第十三章 NoSQL数据库

1 NoSQL简介

2 NoSQL的技术特点

NoSQL简介

概念演变 →

Not only SQL

最初表示“反SQL”运动
用新型的非关系数据库取代关系数据库

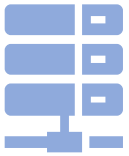
现在表示关系和非关系型数据库各有优缺点
彼此都无法互相取代

NoSQL数据库具有以下几个特点：

（1）灵活的可扩展性

（2）灵活的数据模型

（3）与云计算紧密融合



NoSQL简介

现在已经有很多公司使用了NoSQL数据库：

- Google
- Facebook
- Mozilla
- Adobe
- Foursquare
- LinkedIn
- Digg
- McGraw-Hill Education
- Vermont Public Radio
- 百度、腾讯、阿里、新浪、华为.....

NoSQL兴起的原因

1、关系数据库已经无法满足Web2.0的需求。主要表现在以下几个方面：

（1）无法满足海量数据的管理需求

（2）无法满足数据高并发的需求

（3）无法满足高可扩展性和高可用性的需求

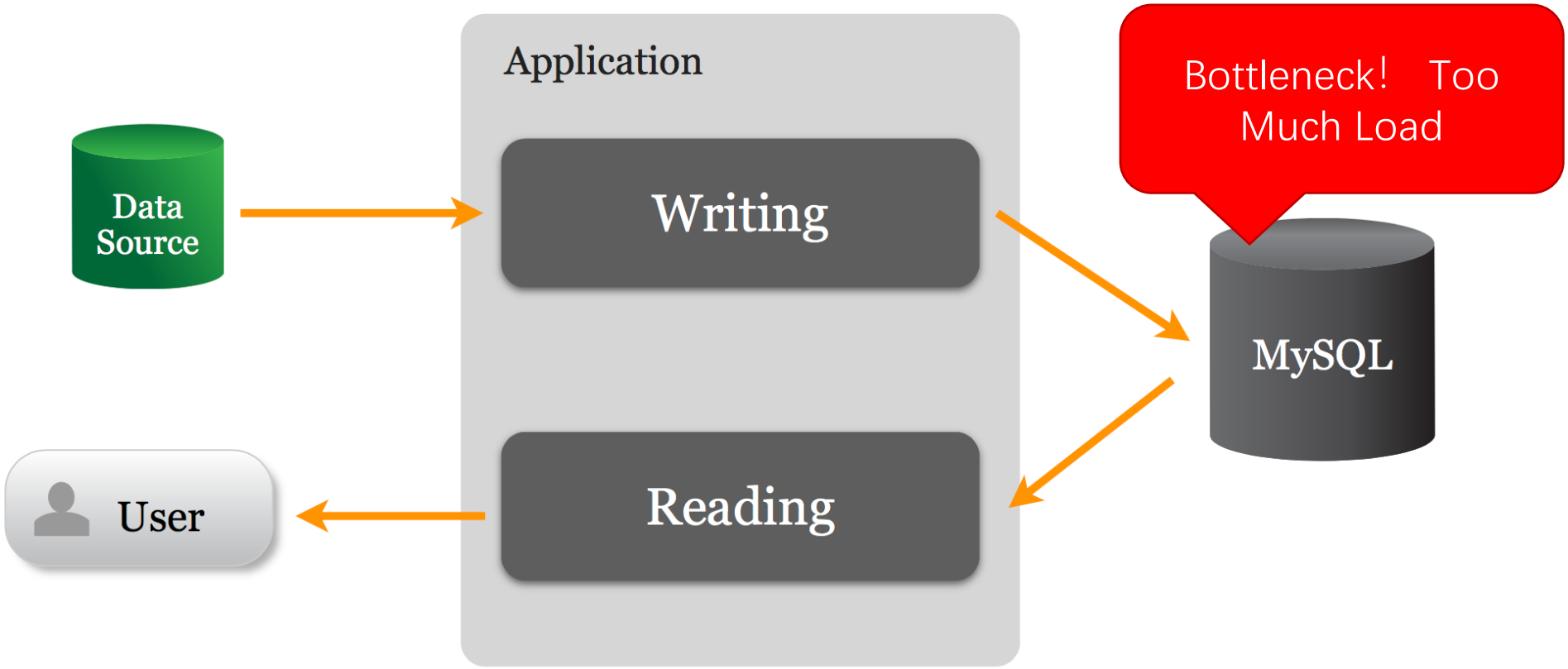
集群的挑战：

- 海量数据负载使得服务器集群成为必然选择
- 然而关系数据库并不适合用在集群上
  - 表连接查询开销巨大
  - 难以水平扩展
  - 阻抗适配问题难以解决
  - 成本高

