软件架构与中间件

涂志莹 tzy_hit@hit.edu.cn

苏统华 thsu@hit.edu.cn

哈尔滨工业大学

软件架构与中间件

Software Architecture and Middleware

第4章

数据层的软件架构技术

第4章 数据层的软件架构技术

4.1 数据驱动的软件架构演化

4.2 数据读写与主从分离

4.3 数据分库分表

4.4 数据缓存

4.5 非关系型数据库

4.6 数据层架构案例

4.3
数据分库分表

1、分库分表的基本概念

2、分库分表的解决方案

3、分库分表的架构设计

4、分库分表的中间件简介

4.3.1 分库分表的基本概念

- 什么是分库分表
- 分库分表的发展阶段
- 什么情况下需要分库分表
- 分库分表的典型实例

什么是分库分表

数据库和数据库实例

- 数据库是文件的集合，是依照某种数据模型组织起来并存放于二级存储器中的数据集合；
- 数据库实例是程序，是位于用户和操作系统之间的一层数据管理软件，用户对数据库中的数据做任何的操作，包括数据定义、数据查询、数据维护、数据库运行控制等等都是在数据库实例下进行的，应用程序只有通过数据库实例才能和数据库进行交互。

- 数据库是由文件组成（一般来说都是二进制文件）的，一般不可能直接对这些二进制文件进行操作，以实现数据库的SELECT、UPDATE、INSERT和DELETE操作，需要数据库实例来完成对数据库的操作。

分库分表的基本定义

- 分库分表的本质是数据拆分，是对数据进行分而治之的通用概念。
- 为了分散数据库的压力，采用分库分表将一个表结构分为多个表，或者将一个表的数据分片后放入多个表，这些表可以放在同一个库里，也可以放到不同的库里，甚至可以放在不同的数据库实例上。

- 数据拆分主要分为：垂直拆分和水平拆分
  - 垂直拆分：根据业务的维度，将原本的一个库（表）拆分为多个库（表），每个库（表）与原有的结构不同。
  - 水平拆分：根据分片（sharding）算法，将一个库（表）拆分为多个库（表），每个库（表）依旧保留原有的结构。

分库分表的发展阶段

分库分表的发展阶段

- 单库单表
  - 例如，将所有用户的信息都存放在同一数据库里的USER表中
- 单库多表
  - 单表内数据越来越多，查询性能下降，有锁表的可能，并阻塞所有其他的操作
  - 例如，将USER表中的数据水平切分，产生多个结构完全一样的表，如User0，User1……UserN，所有表的数据加起来为原有全量的数据
- 多库多表
  - 单台数据库的存储空间不够用，增加和减少索引消耗时间过长
  - 对数据库进行水平切分，将切分的数据库和表水平地分散到不同的数据库实例上

什么情况下需要分库分表

分库分表的操作时机

- 如果在数据库中表的数量达到了一定量级，则需要进行分表，分解单表的大数据量对索引查询带来的压力，并方便对索引和表结构的变更
- 如果数据库的吞吐量达到了瓶颈，就需要增加数据库实例，利用多个数据库实例来分解大量的数据库请求带来的系统压力
- 如果希望在扩容时对应用层的配置改变最少，就需要在每个数据库实例中预留足够的数据库数量

