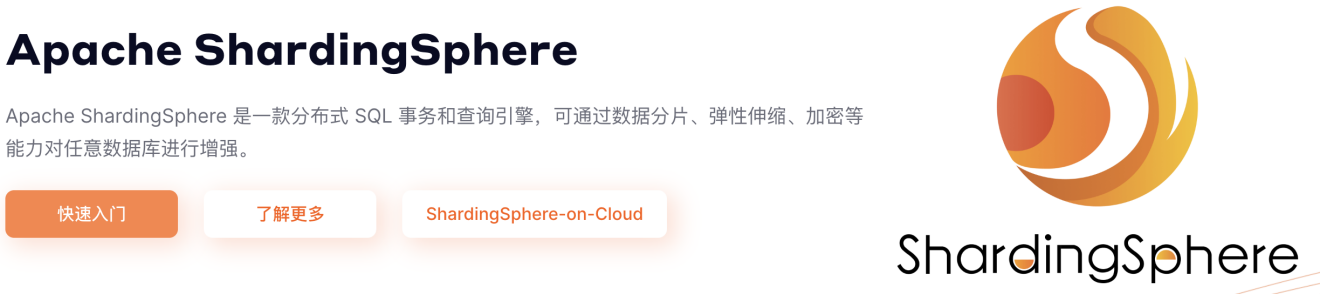
道听途说：ShardingSphere产品介绍

-- 楼兰

这一章我们快速了解即将要面对的ShardingSphere是一个什么样的框架。

一、ShardingSphere分库分表产品介绍

shardingsphere官网地址：https://shardingsphere.apache.org/



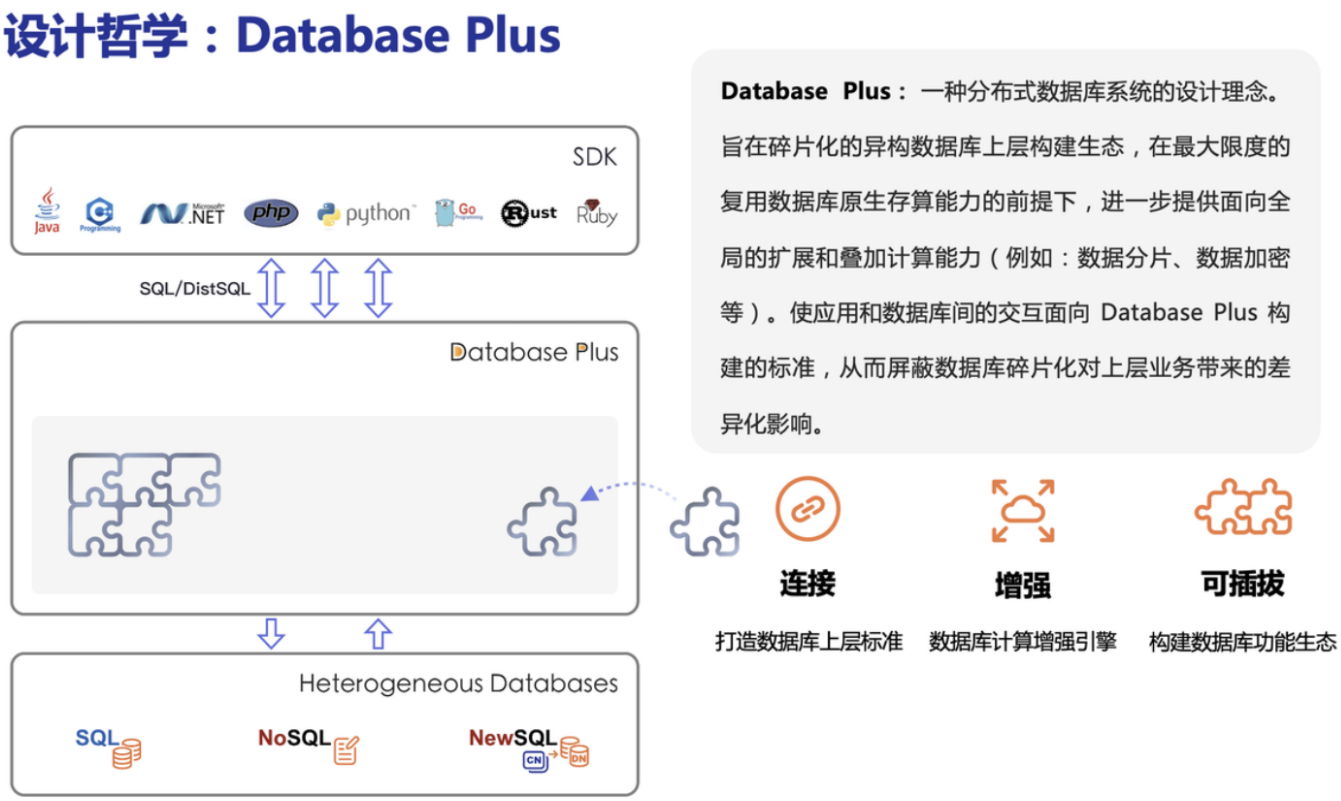
ShardingSphere是一款起源于当当网内部的应用框架。2015年在当当网内部诞生，最初就叫ShardingJDBC。2016年的时候，由其中一个主要的开发人员张亮，带入到京东数科，组件团队继续开发。在国内历经了当当网、电信翼支付、京东数科等多家大型互联网企业的考验，在2017年开始开源。并逐渐由原本只关注于关系型数据库增强工具的ShardingJDBC升级成为一整套以数据分片为基础的数据生态圈，更名为ShardingSphere。到2020年4月，已经成为了Apache软件基金会的顶级项目。发展至今，已经成为了业界分库分表最成熟的产品。