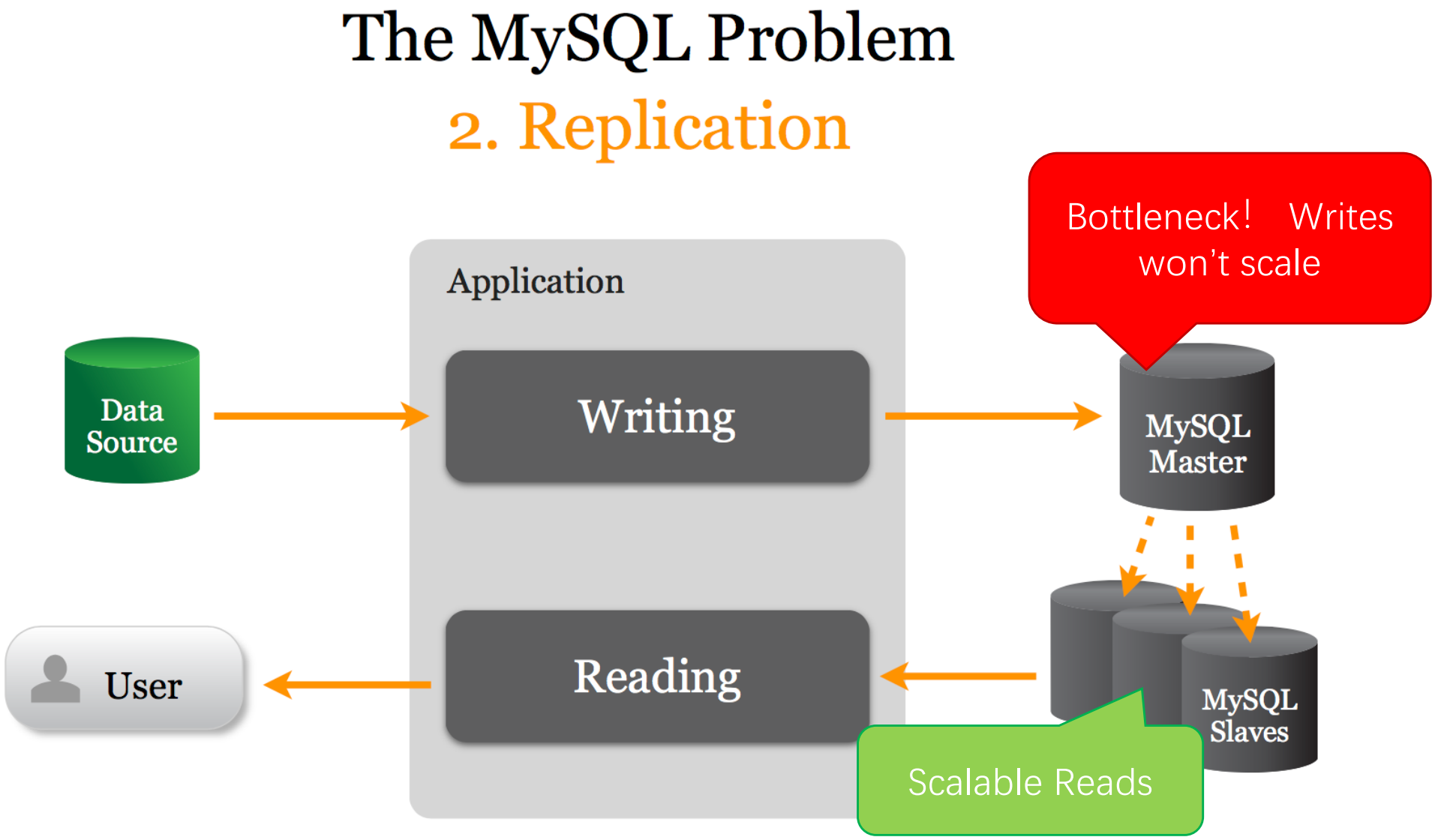
关系数据库在集群上的扩展

The MySQL Problem

1. Default



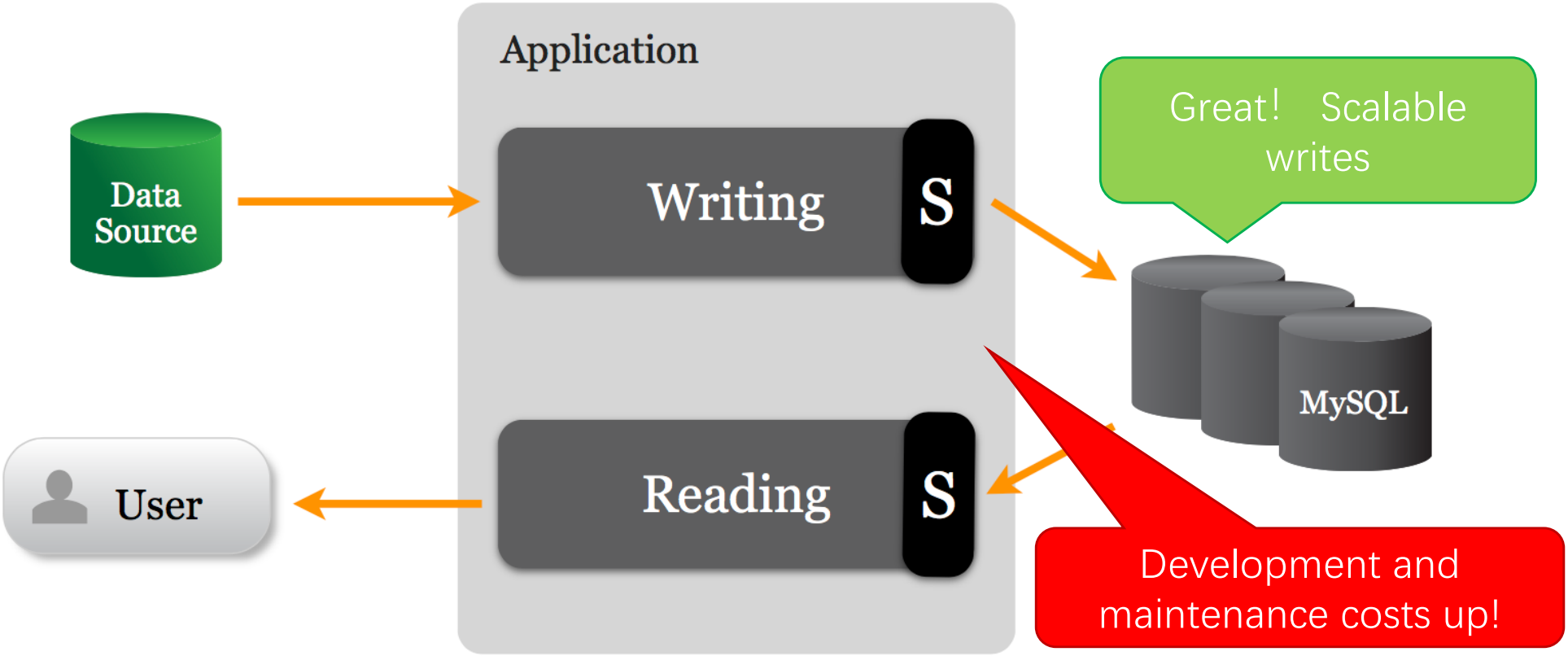
关系数据库在集群上的扩展



关系数据库在集群上的扩展

The MySQL Problem

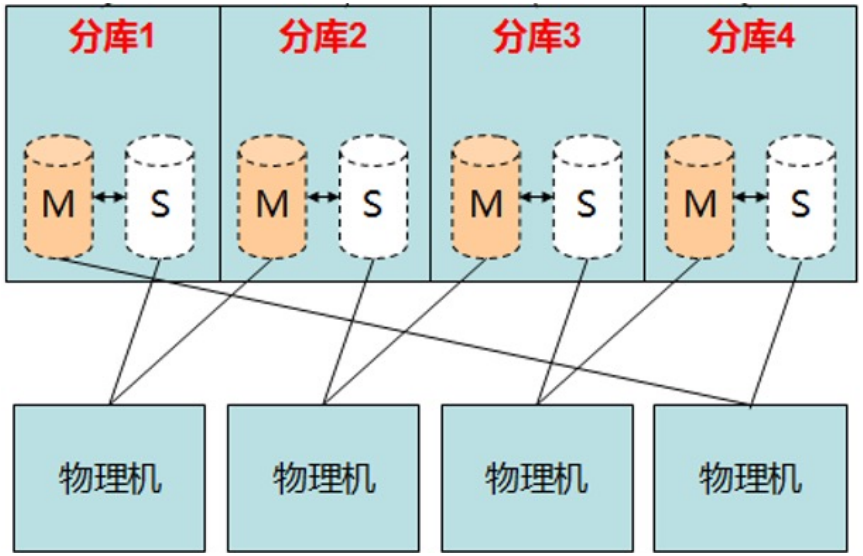
3. Sharding

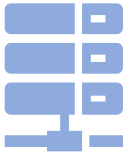


NoSQL兴起的原因

MySQL集群是否可以完全解决问题？

- **复杂性：**
  - 部署、管理、配置很复杂
- **数据库复制：**
  - 主备之间采用异步复制
  - 当主库压力较大时常产生较大延迟
  - 主备切换可能会丢失最后一部分更新事务
  - 此时常需要人工介入，备份和恢复不方便
- **扩容问题：**
  - 需增加新服务器时，过程涉及数据重新划分
  - 整个过程比较复杂，且容易出错
- **动态数据迁移问题：**
  - 因压力需将某数据库组部分数据进行迁移时，迁移过程需要总控节点整体协调，以及数据库节点的配合
  - 过程很难自动化





NoSQL兴起的原因

2、“One size fits all”模式很难适用于截然不同的业务场景

- 关系模型作为统一的数据模型既被用于数据分析，也被用于在线业务
- 数据分析强调高吞吐，在线业务强调低延时
- 用同一套模型来抽象不合适