分库分表的典型实例

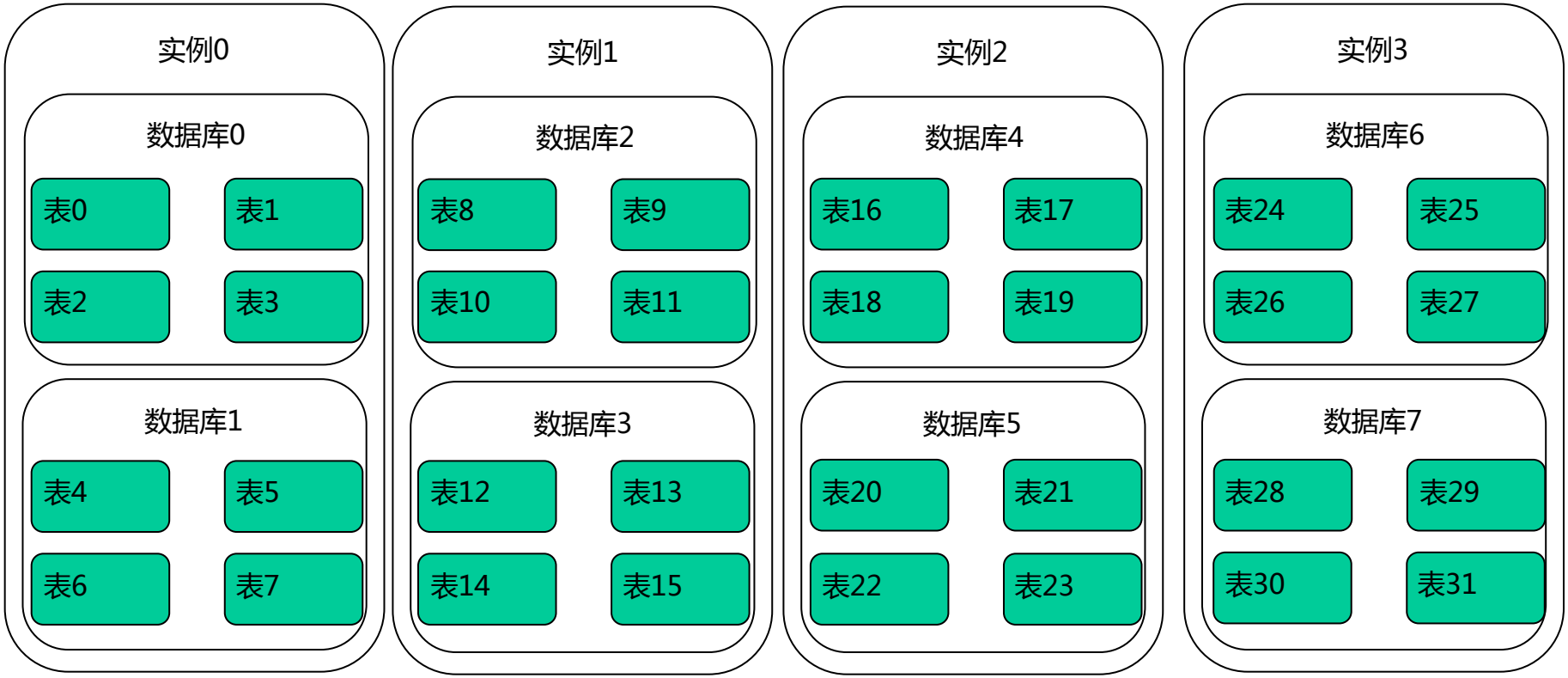
案例说明

- 某互联网应用当前拥有的16亿条用户消费数据记录需要存放在MySQL中
- 一般而言，MySQL单表存储5000万条数据记录就已经达到极限了

切片0	切片1	切片2	切片3	切片4	切片5	切片6	切片7
切片8	切片9	切片10	切片11	切片12	切片13	切片14	切片15
切片16	切片17	切片18	切片19	切片20	切片21	切片22	切片23
切片24	切片25	切片26	切片27	切片28	切片29	切片30	切片31

