# 互聯網公司常用MySQL 分庫分錶方案

叁叁人物 SQL數據庫開發 2022-03-21 08:30



> 來源：叁叁人物
> cnblogs.com/littlecharacter/p/9342129.html

## 一、數據庫瓶頸

不管是IO瓶頸，還是CPU瓶頸，最終都會導致數據庫的活躍連接數增加，進而逼近甚至達到數據庫可承載活躍連接數的閾值。在業務Service來看就是，可用數據庫連接少甚至無連接可用。接下來就可以想像了吧（並發量、吞吐量、崩潰）。

### 1、IO瓶頸

第一種：磁盤讀IO瓶頸，熱點數據太多，數據庫緩存放不下，每次查詢時會產生大量的IO，降低查詢速度-> 分庫和垂直分錶。

第二種：網絡IO瓶頸，請求的數據太多，網絡帶寬不夠-> 分庫。

### 2、CPU瓶頸

第一種：SQL問題，如SQL中包含join，group by，order by，非索引字段條件查詢等，增加CPU運算的操作-> SQL優化，建立合適的索引，在業務Service層進行業務計算。

第二種：單表數據量太大，查詢時掃描的行太多，SQL效率低，CPU率先出現瓶頸-> 水平分錶。

## 二、分庫分錶

## 1、水平分庫



**概念：**以字段為依據，按照一定策略（hash、range等），將一個庫中的數據拆分到多個庫中。

**結果：**

- 每個庫的結構都一樣；
- 每個庫的數據都不一樣，沒有交集；
- 所有庫的並集是全量數據；

**場景：**系統絕對並發量上來了，分錶難以根本上解決問題，並且還沒有明顯的業務歸屬來垂直分庫。

**分析：**庫多了，io和cpu的壓力自然可以成倍緩解。

## 2、水平分錶

**概念：**以字段為依據，按照一定策略（hash、range等），將一個表中的數據拆分到多個表中。

**結果：**

- 每個表的結構都一樣；
- 每個表的數據都不一樣，沒有交集；
- 所有表的並集是全量數據；

**場景：**系統絕對並發量並沒有上來，只是單表的數據量太多，影響了SQL效率，加重了CPU負擔，以至於成為瓶頸。

**分析：**表的數據量少了，單次SQL執行效率高，自然減輕了CPU的負擔。

### 3、垂直分庫



**概念：**以表為依據，按照業務歸屬不同，將不同的表拆分到不同的庫中。

**結果：**

- 每個庫的結構都不一樣；
- 每個庫的數據也不一樣，沒有交集；
- 所有庫的並集是全量數據；

**場景：**系統絕對並發量上來了，並且可以抽像出單獨的業務模塊。

**分析：**到這一步，基本上就可以服務化了。例如，隨著業務的發展一些公用的配置表、字典表等越來越多，這時可以將這些表拆到單獨的庫中，甚至可以服務化。再有，隨著業務的發展孵化出了一套業務模式，這時可以將相關的表拆到單獨的庫中，甚至可以服務化。

### 4、垂直分錶



**概念：**以字段为依据，按照字段的活跃性，将表中字段拆到不同的表（主表和扩展表）中。

**结果：**

- 每个表的结构都不一样；
- 每个表的数据也不一样，一般来说，每个表的字段至少有一列交集，一般是主键，用于关联数据；
- 所有表的并集是全量数据；

**场景：**系统绝对并发量并没有上来，表的记录并不多，但是字段多，并且热点数据和非热点数据在一起，单行数据所需的存储空间较大。以至于数据库缓存的数据行减少，查询时会去读磁盘数据产生大量的随机读IO，产生IO瓶颈。

**分析：**可以用列表页和详情页来帮助理解。垂直分表的拆分原则是将热点数据（可能会冗余经常一起查询的数据）放在一起作为主表，非热点数据放在一起作为扩展表。这样更多的热点数据就能被缓存下来，进而减少了随机读IO。拆了之后，要想获得全部数据就需要关联两个表来取数据。但记住，千万别用join，因为join不仅会增加CPU负担并且会讲两个表耦合在一起（必须在一个数据库实例上）。关联数据，应该在业务Service层做文章，分别获取主表和扩展表数据然后用关联字段关联得到全部数据。

