白皮书

Redis CRDTs 揭秘 (无冲突复制数据类型)

目录

介绍与引入

在最初的CAP定理提出的十二年后，EricBrewer撰写了一篇关于CAP规则如何演进的**精彩的文章**。那篇文章总结说到，通过使用CRDTs（无冲突复制数据类型），可以在C、A和P之间建立一种新的平衡——也就是通常所说的"**强最终一致性**"。

Redis Enterprise使用多主复制架构来实现CRDTs。Redis CRDTs 具备许多优势：

- 全球分布式应用的亚毫秒级延迟读写：Redis CRDTs创建了一个跨越多个数据中心的全球跨度数据库，为每个数据中心的应用程序提供低延迟的读写能力。
- Active-Active多区域部署，实现持续可用性：通过Redis CRDTs，应用程序可以在远程地区的多个数据中心之间进行持续的读写操作，即使某些数据中心完全不可用或者数据中心之间的网络连接断开，读写可用性也能保障是连续的。
- 智能的自动冲突解决简化了开发流程：使用内置的Redis类型和命令，结合Redis CRDTs，您可以简化复杂的、关键任务的应用程序的开发过程，同时自动处理并发读写操作中的冲突。

让我们来深入了解Redis CRDT是如何提供高性能、高可用性和智能冲突解决的。

Redis CRDT 揭秘

Redis Enterprise 概览

Redis Enterprise是一个分布式数据库平台，由部署在数据中心内或跨越本地可用区的相同节点组成。每个节点包含构成管理路径的服务（如下图中的蓝色层）和数据访问路径的服务（如下图中的红色层）。

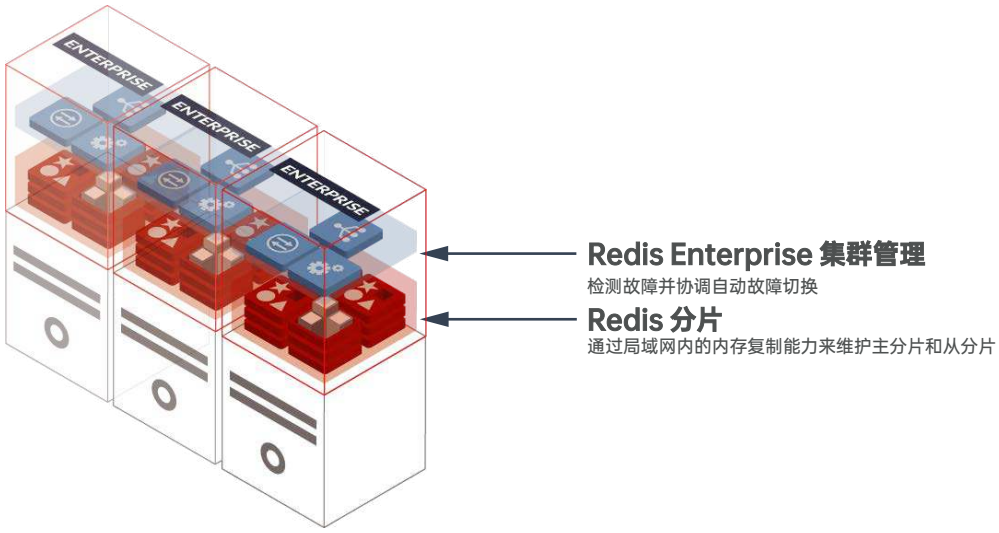

Figure 1. Redis Enterprise节点，蓝色图块代表管理路径，红色图块代表数据访问路径，且其中的Redis实例作为单个分片存在。

- 管理路径包括集群管理器、代理和用于程序化管理的、安全的REST API/UI。集群管理器负责协调集群和数据库分片的部署，以及检测和减轻故障。代理通过智能连接管理帮助扩展连接。管理路径协调创建和持续管理Redis CRDTs。
- 数据访问路径由多个作为主节点和从节点的Redis分片组成。客户端在主分片上执行数据操作。主分片通过内存复制技术来维护与从分片的一致性，以防止主分片不可访问时发生故障。单个Redis Enterprise集群中可以有多个数据库，每个数据库可以包含多个分片。