案例说明

- 如果将这些数据分解到4个数据库实例里，每个数据库实例包含2个数据库，每个数据库里有4个表。



4.3.2 分库分表的解决方案

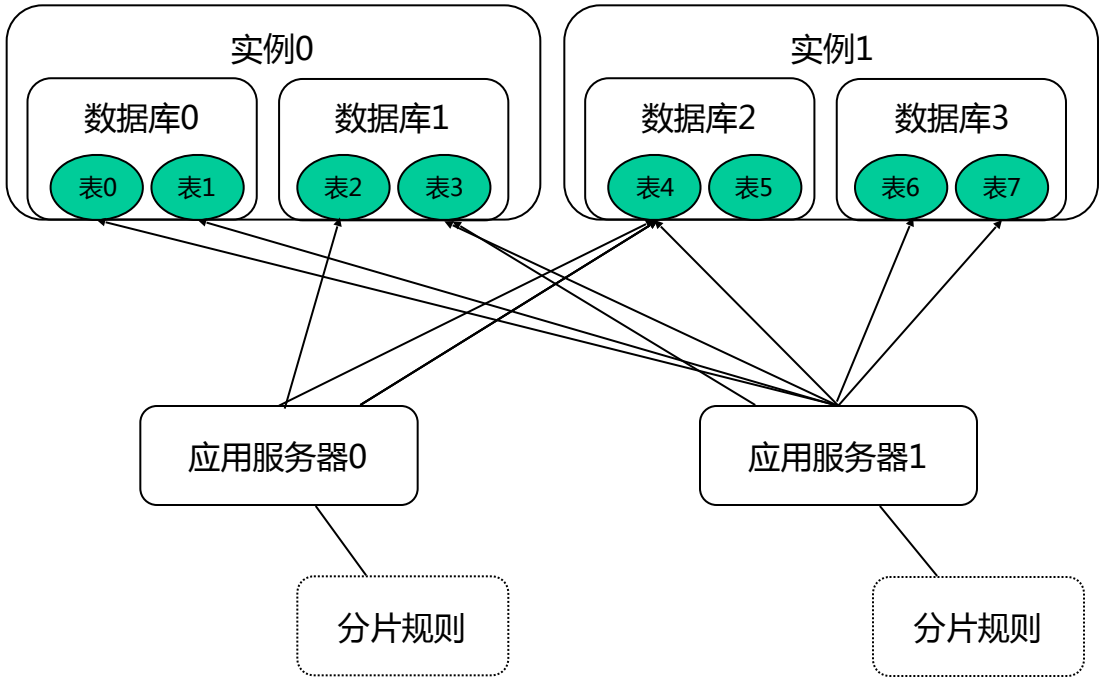
- 客户端分片
- 代理分片
- 支持事务的分布式数据库

客户端分片

基本概念

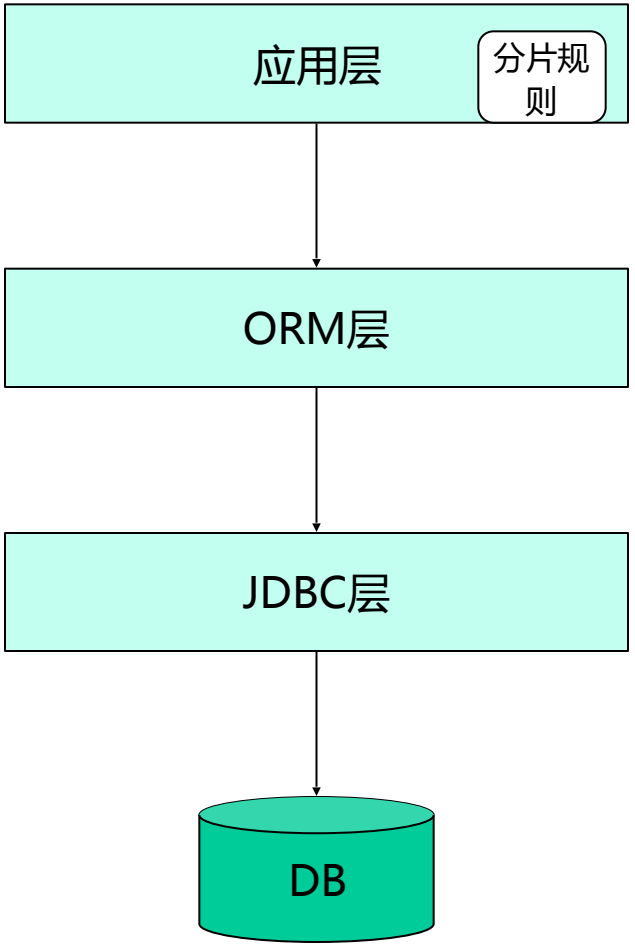
- 客户端分片就是使用分库分表的数据库的应用层直接操作分片逻辑，分片规则需要在同一个应用的多个节点间进行同步，每个应用层都嵌入一个操作切片的逻辑实现，一般通过依赖Jar包来实现。