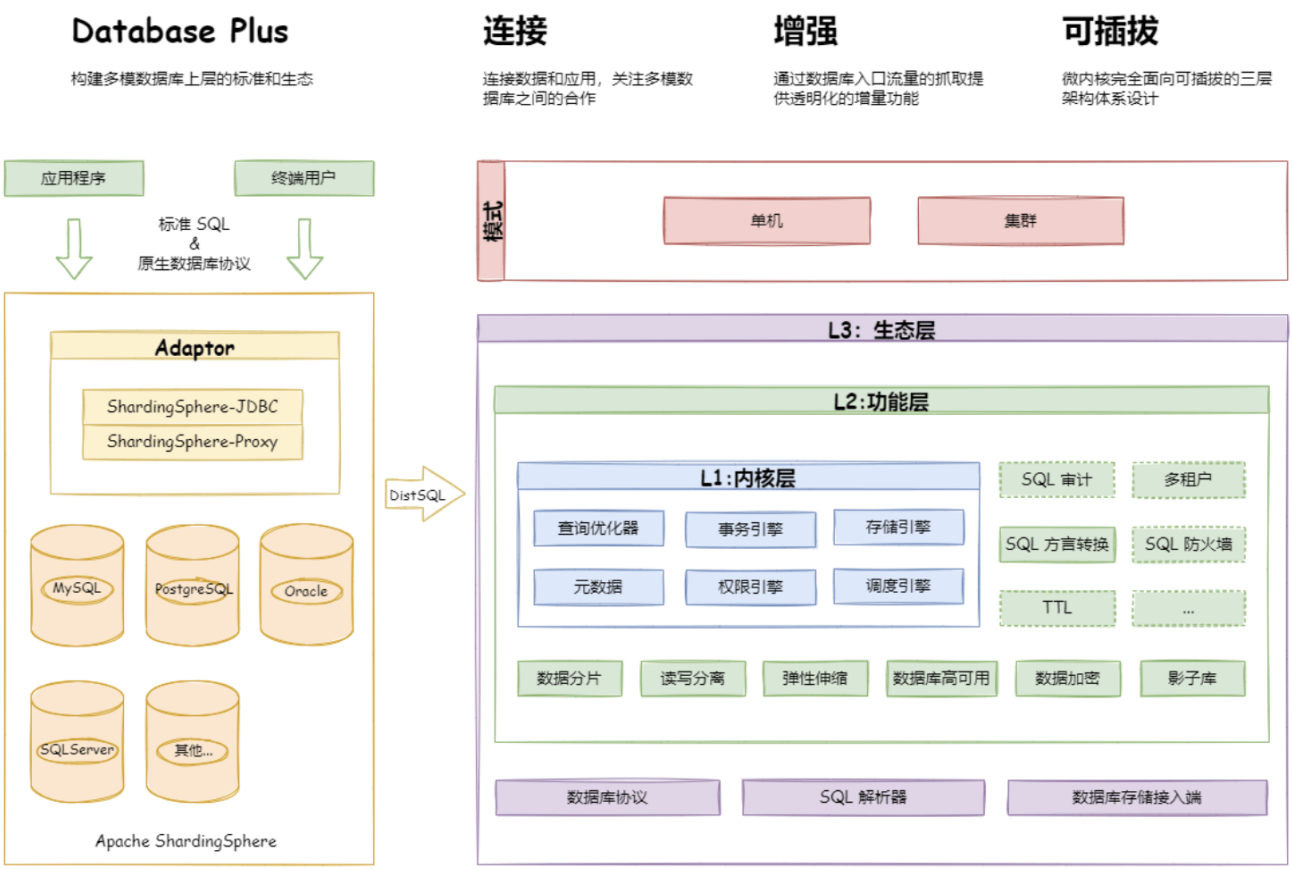
ShardingSphere这个词可以分为两个部分，其中Sharding就是我们之前介绍过的数据分片。从官网介绍上就能看到，他的核心功能就是可以将任意数据库组合，转换成为一个分布式的数据库，提供整体的数据库集群服务。只是给自己的定位并不是Database，而是Database plus。其实这个意思是他自己并不做数据存储，而是对其他数据库产品进行整合。整个ShardingSphere其实就是围绕数据分片这个核心功能发展起来的。