- Hadoop就是针对数据分析
- MongoDB、Redis等是针对在线业务

两者都抛弃了关系模型

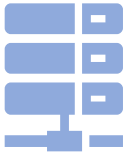
3、关系数据库的关键特性包括完善的事务机制和高效的查询机制，不适合Web2.0时代

（1）Web2.0网站系统通常不要求严格的数据库事务

（2）Web2.0并不要求严格的读写实时性

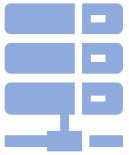
（3）Web2.0通常不包含大量复杂的SQL查询

- 网站设计时通常采用单表主键查询方式，已尽量减少多表连接、选择、投影操作



NoSQL与关系数据库的比较（1）

比较标准	RDBMS	NoSQL	备注
数据库原理	完全支持	部分支持	· RDBMS有关系代数理论作为基础 · NoSQL没有统一的理论基础
数据规模	大	超大	· RDBMS很难实现横向扩展，纵向扩展空间有限，性能会随着数据规模的增大而降低 · NoSQL可很容易通过添加设备来支持更大规模数据
数据库模式	固定	灵活	· RDBMS需要定义数据库模式，严格遵守数据定义和相关约束条件 · NoSQL不存在数据库模式，可自由灵活定义并存储各种不同类型的数据
查询效率	快	· 可高效简单查询 · 不具备高度结构化查询 · 复杂查询性能弱	· RDBMS借助于索引机制可实现快速查询（包括记录查询和范围查询） · 很多NoSQL数据库没有面向复杂查询的索引，虽然NoSQL可以使用MapReduce来加速查询，但是，在复杂查询方面的性能仍然不如RDBMS