- 具体实现方式分为三种：
  - 在应用层直接实现
  - 通过定制JDBC协议实现
  - 通过定制ORM框架实现



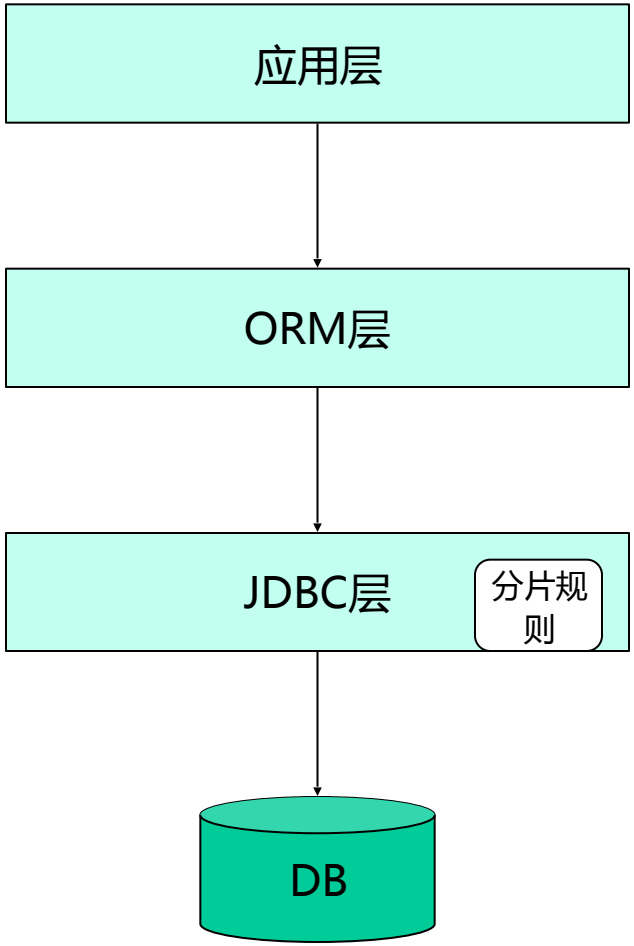
在应用层直接实现

- 直接在应用层读取分片规则，然后解析分片规则，据此实现切分的路由逻辑，从应用层直接决定每次操作应该使用哪个数据库实例、库、表等。
- 需要侵入业务，但实现简单，适合快速上线，切分逻辑由开发者自行定义，容易调试维护。但要求开发者既要实现业务逻辑，还需要实现框架需求。
- 该实现方式会让数据库保持的连接比较多，对整体应用服务器池的维护将造成压力。



通过定制JDBC协议实现

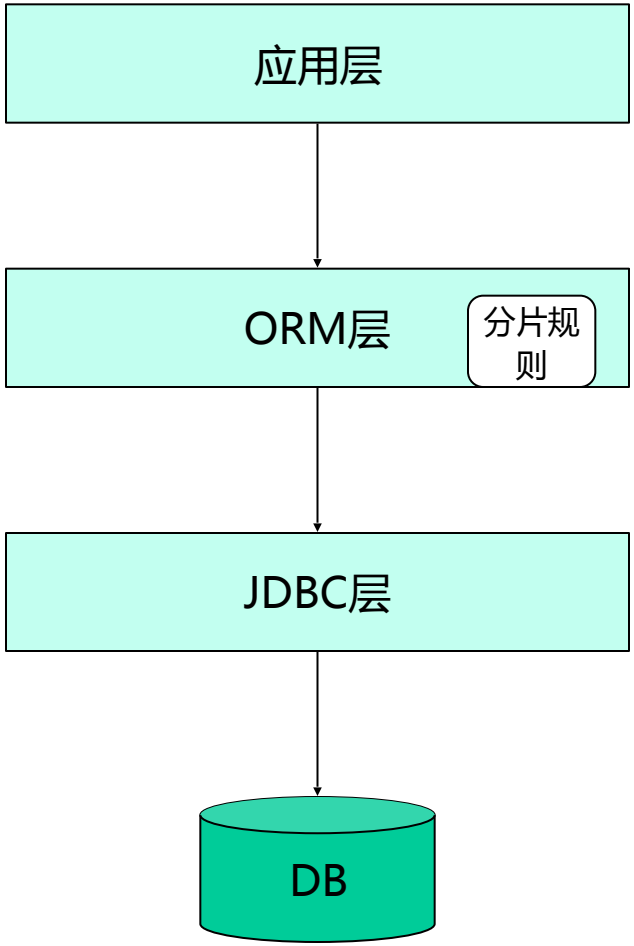
- 可让开发者集中精力实现业务逻辑，无须关心分库分表的实现。
- 通过定制JDBC协议来实现，也就是针对业务逻辑层提供与JDBC一致的接口，分库分表在JDBC的内部实现。
- 开发者需要理解JDBC协议
- 流行的框架由Sharding JDBC