后面的Sphere是生态的意思。这意味着ShardingSphere不是一个单独的框架或者产品，而是一个由多个框架以及产品构成的一个完整的技术生态。目前ShardingSphere中比较成型的产品主要包含核心的ShardingJDBC以及ShardingProxy两个产品，以及一个用于数据迁移的子项目ElasticJob。另外现在也提供了基于公有云的云上服务ShardingSphere-on-Cloud。

ShardingSphere经过这么多年的发展，已经不仅仅只是用来做分库分表，而是形成了一个围绕分库分表核心的技术生态。他的核心功能已经包括了数据分片、分布式事务、读写分离、高可用、数据迁移、联邦查询、数据加密、影子库、DistSQL庞大的技术体系。

目前ShardingSphere已经演进到了5.x大版本。在这个大版本中，ShardingSphere的核心是可插拔。其核心设计哲学就是连接、增强以及可拔插。这是官网对于其整个设计哲学的核心描述。





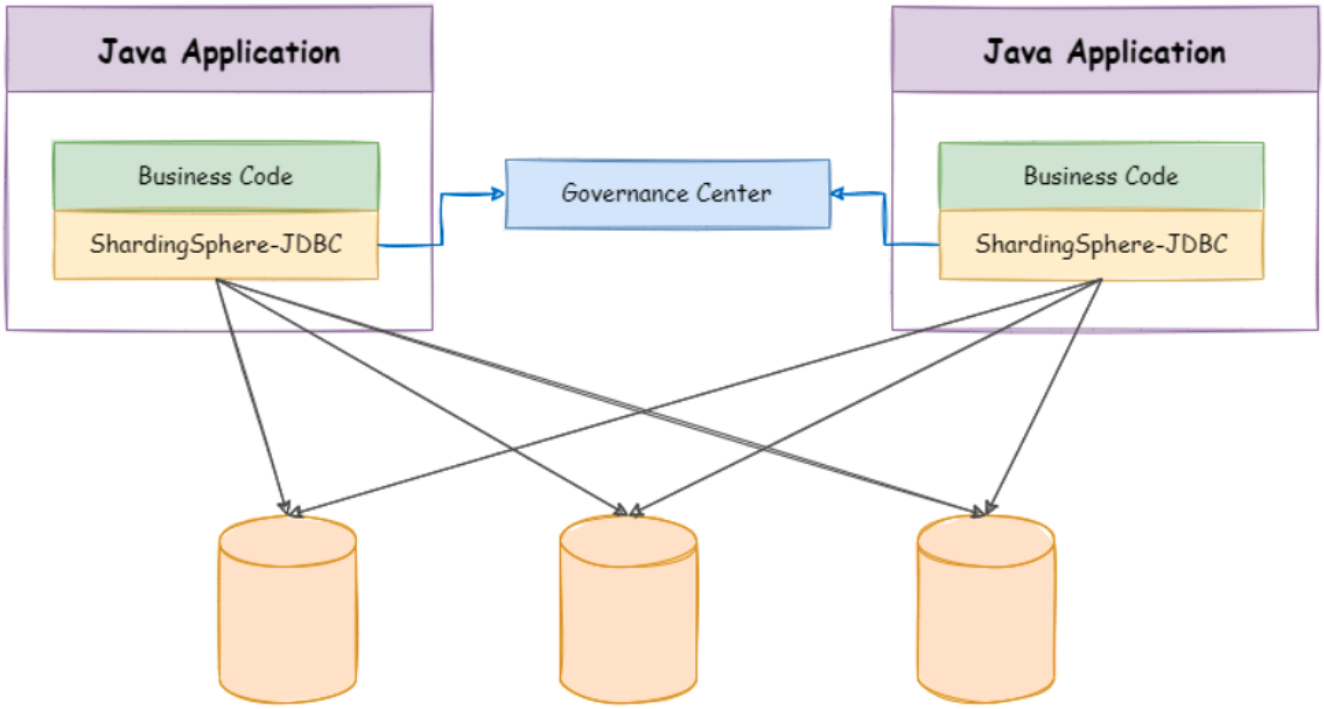
二、客户端分库分表与服务端分库分表

ShardingSphere最为核心的产品有两个：一个是ShardingJDBC，这是一个进行客户端分库分表的框架。另一个是ShardingProxy，这是一个进行服务端分库分表的产品。他们代表了两种不同的分库分表的实现思路。

1、ShardingJDBC客户端分库分表

ShardingSphere-JDBC 定位为轻量级 Java 框架，在 Java 的 JDBC 层提供的额外服务。 它使用客户端直连数据库，以 jar 包形式提供服务，无需额外部署和依赖，可理解为增强版的 JDBC 驱动，完全兼容 JDBC 和各种 ORM 框架。

- 适用于任何基于 JDBC 的 ORM 框架，如：JPA, Hibernate, Mybatis, Spring JDBC Template 或直接使用 JDBC；
- 支持任何第三方的数据库连接池，如：DBCP, C3P0, BoneCP, HikariCP 等；
- 支持任意实现 JDBC 规范的数据库，目前支持 MySQL，PostgreSQL，Oracle，SQLServer 以及任何可使用 JDBC 访问的数据库。

