为什么说TiDB在线扩容对业务几乎没有影响

数据源的TiDB学习之路 发表于 2024-02-01

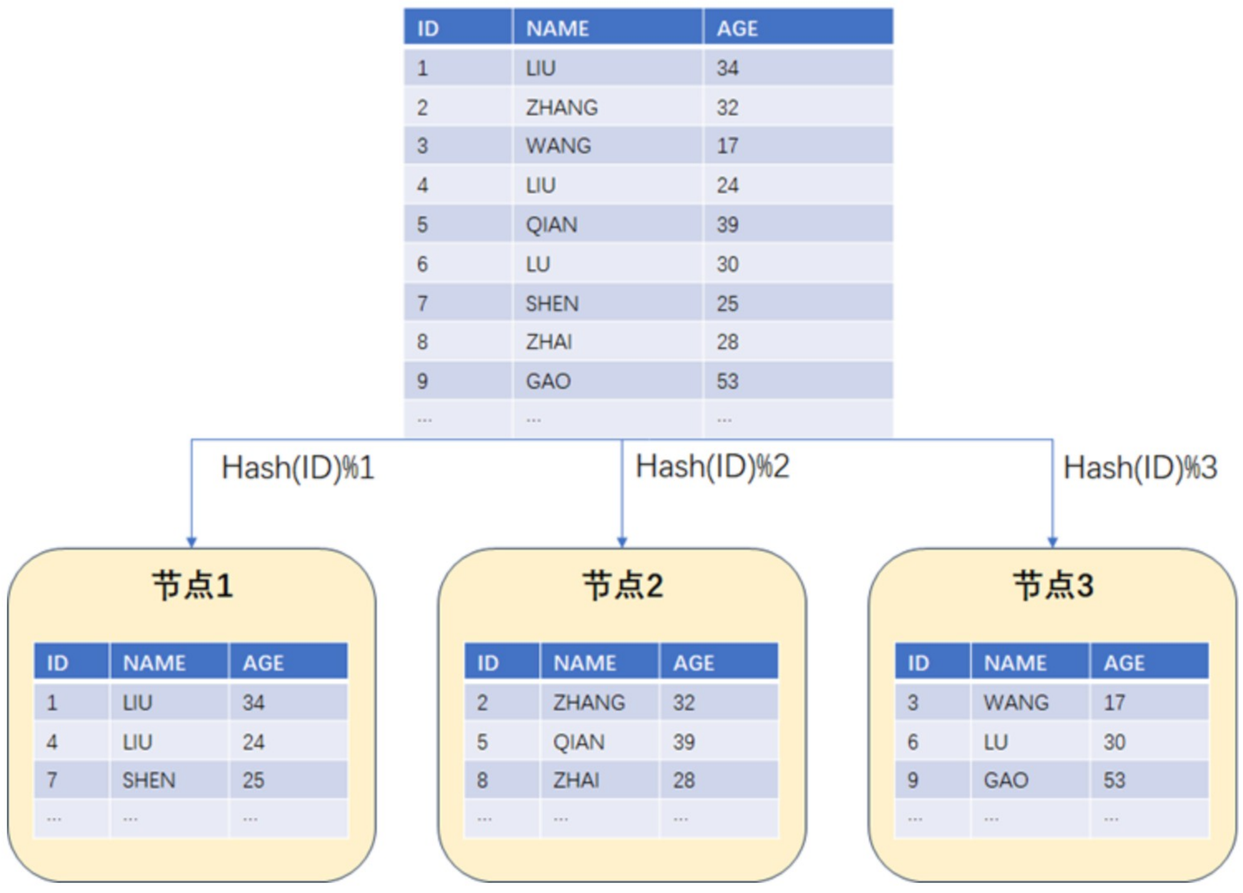
原创 # TiDB 底层架构 # TiKV 底层架构 # 数据库架构选型

当前的数据库种类繁多，墨天轮当前统计的所有国产数据库已经有**290个**，其中属于关系型数据库的有**166个**。关系型数据库从部署架构上又可以分为集中式（典型代表为达梦DM8、金仓KES）、分库分表（典型代表为中兴GoldenDB、腾讯TDSQL）以及原生分布式架构（典型代表为PingCAP TiDB、阿里OceanBase）。