Redis CRDT 架构

Redis CRDTs在Redis Enterprise中基于跨越多个集群的全局数据库进行实现。这些全局跨越的数据库被称为“无冲突复制数据库”或“CRDBs”。

全局CRDB由每个参与的集群中的本地数据库组成，每个本地数据库都可以被称为“成员CRDB”。CRDB之间建立了双向复制。需要注意的是，从应用程序的角度来看，应用程序连接用于读写数据的成员CRDB的行为就像连接到普通的Redis数据库一样，对于应用程序而言这是透明的。

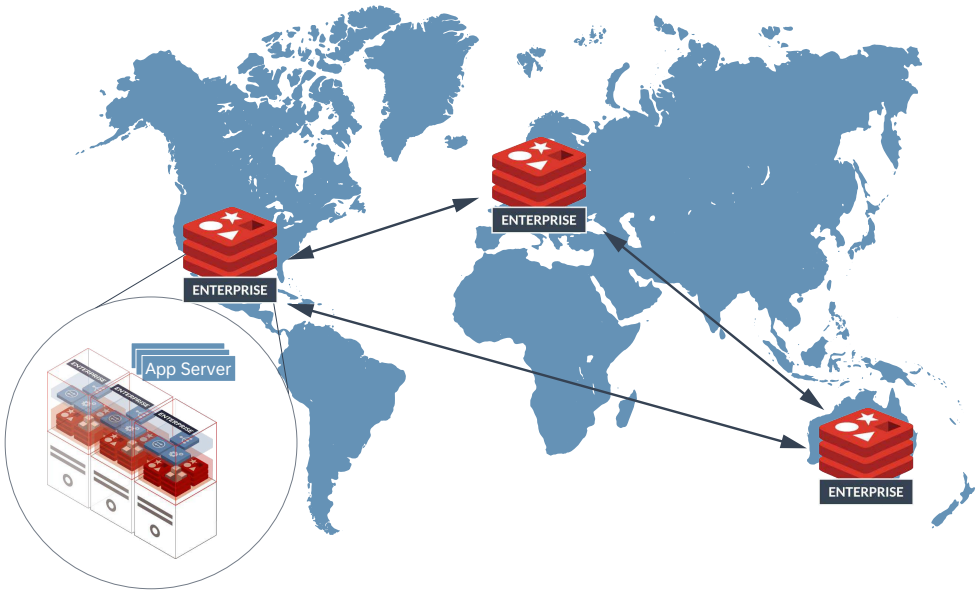

Figure 2. 双向复制能力实现了地理分布式的Active-Active读写拓扑结构。

通过CRDB实现高性能读写

本地部署的应用程序直接连接到参与到集群中的成员CRDB。通过这样，Redis应用程序与CRDB之间的延迟可达到亚毫秒级。应用程序可以同时在成员CRDB之间读写相同的键。通过双向复制，对每个成员CRDB所做的更改会被复制到其他成员CRDB，并在出现并发写操作时，根据每种数据类型定义的规则对冲突进行解决。

双向复制与CRDB Syncer

CRDB Syncer是一个协调所有成员CRDB之间基于广域网的复制的进程。Syncer进程位于参与CRDB部署的所有集群中，并负责数据的初始同步和持续同步。CRDB Syncer与其他成员CRDB建立并发连接，并从从节点读取更改。这样做可以确保只有经过本地复制且持久化的数据才会被复制到其他成员CRDB。复制的更改以流式传输的方式在成员CRDB之间通信，并被应用到本地成员CRDB上。在传输过程中，数据会进行压缩，这可以在通过广域网跨越长距离时节省带宽。