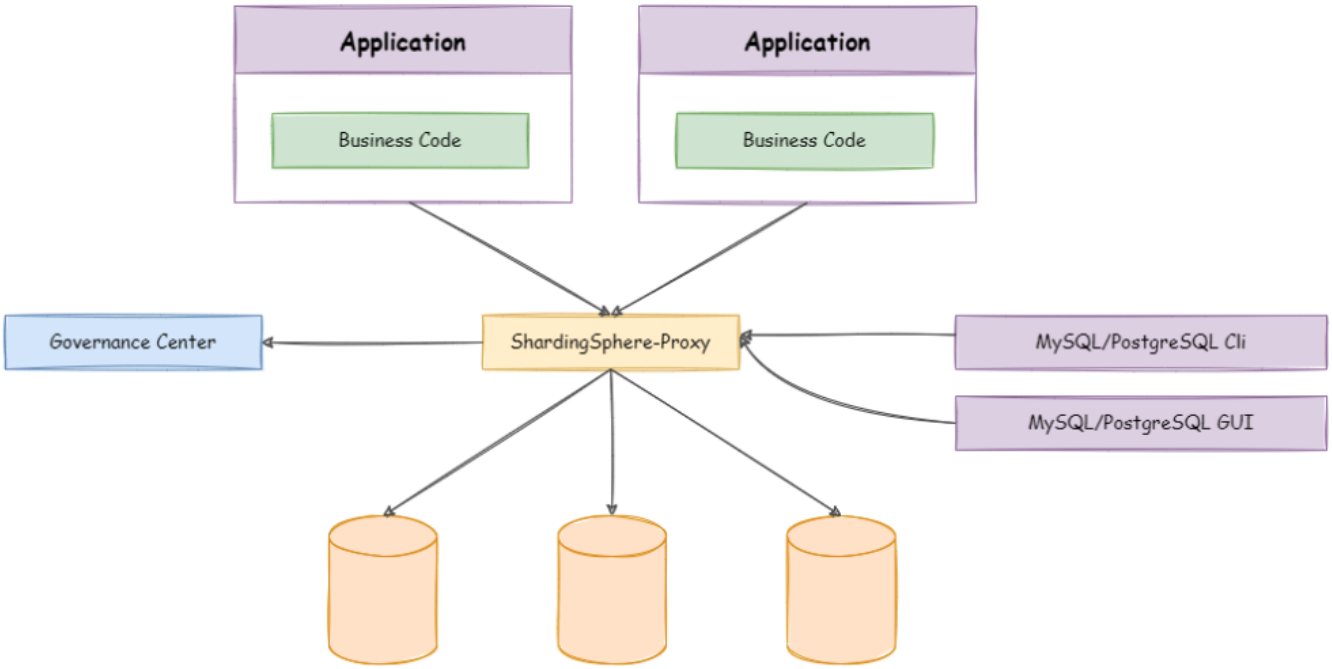


这是ShardingSphere最初的产品形态。

2、ShardingProxy服务端分库分表

ShardingSphere-Proxy 定位为透明化的数据库代理端，通过实现数据库二进制协议，对异构语言提供支持。 目前提供 MySQL 和 PostgreSQL 协议，透明化数据库操作，对 DBA 更加友好。

- 向应用程序完全透明，可直接当做 MySQL/PostgreSQL 使用；
- 兼容 MariaDB 等基于 MySQL 协议的数据库，以及 openGauss 等基于 PostgreSQL 协议的数据库；
- 适用于任何兼容 MySQL/PostgreSQL 协议的的客户端，如：MySQL Command Client, MySQL Workbench, Navicat 等。