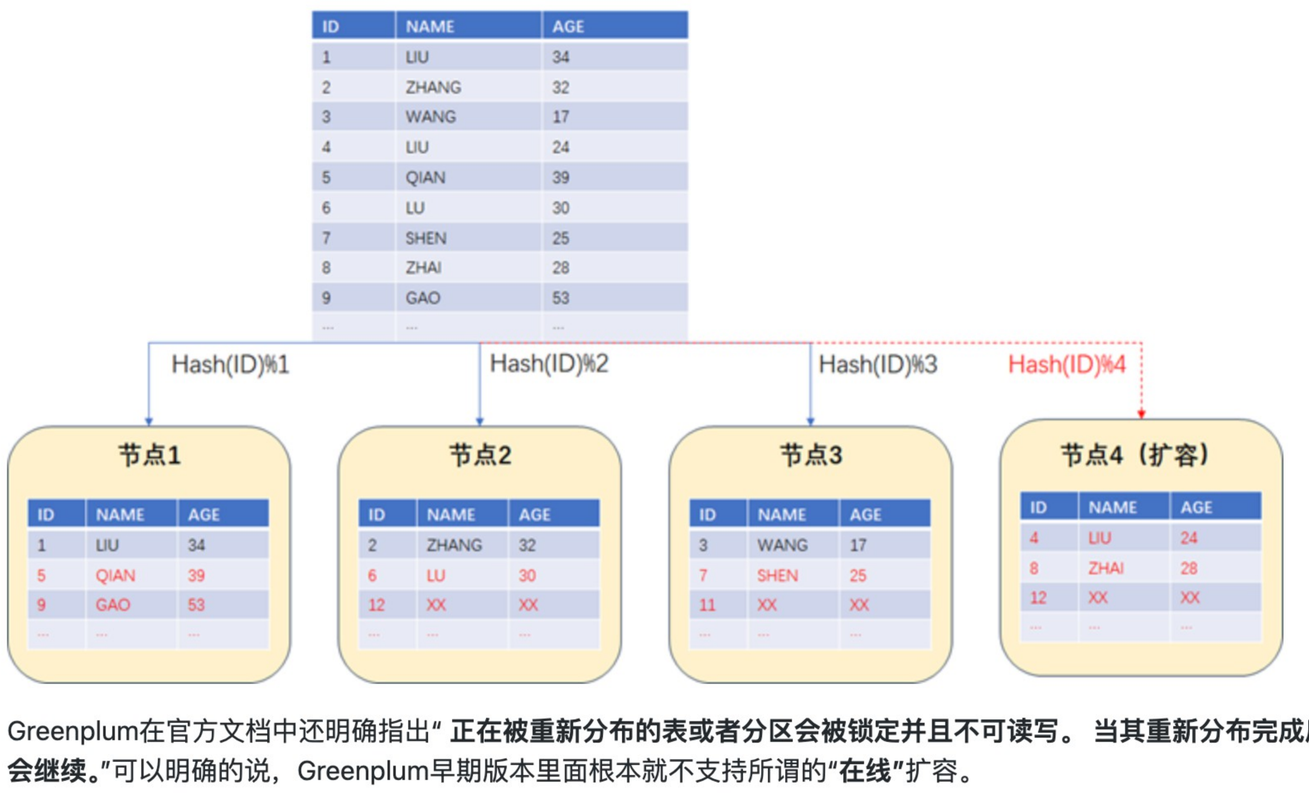
昨天和别人交流PingCAP TiDB时，这位同学对**“TiDB在线扩容对业务几乎没有影响”**这一点表示不太理解，惊讶TiDB到底是怎么做到的。。细聊下来，发现这位同学是一位主要负责集中式和早期分布式架构数据库的DBA人员，比较熟悉Oracle、Greenplum。于是我有点理解他的惊讶了，因为Oracle和Greenplum我也是有一点点经验，本文简单针对一般分布式数据库和TiDB在扩容机制上谈一点个人的理解。

一. 一般分布式数据库在线扩容是怎么做的?

集中式数据库因为其架构本身的限制，一般来说想要实现在线扩容是比较困难的，这里暂且不予讨论，我们主要了解一下一般分布式数据库的扩容是如何进行的。不管是Greenplum这种MPP数据库，还是其它的分库分表数据库，为了实现数据的均衡分布，通常需要在表上定义相关的分布键。通过分布键，再结合哈希算法，可以把数据哈希散列到不同的数据节点中，类似于**hash（key）% N（key代表分布键，N代表数据节点编号）**。举个例子，假如一个分布式数据库有3个数据节点，表的分布键为ID（ID是一个递增序列），那么基于哈希算法散列后数据的分布大致如下图所示：