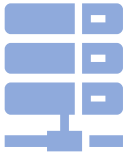


NoSQL与关系数据库的比较（2）

比较标准	RDBMS	NoSQL	备注
一致性	强一致性	弱一致性	· RDBMS严格遵守事务ACID模型，可保证事务强一致性 · 很多NoSQL数据库放松了对事务ACID四性的要求，而是遵守BASE模型，即只能保证最终一致性
数据完整性	容易实现	很难实现	· 任何RDBMS都可以很容易实现数据完整性 · NoSQL数据库却难以实现
扩展性	一般	好	· RDBMS很难实现横向扩展，纵向扩展的空间也有限 · NoSQL在设计之初就充分考虑了横向扩展的需求，可以很容易通过添加廉价设备实现扩展
可用性	好	很好	· RDBMS在任何时候都以保证数据一致性为优先目标，其次才是优化系统性能，随着数据规模的增大，为保证严格的一致性，只能提供相对较弱的可用性 · 大多数NoSQL都能提供较高的可用性

NoSQL与关系数据库的比较（3）

比较标准	RDBMS	NoSQL	备注
标准化	是	否	· RDBMS已经标准化（SQL） · NoSQL还没有行业标准，不同的NoSQL数据库都有自己的查询语言，很难规范应用程序接口
技术支持	高	低	· RDBMS经过几十年的发展，已经非常成熟，通常有很好的技术支持 · NoSQL在技术支持方面仍然处于起步阶段，还不成熟，缺乏有力的技术支持
可维护性	复杂	复杂	· RDBMS需要专门的数据库管理员(DBA)维护 · NoSQL数据库虽然没有DBMS复杂，但难以维护