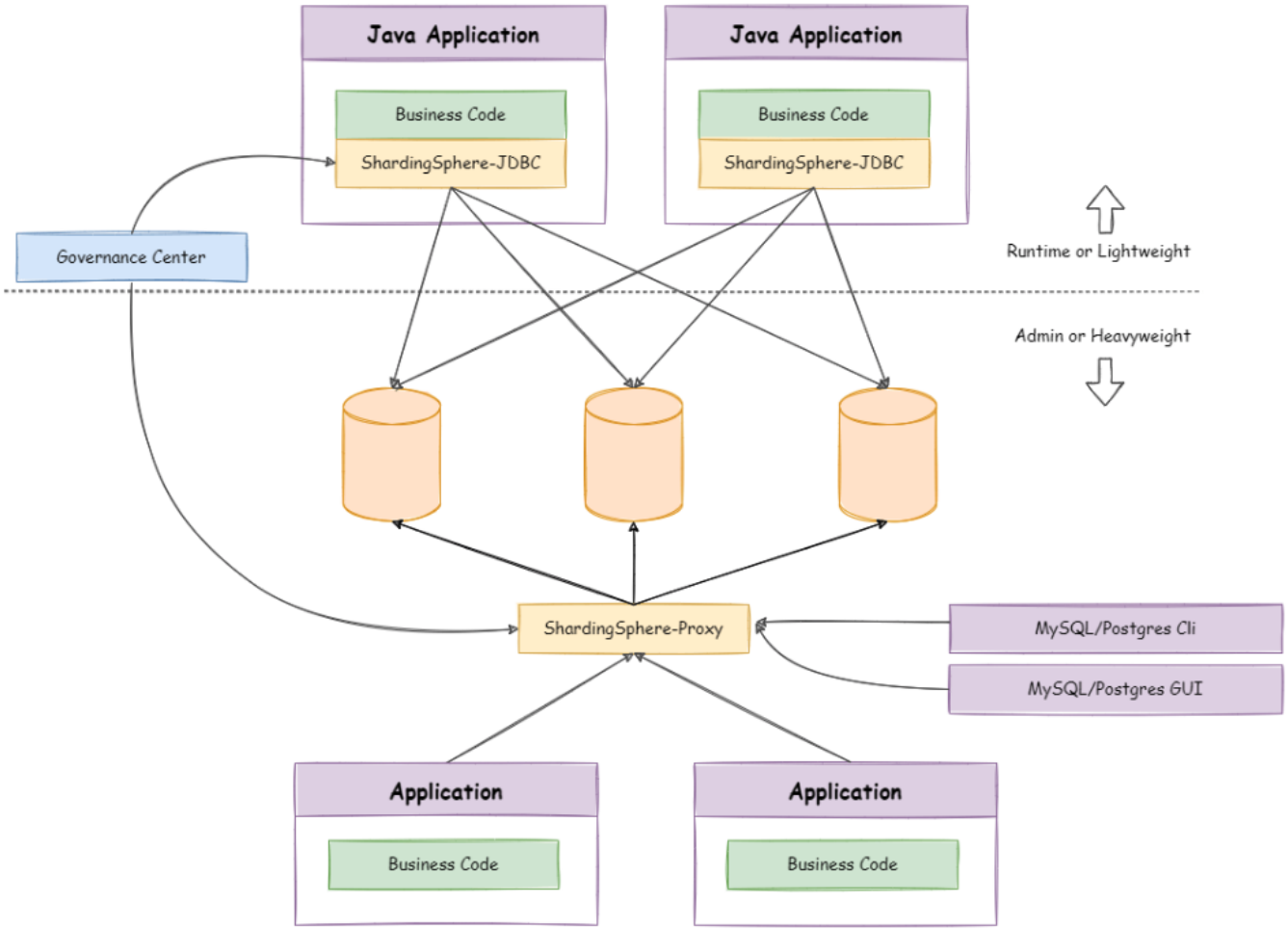
3、ShardingSphere混合部署架构

这两个产品都各有优势。ShardingJDBC跟客户端在一起，使用更加灵活。而ShardingProxy是一个独立部署的服务，所以他的功能相对就比较固定。他们的整体区别如下：

	ShardingSphere-JDBC	ShardingSphere-Proxy
数据库	任意	`MySQL/PostgreSQL`
连接消耗数	高	低
异构语言	仅 Java	任意
性能	损耗低	损耗略高
无中心化	是	否
静态入口	无	有

另外，在产品图中，Governance Center也是其中重要的部分。他的作用有点类似于微服务架构中的配置中心，可以使用第三方服务统一管理分库分表的配置信息，当前建议使用的第三方服务是Zookeeper，同时也支持Nacos，Etcd等其他第三方产品。

由于ShardingJDBC和ShardingProxy都支持通过Governance Center，将配置信息交个第三方服务管理，因此，也就自然支持了通过Governance Center进行整合的混合部署架构。



我们接下来会使用当前最新的5.5.0版本的ShardingSphere。在学习过程中，除了要学会ShardingSphere的各种使用方法，更要逐步深入ShardingSphere的内核，尝试去理解ShardingSphere的这些设计哲学。ShardingSphere处理分库分表相关问题的思路，才是最大的价值所在。

三、分库分表，能不分就不分！

1、为什么要分库分表?