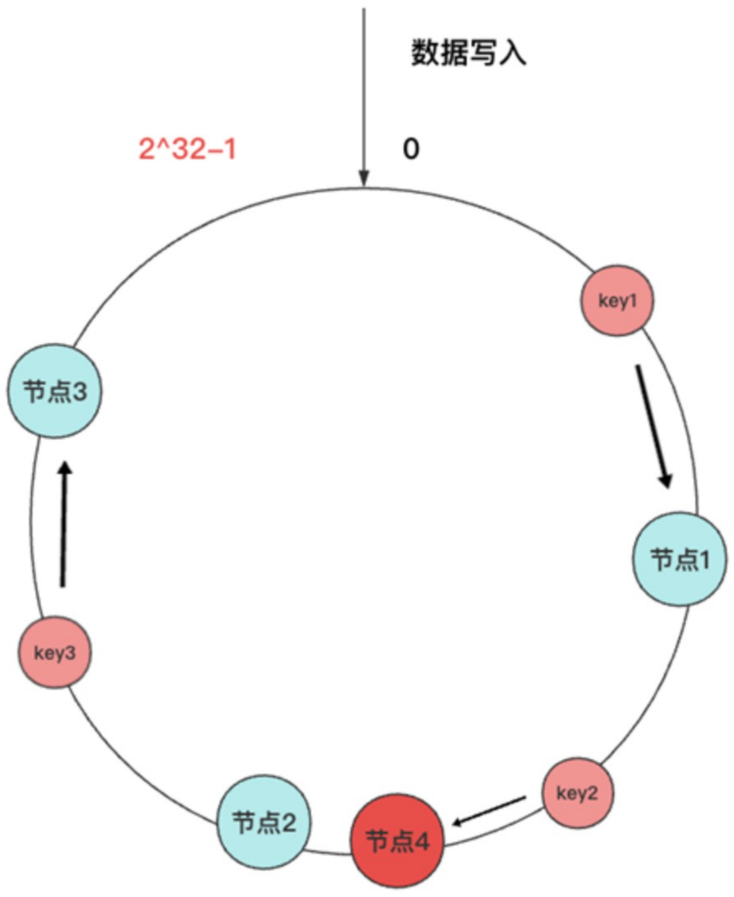


现在我们需要扩容一个节点，从原来的3节点扩容到4节点。为了保证原来哈希散列结果的一致性数据需要重新平衡，平衡后的数据分布应该如下面图中所示。可以发现，这个时候大部分的数据基本都搬迁了一遍。先不说数据的迁移是否对业务造成阻塞，光是这现有的大面积数据均衡足以导致整个系统的IO消耗极高，**严重影响整个系统的可用性**。



Greenplum在官方文档中还明确指出“ **正在被重新分布的表或者分区会被锁定并且不可读写。 当其重新分布完成后，常规操作才会继续。**”可以明确的说，Greenplum早期版本里面根本就不支持所谓的**“在线”**扩容。