## 三、分库分表工具

- sharding-sphere：jar，前身是sharding-jdbc；
- TDDL：jar，Taobao Distribute Data Layer；
- Mycat：中间件。

> 注：工具的利弊，请自行调研，官网和社区优先。

## 四、分库分表步骤

根据容量（当前容量和增长量）评估分库或分表个数 -> 选key（均匀）-> 分表规则（hash或range等）-> 执行（一般双写）-> 扩容问题（尽量减少数据的移动）。

## 五、分库分表问题

### 1、非partition key的查询问题

基于水平分库分表，拆分策略为常用的hash法。

**端上除了partition key只有一个非partition key作为条件查询**

映射法

基因法



> 注：写入时，基因法生成user_id，如图。关于xbit基因，例如要分8张表，23=8，故x取3，即3bit基因。根据user_id查询时可直接取模路由到对应的分库或分表。根据user_name查询时，先通过user_name_code生成函数生成user_name_code再对其取模路由到对应的分库或分表。id生成常用snowflake算法。

**端上除了partition key不止一个非partition key作为条件查询**

映射法

冗余法



> 注：按照order_id或buyer_id查询时路由到db_o_buyer库中，按照seller_id查询时路由到db_o_seller库中。感觉有点本末倒置！有其他好的办法吗？改变技术栈呢？

**后台除了partition key还有各种非partition key组合条件查询**

NoSQL法

冗余法



### 2、非partition key跨库跨表分页查询问题

基于水平分库分表，拆分策略为常用的hash法。

> 注：用NoSQL法解决（ES等）。

### 3、扩容问题

基于水平分库分表，拆分策略为常用的hash法。

**水平扩容库（升级从库法）**



> 注：扩容是成倍的。

**水平扩容表（双写迁移法）**



- 第一步：（同步双写）修改应用配置和代码，加上双写，部署；
- 第二步：（同步双写）将老库中的老数据复制到新库中；
- 第三步：（同步双写）以老库为准校对新库中的老数据；
- 第四步：（同步双写）修改应用配置和代码，去掉双写，部署；

> 注：双写是通用方案。

## 六、分库分表总结

- 分库分表，首先得知道瓶颈在哪里，然后才能合理地拆分（分库还是分表？水平还是垂直？分几个？）。且不可为了分库分表而拆分。
- 选key很重要，既要考虑到拆分均匀，也要考虑到非partition key的查询。
- 只要能满足需求，拆分规则越简单越好。

## 七、分库分表示例

> 示例GitHub地址：https://github.com/littlecharacter4s/study-sharding

我是岳哥，最后给大家分享我写的SQL两件套：《SQL基础知识第二版》和《SQL高级知识第二版》的PDF电子版。里面有各个语法的解释、大量的实例讲解和批注等等，非常通俗易懂，方便大家跟着一起来实操。

有需要的读者可以下载学习，在下面的公众号「数据前线」(非本号)后台回复关键字：SQL，就行

数据前线

数据前线
专注数据相关领域，主要分享MySQL，数据分析，Python，Excel 等相关技术内...
58篇原创内容

公众号

后台回复关键字：1024，获取一份精心整理的技术干货

后台回复关键字：进群，带你进入高手如云的交流群。

## 推荐阅读

- 一条SQL能占多大内存?
- SQL中的递归查询
- SQL高效查询建议
- MySQL中，21个写SQL的好习惯
- Linux系统常用命令速查手册

喜欢此内容的人还喜欢

大数据算法天花乱坠的时代，如何识别“数据陷阱”?
中信出版

數字化運維時代專家沒用了嗎

白鱔的洞穴

高逼格的SQL 寫法：行行比較，別問為什麼，問就是逼格高。。

戀習Python