数据库，应该是一个应用当中最为核心的价值所在，也是开发过程中必须熟练掌握的工具。之前我们就学习过很多对MySQL的调优。但是随着现在互联网应用越来越大，数据库会频繁的成为整个应用的性能瓶颈。我们经常使用的MySQL数据库，也就不断面临数据量太大、数据访问太频繁、数据读写速度太快等一系列的问题。而传统的这些调优方式，在真正面对海量数据冲击时，往往就会显得很无力。因此，现在互联网对于数据库的使用也越来越小心谨慎。例如添加Redis缓存、增加MQ进行流量削峰等。但是，数据库本身如果性能得不到提升，这就相当于是水桶理论中的最短板。

要提升数据库的性能，最直接的思路，当然是对数据库本身进行优化。例如对MySQL进行调优，优化SQL逻辑，优化索引结构，甚至像阿里等互联网大厂一样，直接优化MySQL的源码。但是这种思路在面对互联网环境时，会有很多非常明显的弊端。

- 数据量和业务量快速增长，会带来性能瓶颈、服务宕机等很多问题。
- 单点部署的数据库无法保证服务高可用。
- 单点部署的数据库无法进行水平扩展，难以应对业务爆发式的增长。

这些问题背后的核心还是数据。数据库不同于上层的一些服务，他所管理的数据甚至比服务本身更重要。即要保证数据能够持续稳定的写入，又不能因为服务故障造成数据丢失。现在互联网上的大型应用，动辄几千万上亿的数据量，就算做好数据的压缩，随随便便也可以超过任何服务器的存储能力。并且，服务器单点部署，也无法真正解决把鸡蛋放在一个篮子里的问题。将数据放在同一个服务器上，如果服务器出现崩溃，就很难保证数据的安全性。这些都不是光靠优化MySQL产品，优化服务器配置能够解决的问题。

2、分库分表的优势

那么自然就需要换另外一种思路了。我们可以像微服务架构一样，来维护数据库的服务。把数据库从单体服务升级到数据库集群，这样才能真正全方位解放数据库的性能瓶颈，并且能够通过水平扩展的方式，灵活提升数据库的存储能力。这也就是我们常说的分库分表。通过分库分表可以给数据库带来很大的好处：