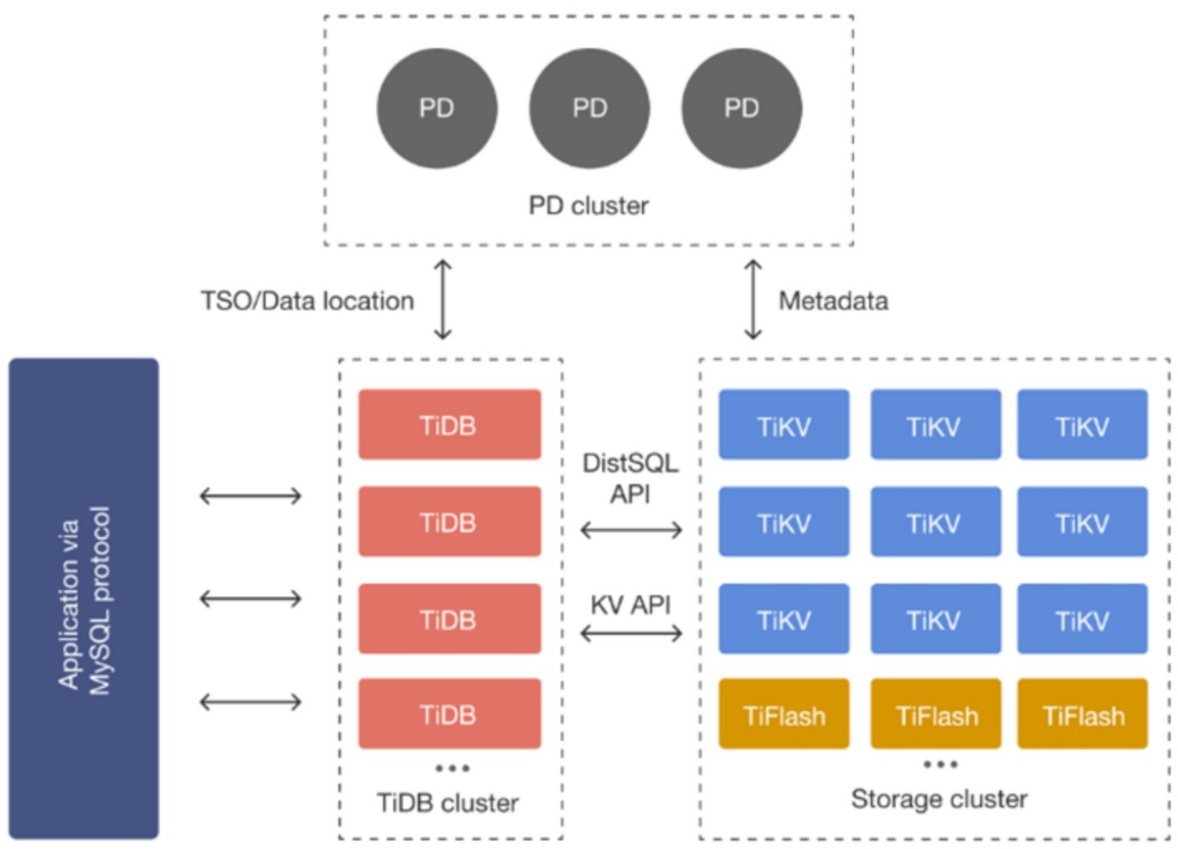
时代在进步，数据库技术也在进步。为了尽可能实现在线扩容的能力，Greenplum数据库包括其他的分库分表数据库开始引入一些新的算法来优化此事。**一致性哈希算法**开始被普遍应用，它与传统哈希算法最主要的不同是**不再使用节点编号来进行散列**，而是使用2^32这样一个固定值做取模运算。一致性哈希算法将表中的数据和节点编号映射到一个圆环上，当增加节点时影响的数据范围只是圆环上的一小段数据范围。比如下图中增加节点4，影响的数据只有节点1到节点4之间的这部分数据。



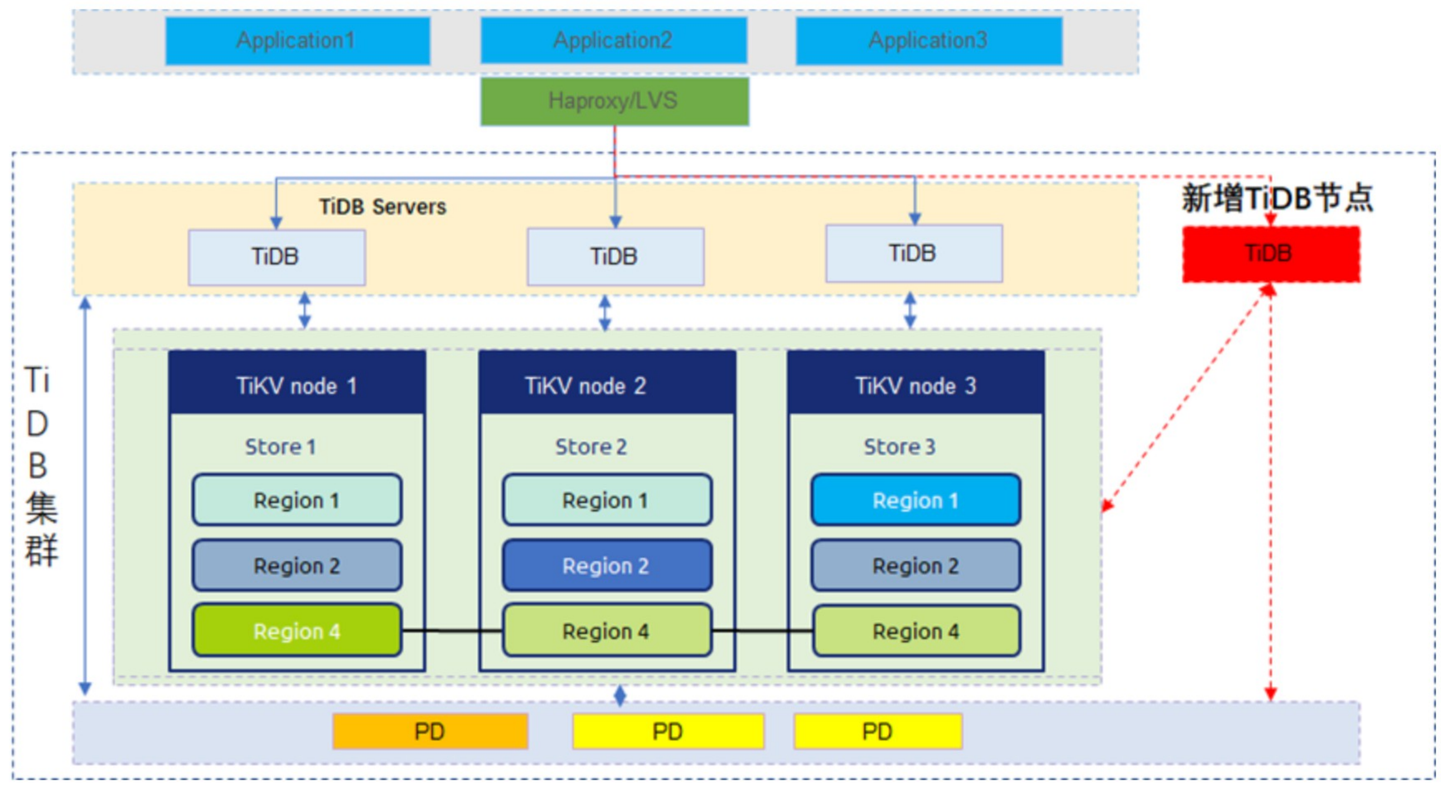
一致性哈希算法解决了数据重分布时大量数据搬迁的问题，减少了数据搬迁时的网络IO和磁盘IO。不过要真正实现不影响业务，还需要改进数据重分布内部的机制，比如**重分布时锁表等**问题。