NoSQL与关系数据库的比较（总结）

（1）关系数据库

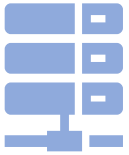
优势：以完善的**关系代数理论**作为基础，有**严格的标准**，**支持事务ACID**四性，借助索引机制可以实现**高效的查询**，**技术成熟**，有专业公司的**技术支持**

劣势：**可扩展性较差**，**无法**较好支持海量数据存储，**数据模型过于死板**、**无法**较好支持Web2.0应用，事务机制影响了**系统的整体性能**等

（2）NoSQL数据库

优势：可以**支持超大规模数据存储**，**灵活的数据模型**可以很好地支持Web2.0应用，具有**强大的横向扩展能力**等

劣势：**缺乏**数学理论基础，复杂**查询性能不高**，大都**不能实现**事务强一致性，**很难实现**数据完整性，**技术尚不成熟**，**缺乏**专业团队的技术支持，**维护较困难**等



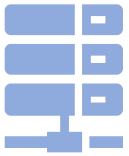
NoSQL与关系数据库的比较（总结）

两者各有优缺点，彼此无法取代

- **关系数据库应用场景**
  - 电信、银行等领域的关键业务系统，需要保证强事务一致性
- **NoSQL数据库应用场景**
  - 互联网企业、传统企业的非关键业务（比如数据分析）

采用混合架构

- 案例: 亚马逊公司就使用不同类型的数据库来支撑它的电子商务应用
- 对于“购物篮”这种临时性数据，采用键值存储会更加高效
- 当前的产品和订单信息则适合存放在关系数据库中
- 大量的历史订单信息则适合保存在类似MongoDB的文档数据库中