- 提高系统性能：分库分表可以将大型数据库分成多个小型数据库，每个小型数据库只需要处理部分数据，因此可以提高数据库的并发处理能力和查询性能。
- 提高系统可用性：分库分表可以将数据复制到多个数据库中，以提高数据的可用性和可靠性。如果一个数据库崩溃了，其他数据库可以接管其工作，以保持系统的正常运行。
- 提高系统可扩展性：分库分表可以使系统更容易扩展。当数据量增加时，只需要增加更多的数据库和表，而不是替换整个数据库，因此系统的可扩展性更高。
- 提高系统灵活性：分库分表可以根据数据的使用情况，对不同的数据库和表进行不同的优化。例如，可以将经常使用的数据存储在性能更好的数据库中，或者将特定类型的数据存储在特定的表中，以提高系统的灵活性。
- 降低系统成本：分库分表可以使系统更加高效，因此可以降低系统的运营成本。此外，分库分表可以使用更便宜的硬件和存储设备，因为每个小型数据库和表需要的资源更少。

3、分库分表的挑战

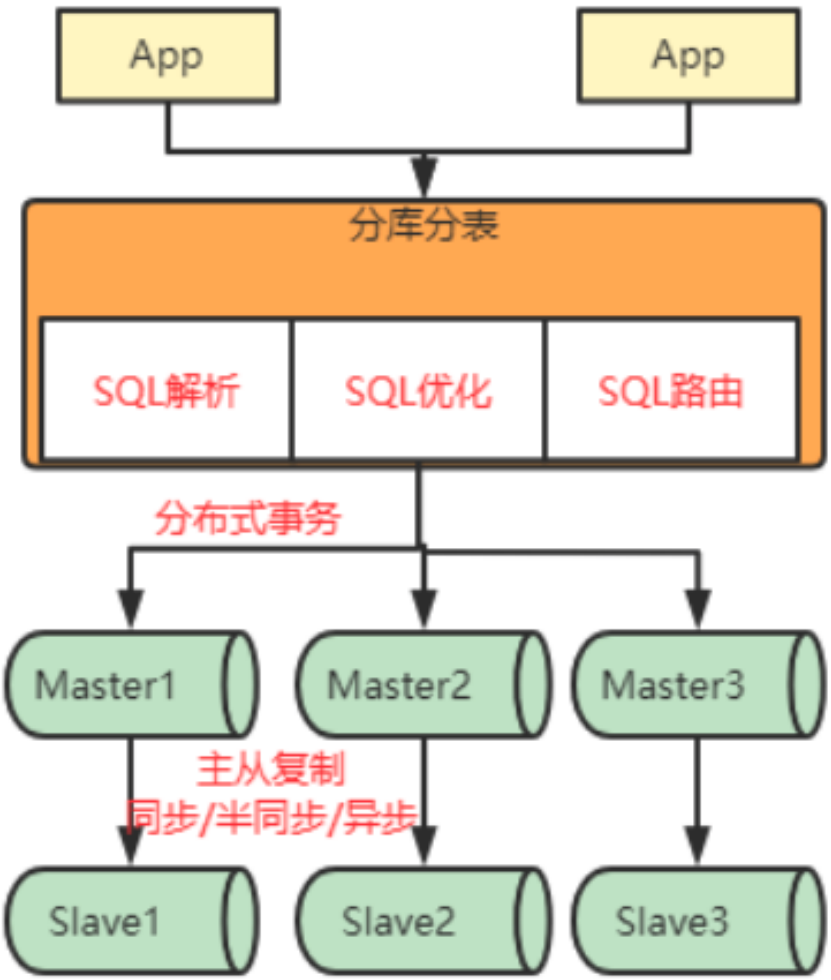
可能你会说，分库分表嘛， 也不是很难。一个库存不下，那就把数据拆分到多个库。一张表数据太多了，就把同一张表的数据拆分到多张。至于怎么做，也不难啊。要操作多个数据库，那么建立多个JDBC连接就行了。要写到多张表，修改下SQL语句就行了。