二． TiDB的扩容是怎么做的以及为什么它几乎不影响业务?

TiDB的扩容机制离不开TiDB整体的架构实现。作为一个存算分离的原生分布式架构，TiDB集群主要由三大模块构成：用于集群元数据管理及集群调度的PD、用于接收外部请求并解析编译执行SQL的计算引擎TiDB Server以及用于数据存储以及多副本数据一致性保证的存储引擎TiKV/TiFlash。



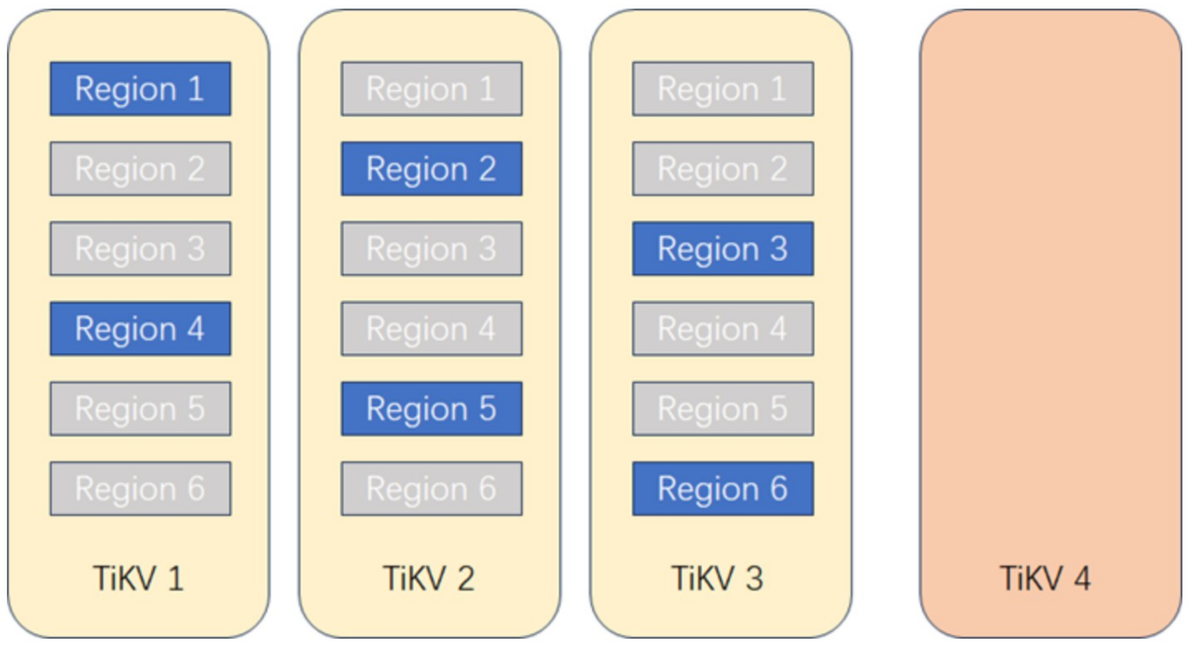
基于存算分离的架构，TiDB可以单独进行计算层扩容和存储层扩容。计算层的扩容相对简单，因为TiDB Server本身是**无状态**的。TiDB Server节点**不持久化数据**，每个节点也是**完全对等**的，当TiDB Server计算资源不够了，只需要增加TiDB Server节点，然后修改上层的负载均衡组件将客户端连接均衡分发到新的TiDB Server节点即可（目前大多数负载均衡组件都支持动态修改配置）。因此，**计算节点的扩容完全不会影响现有的业务。**