通过定制ORM框架实现

- 分片规则实现到ORM框架中或者通过ORM框架支持的扩展机制来完成分库分表的逻辑。
- 以Mybatis为例：

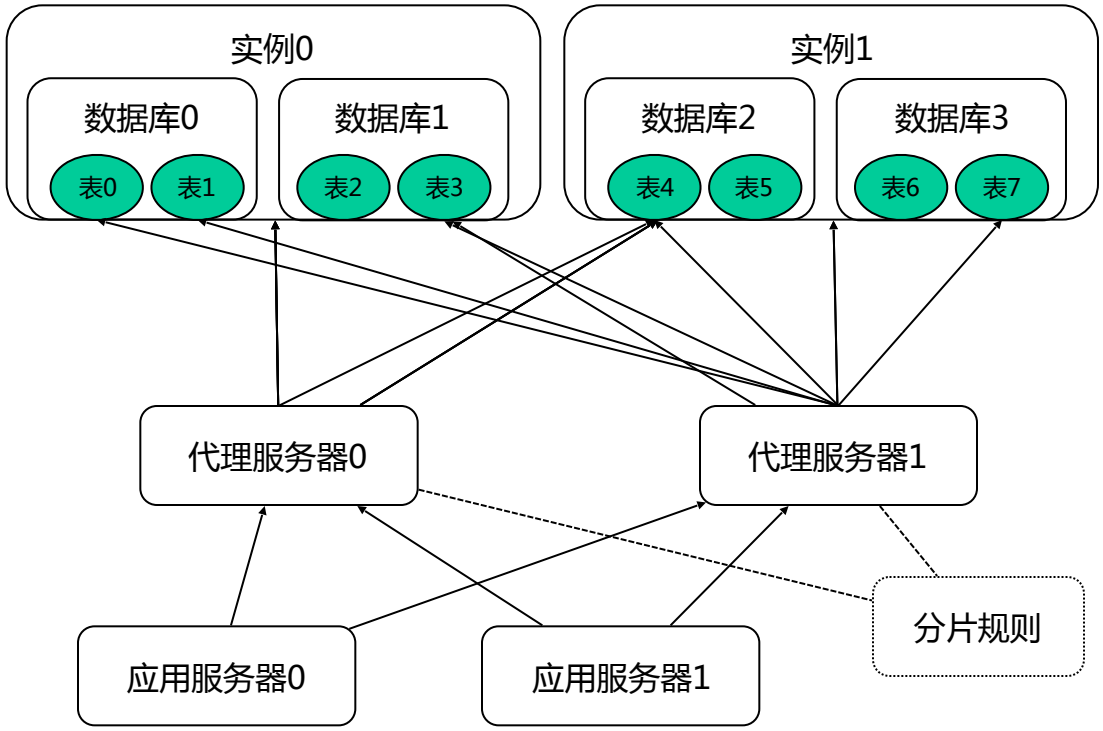
```
<select id="getUser" parameterType="java.util.Map"
            resultType="User">
    SELECT userId, username
    FROM USER_#{index}
    WHERE userId = #{userId}
</select>
```



代理分片

代理分片

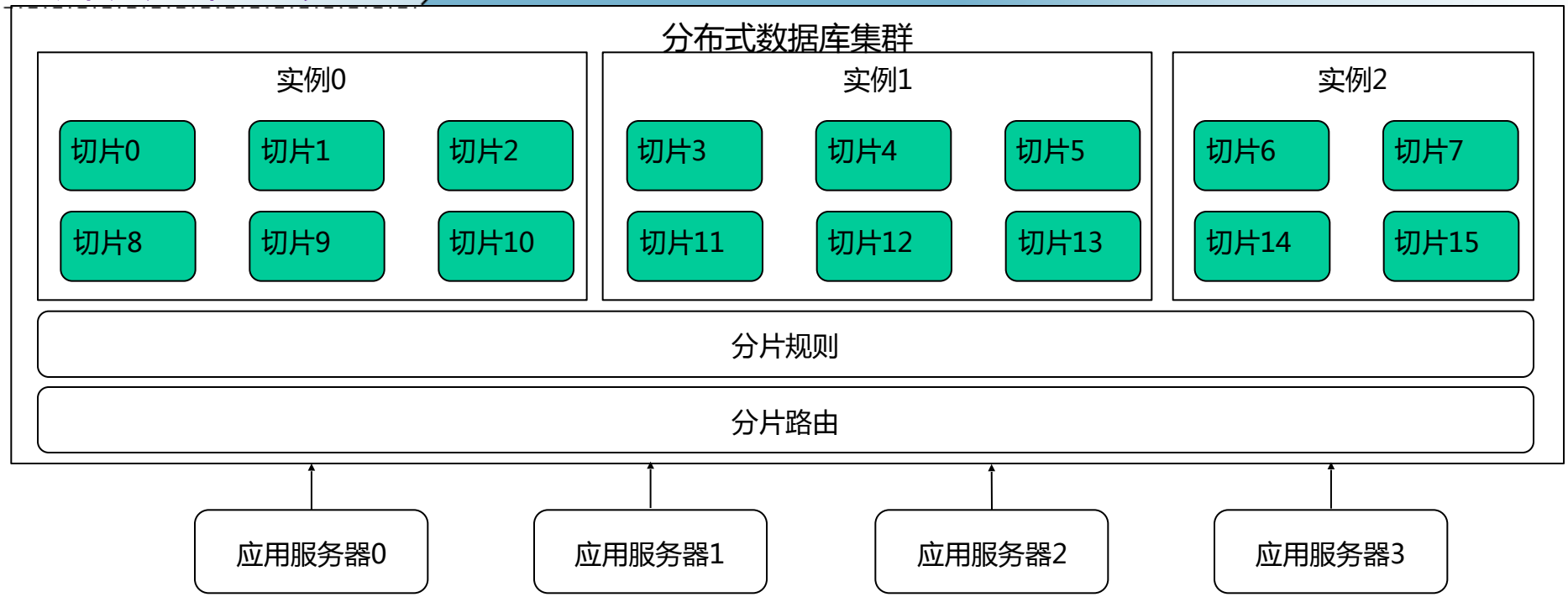
- 代理分片就是在应用层和数据库层之间增加一个代理层，把分片的路由规则配置在代理层，代理层对外提供与JDBC兼容的接口给应用层。
- 应用层的开发人员不用关心分片规则，只需关心业务逻辑的实现，待业务逻辑实现之后，在代理层配置路由规则即可。
- 代理层的引入增加了一层网络传输，对性能会造成影响。
- 需要维护代理层，增加了人员和硬件的成本。
- 可用的框架：Cobar和Mycat等。