如果你也这么觉得，那么就大错特错了。**分库分表也并不是字面意义上的将数据分到多个库或者多个表这么简单，他需要的是一系列的分布式解决方案。**

因为数据的特殊性，造成数据库服务其实是几乎没有试错的成本的。在微服务阶段，从单机架构升级到微服务架构是很灵活的，中间很多细节步骤都可以随时做调整。比如对于微服务的接口限流功能，你并不需要一上来就用Sentinel这样复杂的流控工具。一开始不考虑性能，自己进行限流是很容易的事情。然后你可以慢慢尝试用Guava等框架提供的一些简单的流控工具进行零散的接口限流。直到整个应用的负载真正上来了之后，流控的要求更高更复杂了，再开始引入Sentinel，进行统一流控，这都没有问题。这种试错的过程其实是你能够真正用好一项技术的基础。

但是对于数据库就不一样了。当应用中用来存储数据的数据库，从一个单机的数据库服务升级到多个数据库组成的集群服务时，需要考虑的，除了分布式的各种让人摸不着边际的复杂问题外，还要考虑到一个更重要的因素，数据。**数据的安全性甚至比数据库服务本身更重要！**因此，如果你在一开始做分库分表时的方案不太成熟，对数据的规划不是很合理，那么这些问题大概率会随着数据永远沉淀下去，成为日后对分库分表方案进行调整时最大的拦路虎。

所以在决定进行分库分表之前，一定需要提前对于所需要面对的各种问题进行考量。如果你没有考虑清楚数据要如何存储、计算、使用，或者你对于分库分表的各种问题都还没有进行过思考，那么千万不要在真实项目中贸然的进行分库分表。



分库分表，也称为Sharding。其实我觉得，Sharding应该比中文的分库分表更为贴切，他表示将数据拆分到不同的数据片中。由于数据往往是一个应用的基础，随着数据从单体服务拆分到多个数据分片，应用层面也需要面临很多新的问题。比如：

- **主键避重问题**

  在分库分表环境中，由于表中数据同时存在不同数据库中，某个分区数据库生成的ID就无法保证全局唯一。因此需要单独设计全局主键，以避免跨库主键重复问题。

- **数据备份问题**

  随着数据库由单机变为集群，整体服务的稳定性也会随之降低。如何保证集群在各个服务不稳定的情况下，依然保持整体服务稳定就是数据库集群需要面对的重要问题。而对于数据库，还需要对数据安全性做更多的考量。

- **数据迁移问题**

  当数据库集群需要进行扩缩容时，集群中的数据也需要随着服务进行迁移。如何在不影响业务稳定性的情况下进行数据迁移也是数据库集群化后需要考虑的问题。

- **分布式事务问题**

  原本单机数据库有很好的事务机制能够帮我们保证数据一致性。但是分库分表后，由于数据分布在不同库甚至不同服务器，不可避免会带来分布式事务问题。

- **SQL路由问题**

  数据被拆分到多个分散的数据库服务当中，每个数据库服务只能保存一部分的数据。这时，在执行SQL语句检索数据时，如何快速定位到目标数据所在的数据库服务，并将SQL语句转到对应的数据库服务中执行，也是提升检索效率必须要考虑的问题。

- **跨节点查询，归并问题**

  跨节点进行查询时，每个分散的数据库中只能查出一部分的数据，这时要对整体结果进行归并时，就会变得非常复杂。比如常见的limit、order by等操作。

在实际项目中，遇到的问题还会更多。从这里可以看出，Sharding其实是一个很复杂的问题，往往很难通过项目定制的方式整体解决。因此，大部分情况下，都是通过第三方的服务来解决Sharding的问题。比如像TiDB、ClickHouse、Hadoop这一类的NewSQL产品，大部分情况下是将数据问题整体封装到一起，从而提供Sharding方案。但是这些产品毕竟太重了。更灵活的方式还是使用传统数据库，通过软件层面来解决多个数据库之间的数据问题。这也诞生了很多的产品，比如早前的MyCat，还有后面我们要学习的ShardingSphere等。

另外，关于何时要开始考虑分库分表呢？当然是数据太大了，数据库服务器压力太大了就要进行分库分表。但是这其实是没有一个具体的标准的，需要根据项目情况进行灵活设计。业界目前唯一比较值得参考的详细标准，是阿里公开的开发手册中提到的，**建议预估三年内，单表数据超过500W，或者单表数据大小超过2G**，就需要考虑分库分表。

> 有道云笔记链接：【有道云笔记】一、道听途说：ShardingSphere产品介绍.md
> https://note.youdao.com/s/llk99xJv