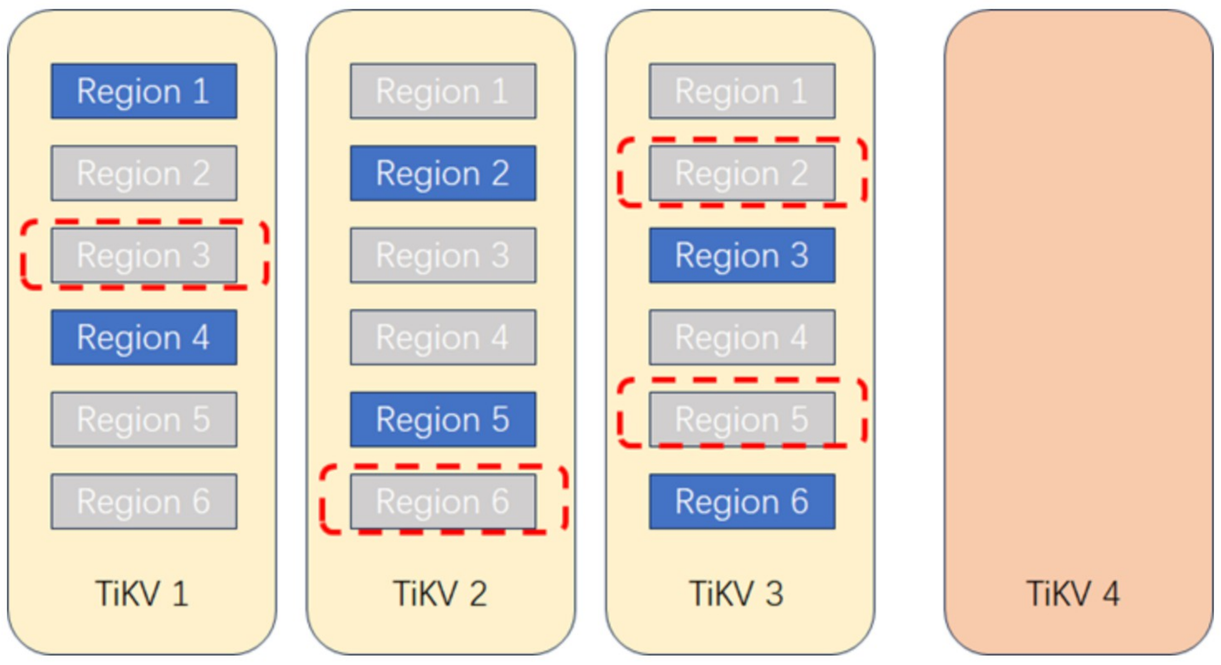
针对存储节点，TiKV的扩容与一般分布式数据库的扩容机制是完全不同的，这主要因为TiKV是一种**基于Multi Raft协议的分布式存储引擎**，而不是像Greenplum或分库分表那种底层是多个MySQL或PG的单机数据库。

