数据发生变化时,系统会动态更新查询缓存。有了预处理及数据同步过程维护的动态查询缓存,单次查询可以控制在10ms-300ms之间,同时10分钟的固有延迟也不存在了。
- 订单查询系统改造，在改造之前的系统运行模式下，每秒只能支持300-400个QPS的吞吐量，高流量的并发查询只能通过分库来实现。改造之后，可以实现高达上万个QPS的吞吐量，而且查询速度可以保障在20毫秒左右。新的技术架构可以按需弹性动态扩展，并发量增加时，还可以通过动态增加X86服务器来应对，保持毫秒级的响应时间。
- 在以往的春运期间，12306售票系统部署Gemfire集群在2个[数据中心](#)，提供服务。
- 在2015年春运购票高峰之前，考虑到超大并发会造成网络流量大以及阻塞的问题，特别在阿里云建立一个[数据中心](#)，由阿里云提供“虚拟机”的租赁服务，将基于Gemfire实现余票查询功能的系统以及Web服务部署在这些虚拟机上，以分流“余票查询”请求，解决因为高峰期超高并发造成的网络阻塞问题，以进一步提高服务品质。为此，12306在2014年下半年在阿里云做了小规模部署和调试。2015年春运购票高峰期的12306高效平稳运行，也验证了混合架构的可行性。

Oracle TimesTen

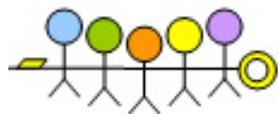


ArangoDB

- **1. 多数据模型:**
 - 可以灵活的使用Key-Value,document,graph或者他们的组合作为你的数据模型.
- **2.方便的查询:**
 - 支持类似SQL的查询语法AQL.或者通过REST以及其他查询。
- **3.Ruby和JS扩展:**
 - 没有语言范围限制,你可以从前台到后台都使用同一种语言。
- **4.高性能以及低空间占用:**
 - ArangoDB自测比其他nosql都要快,同时占用的空间更小
- **5. 简单易用:**
 - 可以在几秒内启动并且使用,同时可以通过图形界面来管理你的ArangoDB
- **6. 开源且免费:**
 - ArangoDB遵守Apache协议

Nosql数据库总结

Name	Type	Data storage options	Query types	Additional features
Redis	In-memory non-relational database	Strings, lists, sets, hashes, sorted sets	Commands for each data type for common access patterns, with bulk operations, and partial transaction support	Publish/Subscribe, master/slave replication, disk persistence, scripting (stored procedures)
memcached	In-memory key-value cache	Mapping of keys to values	Commands for create, read, update, delete, and a few others	Multithreaded server for additional performance
MySQL	Relational database	Databases of tables of rows, views over tables, spatial and third-party extensions	SELECT, INSERT, UPDATE, DELETE, functions, stored procedures	ACID compliant (with InnoDB), master/slave and master/master replication
PostgreSQL	Relational database	Databases of tables of rows, views over tables, spatial and third-party extensions, customizable types	SELECT, INSERT, UPDATE, DELETE, built-in functions, custom stored procedures	ACID compliant, master/slave replication, multi-master replication (third party)
MongoDB	On-disk non-relational document store	Databases of tables of schema-less BSON documents	Commands for create, read, update, delete, conditional queries, and more	Supports map-reduce operations, master/slave replication, sharding, spatial indexes



各种Nosql概览

列存储	Hbase Cassandra Hypertable	顾名思义，是按列存储数据的。最大的特点是方便存储结构化和半结构化数据，方便做数据压缩，对针对某一系列或者某几列的查询有非常大的 IO 优势。
文档存储	MongoDB CouchDB	文档存储一般用类似 json 的格式存储，存储的内容是文档型的。这样也就有机会对某些字段建立索引，实现关系数据库的某些功能。
key-value 存储	Tokyo Cabinet / Tyrant Berkeley DB MemcacheDB Redis	可以通过 key 快速查询到其 value。一般来说，存储不管 value 的格式，照单全收。（Redis 包含了其他功能）

各种Nosql概览

图存储	Neo4J FlockDB	图形关系的最佳存储。使用传统关系数据库来解决的话性能低下，而且设计使用不方便。
对象存储	db4o Versant	通过类似面向对象语言的语法操作数据库，通过对象的方式存取数据。
xml 数据库	Berkeley DB XML BaseX	高效的存储 XML 数据，并支持 XML 的内部查询语法，比如 XQuery, Xpath。