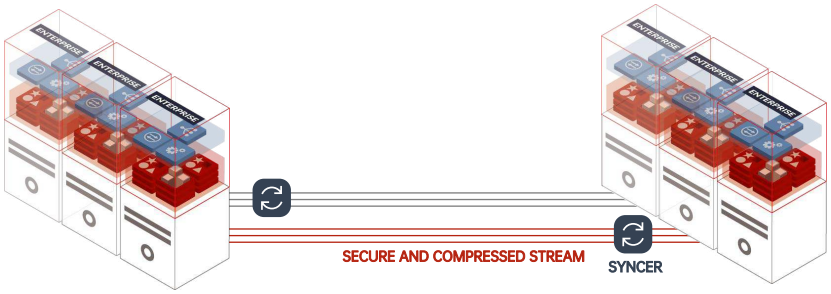

Figure 3. 基于WAN的CRDB同步器架构

CRDB Syncer可以自动处理由于网络故障或源数据库和目标数据库的拓扑变化而引起的中断。在中断期间，Syncer会持续轮询，并在合适的时机恢复复制。在某些情况下，恢复复制可能需要对数据过期的成员CRDB进行完全同步复制。

Active-Active 多区域部署实现不间断可用性

数据中心可能会遭遇短暂或长时间的故障，或仅仅是需要进行维护。与Redis Enterprise在节点、机架或区域故障时所执行的自动故障转移不同，使用Active-Active部署的地理故障转移不会自动由Redis Enterprise执行。应用程序可以通过将其用户请求重定向到另一个地理位置的参与集群之一来方便地进行地理故障转移。

地理故障转移是由CRDT（Conflict-free Replicated Data Types）优雅解决的问题。如果没有CRDT，处理地理故障转移将变得复杂，并可能导致更新丢失。以下是一个示例：

假设一下，我们有一个在Redis Set中维护的购物车，并且随着时间的推移发生了以下事件：

- 西海岸的用户在键“cart1”下维护一个购物车。在时间“t1”时，购物车被更新，即用户添加了一个名为“costume”的新产品。

- 在时间“t2”之前，当购物车“cart1”的更新尚未被复制同步到东海岸数据中心时，西海岸数据中心发生了故障。然后，用户请求被引导到东海岸数据中心以实现持续的可用性。

- 在时间“t3”，东海岸数据中心尚未在购物车“cart1”中收到物品“costume”。然而，用户将一个名为“mask”的新产品添加到购物车中。此时，西海岸数据中心的购物车中包含产品“costume”，而东海岸数据中心的购物车中包含产品“mask”。

时间	西海岸数据中心	东海岸数据中心
t1	SADD cart1 “costume”	
t2	西海岸数据中心失效 - 同步失效	
t3		SADD cart1 “mask”
t4	西海岸数据中心恢复 - 重新同步	
t5	SMEMBERS cart1 “costume” “mask”	SMEMBERS cart1 “costume” “mask”

- 在时间“t4”，西海岸数据中心恢复并且双向复制重新开始。如果您正在使用使用LWW（最后写入者优先）等机制进行冲突解决的数据库，购物车只能包含“costume”或“mask”中的一个，因为对“cart1”购物车键的更新只会保留一个。结果将是购物车的更新丢失！CRDTs确保购物车中没有更新丢失。通过Redis CRDTs，使用了Redis SET数据类型的SADD方法，购物车可以正确地反映了用户添加的所有物品。

智能的自动冲突解决功能简化开发过程

CRDTs为处理冲突写提供了一个数学模型。CRDTs的目标是为常见的使用案例提供明确定义的冲突解决行为。结合Redis的类型和命令，CRDTs在冲突解决过程中可以检测到"开发者意图"。例如，当使用INCR等方法时，Redis CRDTs会以一种允许您构建分布式计数器的方式进行操作。当使用SET类型的SADD或SREM方法时，Redis CRDTs使用CRDTs中定义的"添加优先，Observed-Remove Set"行为。

总体而言，Redis CRDTs允许用户专注于开发他们的业务逻辑，而无需编写自定义的冲突解决逻辑来处理各种并发写入冲突场景。开发人员可以选择适合其应用程序的数据类型和冲突解决规则，Redis CRDTs会根据每种类型和命令的明确定义规则自动解决冲突。为了实现这个效果，Redis CRDTs会为每种类型保留额外的元数据。这些额外的元数据稍后在双向复制中用于同步所有参与的集群。您可以联系我们获取详细的开发人员指南以及规定Redis类型与CRDBs冲突解决的规则和行为。

现在就开始尝试Redis CRDTs

开始使用Redis Enterprise和Redis CRDTs非常简单。您也可以联系我们，获得技术支持。


redis