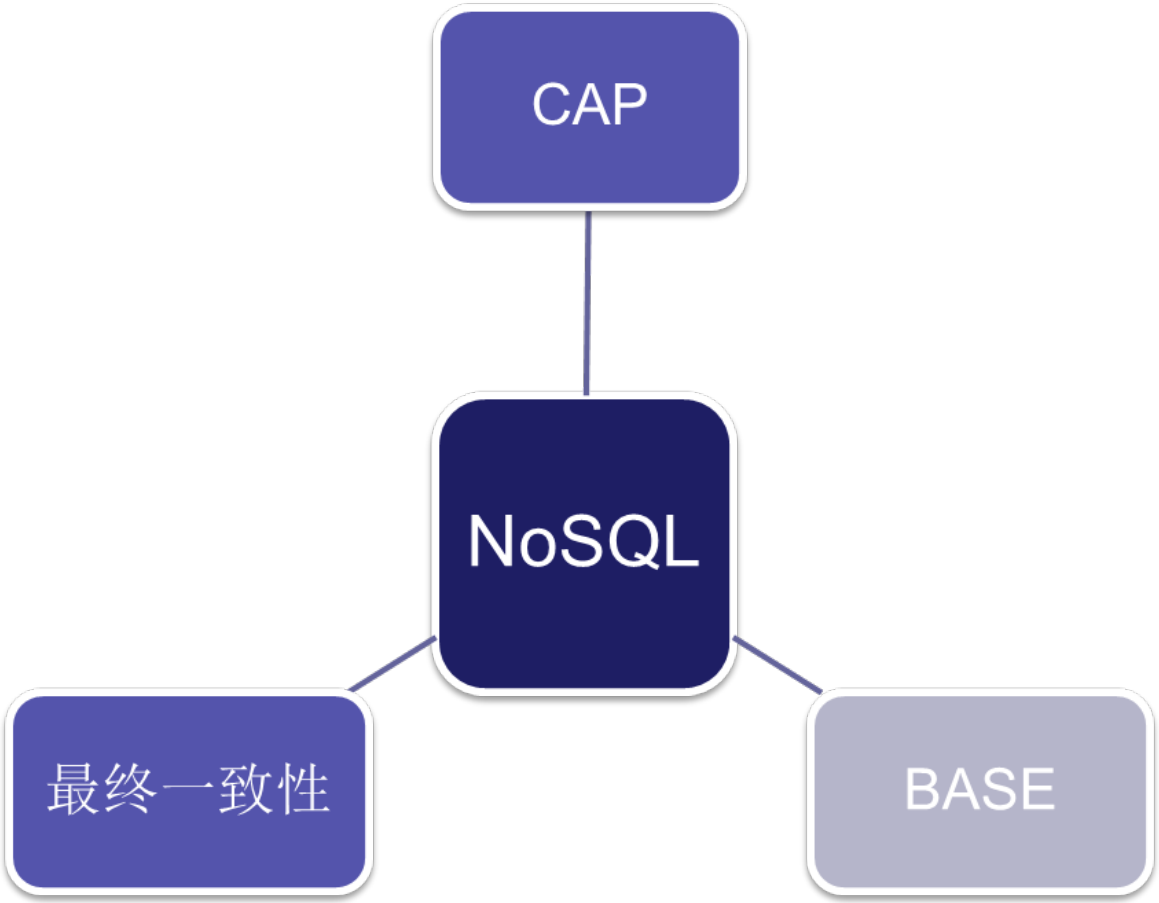


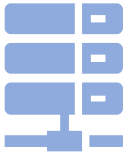
第十三章 NoSQL数据库

1 NoSQL简介

2 NoSQL的技术特点

NoSQL的三大基石





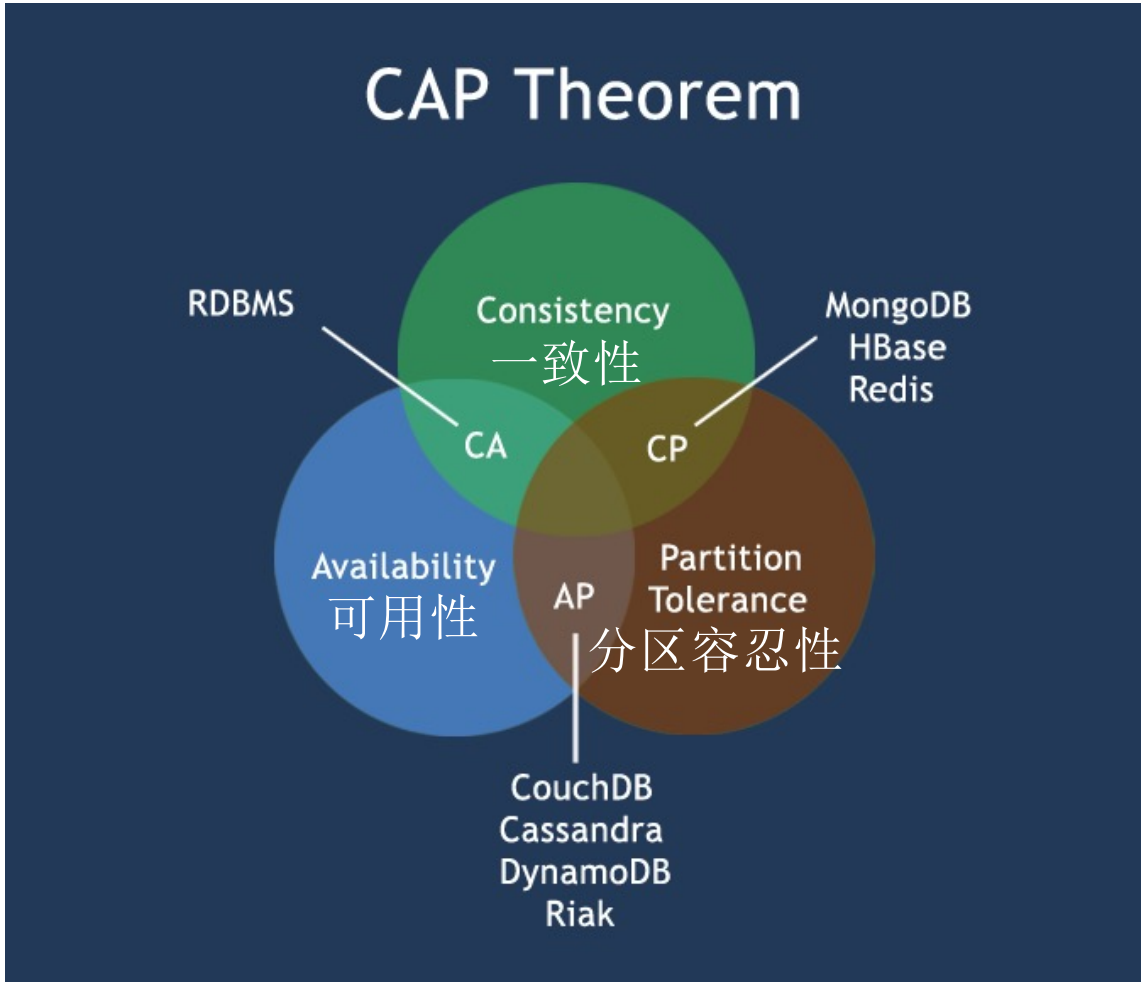
CAP理论

所谓的CAP指的是：

- **C（Consistency）：一致性**，是指任何一个读操作总是能够读到之前完成的写操作的结果，也就是在分布式环境中，多点的数据是一致的，或者说，所有节点在同一时间具有相同的数据
- **A（Availability）：可用性，**是指快速获取数据，可以在确定的时间内返回操作结果，保证每个请求不管成功或者失败都有响应；
- **P（Tolerance of Network Partition）：分区容忍性，**是指当出现网络分区的情况时（即系统中的一部分节点无法和其他节点进行通信），分离的系统也能够正常运行，也就是说，系统中任意信息的丢失或失败不会影响系统的继续运作。

CAP理论

CAP理论告诉我们，一个分布式系统不可能同时满足一致性(C)、可用性(A)和分区容忍性(P)这三个需求，最多只能同时满足其中两个——“鱼和熊掌不可兼得”



Consistency:

When I ask the same question to any part of the system I should get the same answer.

Consistency:

When I ask the same question to any part of the system I should get the same answer.

Consistency:

When I ask the same question to any part of the system I should get the same answer.

Availability:

When I ask a question I will get an answer.

Availability:

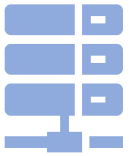
When I ask a question I will get an answer.

Partition Tolerance:

I can ask questions even if the system is having intra-system communication problems

Partition Tolerance:

I can ask questions even if the system is having intra-system communication problems