支持事务的分布式数据库

分库分表解决方案 → 支持事务DDB

支持事务的分布式数据库



- 现在有很多产品如OceanBase、TiDB等对外提供可伸缩的体系架构，并提供一定的分布式事务支持，将可伸缩的特点和分布式事务的实现包装到分布式数据库内部，对使用者透明，使用者不需要直接控制这些特性。
- TiDB对外提供JDBC的接口，让应用层像使用MySQL等传统数据库一样，无需关注伸缩、分片、事务管理等任务。
- 目前不太适用于交易系统，较多用于大数据日志系统、统计系统、查询系统、社交网络等。

4.3.3 分库分表的架构设计

- 切分方法
- 水平切分方式的路由过程和分片维度
- 分片后的事务处理机制
- 分库分表引起的问题

切分方法

垂直切分

- 垂直切分是指按照业务将表进行分类或分拆，将其分布到不同数据库上
  - 按业务进行分库
  - 按业务进行分表
- 不同业务模块的数据可以分散到不同数据库服务器
  - 例如，User数据、Pay数据、Commodity数据
- 也可以冷热分离，根据数据的活跃度将数据进行拆分。
  - 冷数据：变化更新频率低，查询次数多的数据。
  - 热数据：变化更新频率高，活跃的数据。
- 也可以人为将一个表中的内容划分为多个表，例如将查询较多，变化不多的字段拆分成一张表放在查询性能高的服务器，而将频繁更新的字段拆分并部署到更新性能高的服务器。

垂直切分例子

- 在微博系统的设计中，一个微博对象包括文章标题、作者、分类、创建时间等属性字段，这些字段属于变化频率低的冷数据，而每篇微博的浏览数、回复数、点赞数等类似的统计信息属于变化频率高的热数据。
- 因此，一篇博客的数据可以按照冷热差异，拆分成两张表。冷数据存放的数据库可以使用MyISAM引擎，能更好地进行数据查询；热数据存放的数据库可以使用InnoDB存储引擎，更新性能好。
- 读多写少的冷数据库可以部署到缓存数据库上。

垂直切分的优缺点

- 优点：
  - 拆分后业务清晰，拆分规则明确
  - 系统之间进行整合或扩展很容易
  - 按照成本、应用的等级或类型等将表放到不同的机器上，便于管理
  - 便于实现动静分离、冷热分离的数据库表的设计模式
  - 数据维护简单
- 缺点：
  - 部分业务表无法关联（Join），只能通过接口方式解决，提高了系统的复杂度
  - 受每种业务的不同限制，存在单库性能瓶颈，不易进行数据扩展和提升性能
  - 事务处理复杂

第4章

数据层的软件架构技术

Thanks for listening

涂志莹、苏统华
哈尔滨工业大学计算机学院
企业与服务计算研究中心