假如某集群要从3个TiKV节点扩容到4个TiKV节点，扩容步骤大致可以概括如下：

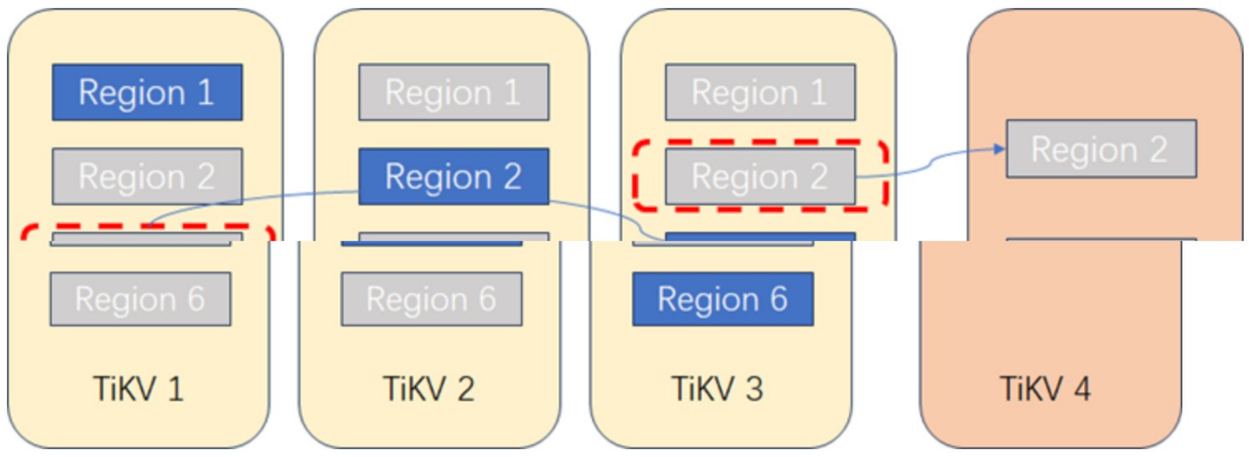
1. **扩容TiKV节点。**集群增加一个TiKV 4节点，此时TiKV 4上没有任何Region。PD节点识别到新的TiKV节点启动负载调度机制，计算哪些Region需要迁移到TiKV 4 。



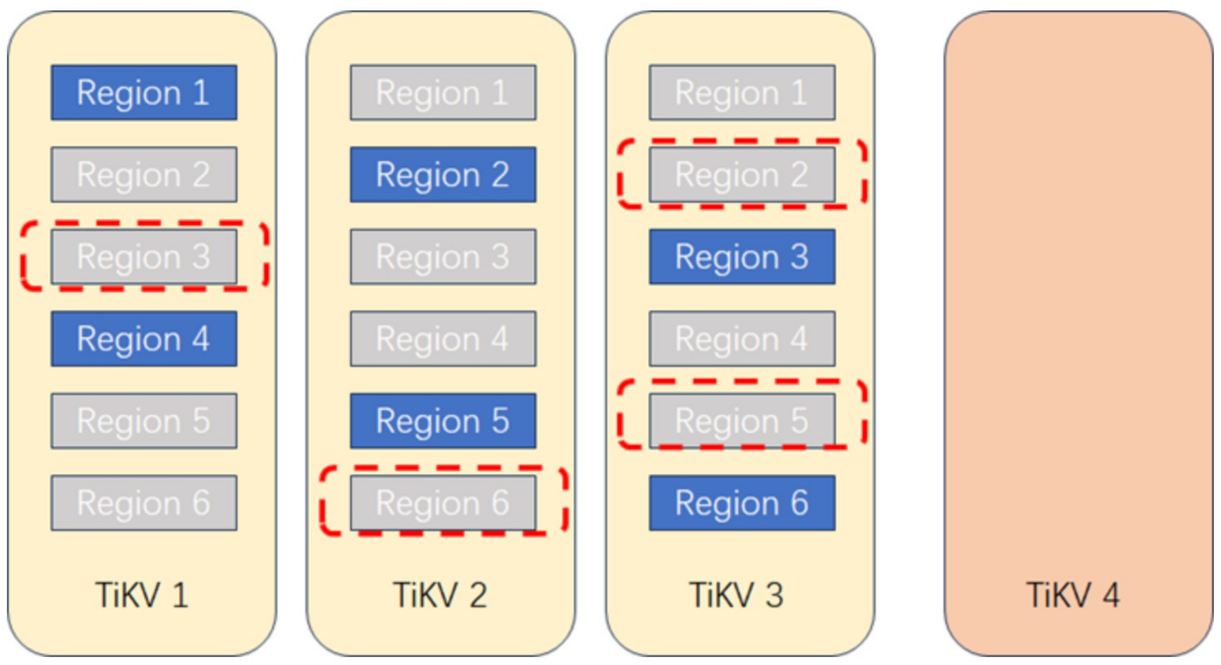
2. **调度算法确定迁移Region。**PD节点根据调度机制，确定将哪些Region副本迁移到TiKV 4上（假如开始3个节点上各有6个Region，平均到4个节点后每个节点的Region数为18/4=4~5个副本）。PD对TiKV1~3上Region对应的Leader副本发起复制指令。