CAP理论

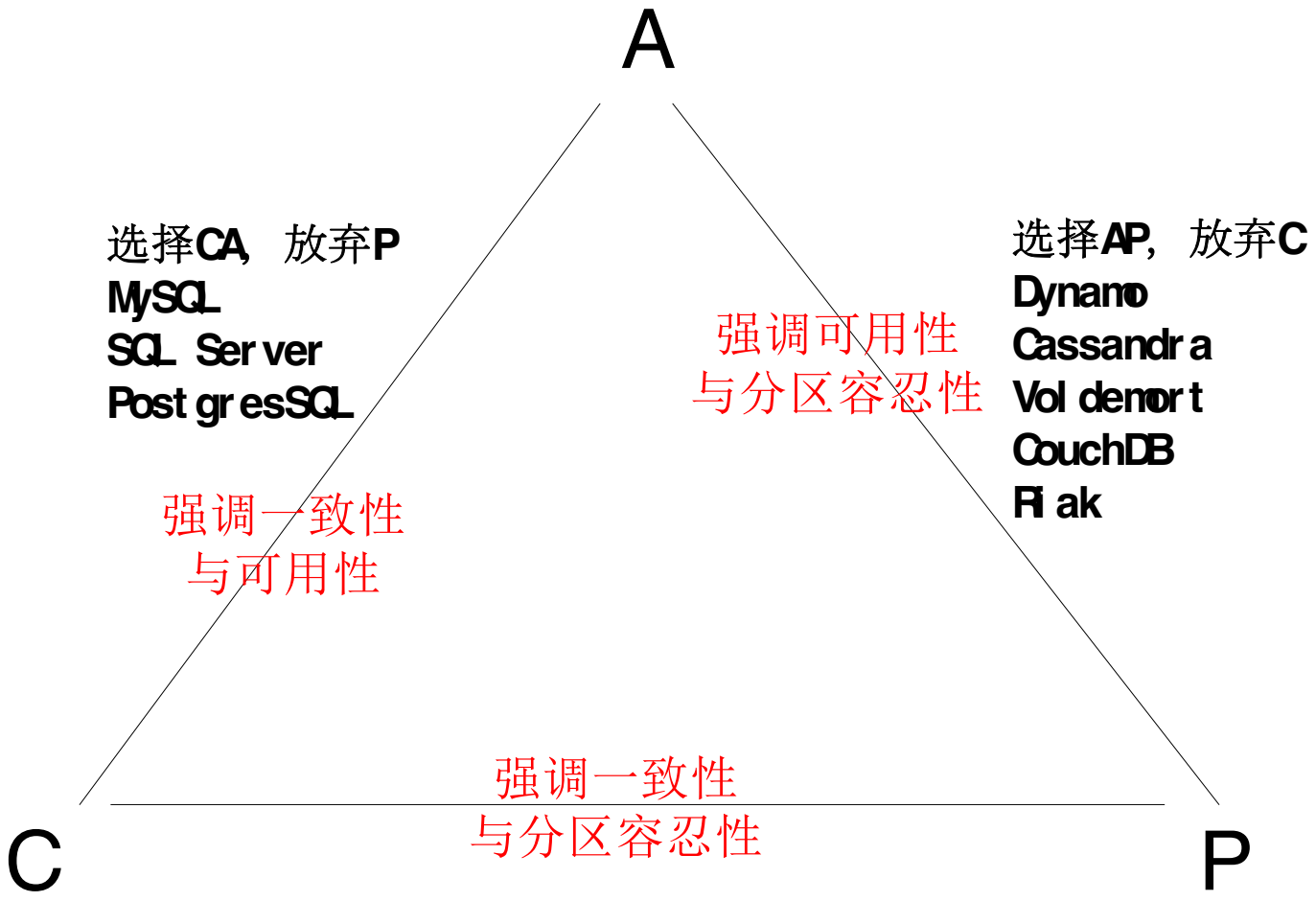
当处理CAP的问题时，可以有几个明显的选择：

1. CA：也就是**强调一致性（C）和可用性（A），**放弃分区容忍性（P），最简单的做法是把所有与事务相关的内容都放到同一台机器上。很显然，这种做法会严重影响系统的可扩展性。传统的关系数据库（MySQL、SQL Server和PostgreSQL），都采用了这种设计原则，因此，扩展性都比较差

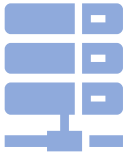
2. CP：也就是**强调一致性（C）和分区容忍性（P），**放弃可用性（A），当出现网络分区的情况时，受影响的服务需要等待数据一致，因此在等待期间就无法对外提供服务

3. AP：也就是**强调可用性（A）和分区容忍性（P），**放弃一致性（C），允许系统返回不一致的数据

CAP理论



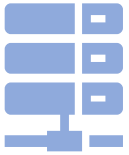
不同产品在CAP理论下的不同设计原则



BASE理论

要理解BASE(**Basically Availble, Soft-state, Eventual consistency)**，先弄清楚ACID：

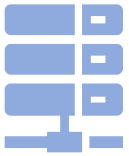
ACID	BASE
原子性(**A**tomicity)	基本可用(**B**asically **A**vailable)
一致性(**C**onsistency)	软状态/柔性事务(**S**oft state)
隔离性(**I**solation)	最终一致性 (**E**ventual consistency)
持久性 (**D**urable)	



BASE理论

一个数据库事务具有ACID四性：

- A（Atomicity）：**原子性**，是指事务必须是原子工作单元，对于其数据修改，要么全都执行，要么全都不执行
- C（Consistency）：**一致性**，是指事务在完成时，必须使所有的数据都保持一致状态
- I（Isolation）：**隔离性**，是指由并发事务所做的修改必须与任何其它并发事务所做的修改隔离
- D（Durability）：**持久性**，是指事务完成之后，它对于系统的影响是永久性的，该修改即使出现致命的系统故障也将一直保持

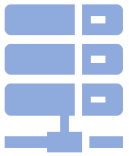


BASE理论

BASE的基本含义是基本可用（Basically Availble）、软状态（Soft-state）和最终一致性（Eventual consistency）：

- **基本可用**
  - 指一个分布式系统的一部分发生问题变得不可用时，其他部分仍然可以正常使用，也就是允许分区失败的情形出现

- **软状态**
  - 是与“硬状态（hard-state）”相对应的一种提法
  - “硬状态” 是数据库保存的数据可以保证数据一直是正确的(一致性)
  - “软状态”是指状态可以有一段时间不同步，具有一定的滞后性



BASE理论

- 最终一致性
  - 一致性的类型包括强一致性和弱一致性
  - 二者的主要区别在于高并发的数据访问操作下，后续操作是否能够获取最新的数据
  - 强一致性：当执行完一次更新操作后，后续其他读操作就可以保证读到更新后的最新数据
  - 弱一致性：如果不能保证后续访问读到的都是更新后的最新数据
  - 最终一致性：是弱一致性的一种特例，允许后续的访问操作可以暂时读不到更新后的数据，但是经过一段时间之后，必须最终读到更新后的数据
  - 最常见的实现最终一致性的系统是DNS（域名系统）。一个域名更新操作根据配置的形式被分发出去，并结合有过期机制的缓存；最终所有的客户端可以看到最新的值。

总结

- 若需严格的数据一致性、复杂查询，或业务逻辑依赖关系模型，优先选关系数据库。
- 若数据模型灵活、读写性能要求极高，或需处理海量非结构化数据，NoSQL 是更优解。
- 实际应用中两者常结合使用，形成互补。