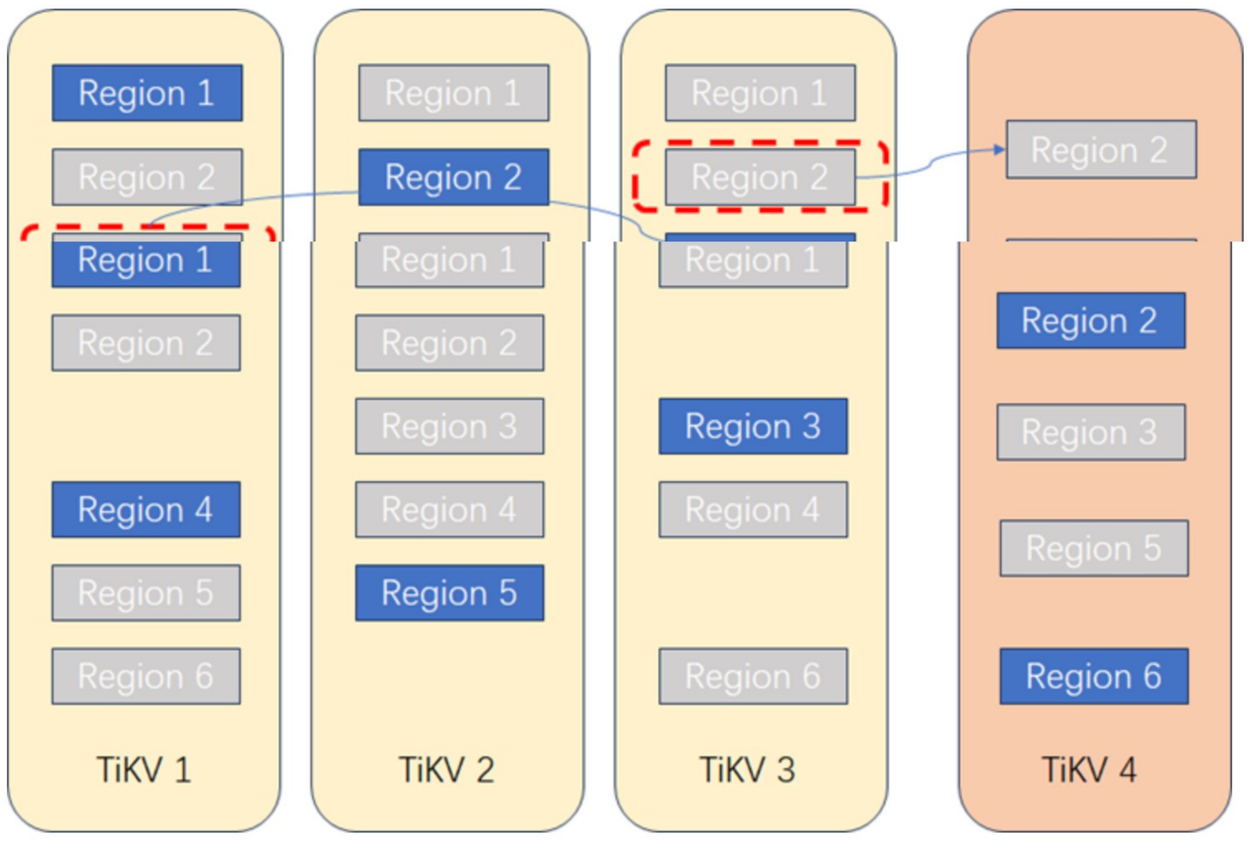
3. **复制Region到新节点。**在TiKV上创建要复制的Region的副本，通过Raft机制开始复制数据。此过程中应用读写访问不受影响，不过因复制过程产生的IO消耗可能会对性能产生一点影响，不过TiDB本身提供了流控，可以动态调整复制的速度。



2. **调度算法确定迁移Region。**PD节点根据调度机制，确定将哪些Region副本迁移到TiKV 4上（假如开始3个节点上各有6个Region，平均到4个节点后每个节点的Region数为18/4=4~5个副本）。PD对TiKV1~3上Region对应的Leader副本发起复制指令。



3. **复制Region到新节点。**在TiKV上创建要复制的Region的副本，通过Raft机制开始复制数据。此过程中应用读写访问不受影响，不过因复制过程产生的IO消耗可能会对性能产生一点影响，不过TiDB本身提供了流控，可以动态调整复制的速度。



上述步骤简单理解下来就是说，TiKV的扩容是一种**先生成副本再迁移Leader**的一个过程，扩容对业务有影响的地方主要在于生成副本产生的IO消耗以及Leader切换的影响。对于前者，数据库有流控机制可以保证对业务几乎没有影响；对于后者，一方面Leader的切换本身时间非常短，另一方面当TiDB意识到Region迁移后也能够通过内部重试保证前端业务的正常执行。因此，**存储节点的扩容也几乎不会影响现有的业务。**

评论

T 添加评论 评 论

暂无评论

< 1 >