

腾讯

分布式数据库

DCDB 白皮书

V2.0



腾讯云

【版权声明】

©2013-2018 腾讯云 版权所有

本文档著作权归腾讯云单独所有，未经腾讯云事先书面许可，任何主体不得以任何形式复制、修改、抄袭、传播全部或部分本文档内容。

【商标声明】

及其它腾讯云服务相关的商标均为腾讯云计算（北京）有限责任公司及其关联公司所有。本文档涉及的第三方主体的商标，依法由权利人所有。

【服务声明】

本文档旨在向客户介绍腾讯云全部或部分产品、服务的当时的整体概况，部分产品、服务的内容可能有所调整。您所购买的腾讯云产品、服务的种类、服务标准等应由您和腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或模式的承诺或保证。

目 录

1	前言.....	4
2	通用约定.....	4
3	产品介绍.....	5
3.1	分布式数据库概述.....	5
3.2	分布式数据库相关概念.....	5
4	产品架构.....	7
4.1	分布式数据库系统架构.....	7
4.2	分库与分表.....	9
4.2.1	分库	9
4.2.2	分表	9
4.2.3	DCDB 分表原理简介	10
4.2.4	选择拆分键（shardkey）	12
4.2.5	同表不同拆分建的场景	14
4.2.6	拆分键选择的限制	14
4.3	分布式数据库技术特征.....	16
4.3.1	自动水平拆分	16
4.3.2	高度兼容 MySQL 语法.....	16
4.3.3	补齐分布式架构的不足	16
4.3.4	不停机弹性扩展	16
4.3.5	基于数据强一致的高可用	16
4.3.6	超高性能	17
4.3.7	易于使用的托管部署	17
4.3.8	OLAP 扩展	17
5	产品优势	17
5.1	高可用与强同步	17
5.2	性能与容量线性增长.....	19
5.3	逻辑表.....	19

5.4 实时监控告警	20
5.5 高性能分布式事务	21
5.6 全局唯一数字序列	21
5.7 多租户与独享集群	22
5.8 弹性扩展	23
5.9 容灾与恢复	24
6 应用场景	25
6.1 电子商务类应用	25
6.2 金融类应用	25
6.3 IoT 类应用	25
6.4 游戏应用	26
6.5 成为去 O 的中坚力量	26
6.6 分支业务聚合到总部	26

1 前言

本文档针对腾讯公有云的 DCDB 分布式数据库的市场定位及特点，帮助用户从产品架构、产品优势、产品功能、应用场景等方面全面了解本产品。

本文档主要适用于以下工程师：

- 技术支持工程师
- 维护工程师

2 通用约定

表 1: 格式约定

格式	说明	样例
	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	 危险：重置操作将丢失用户配置数据。
	该类警示信息可能导致系统重大变更甚至故障，或者导致人身伤害等结果。	 警告：重启操作将导致业务中断，恢复业务所需时间约10分钟。
	用于警示信息、补充说明等，是用户必须了解的内容。	 注意：导出的数据中包含敏感信息，请妥善保管。
	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	 说明：您也可以通过按 Ctrl + A 选中全部文件。
>	多级菜单递进。	设置 > 网络 > 设置网络类型
粗体	表示按键、菜单、页面名称等UI元素。	单击 确定 。
courier字体	命令。	执行 <code>cd /d C:/windows</code> 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	<code>bae log list --instanceid <i>Instance_ID</i></code>

[]或者[a b]	表示可选项，至多选择一个。	ipconfig [-all]-t]
{ }或者{a b}	表示必选项，至多选择一个。	swich {stand slave}

3 产品介绍

3.1 分布式数据库概述

腾讯分布式数据库（Distributed Cloud Database，DCDB）是支持自动水平拆分的高性能数据库服务——即业务感受完整的逻辑表，而数据却均匀的拆分到多个物理分片中，可以有效解决超大并发、超高性能、超大容量的 OLTP 类场景；DCDB 的每个分片默认采用主从高可用架构，提供弹性扩展、备份、恢复、监控等全套解决方案，为您有效解决业务快速发展时的数据库性能瓶颈，让您能更加专注于业务发展。

当前，腾讯分布式数据库 DCDB 支持 MySQL 协议和语法。

 **注意：**您可能注意到部分文档中，TDSQL 也代指 DCDB；TDSQL 是腾讯云数据库团队维护的金融级分布式架构和 MySQL 内核分支的统称，腾讯 90% 的金融、计费、交易类业务核心系统承载在 TDSQL 中。目前已应用于众多政府、银行、保险、制造业、物流、电商等用户的核心系统中。TDSQL 可以提供公有云、专有云两种部署方案，可以分配 CDB（关系型数据库）实例、DCDB（分布式数据库）实例、ADB（分析性数据库）实例，即 TDSQL 是腾讯公有云 MySQL 协议的 CDB、DCDB、ADB 的集合；并提供数据复制（即强同步）、线程池、热点更新、内核优化等能力；并提供事前、事中、事后的全维度安全方案；以及获得多项国际和国家认证。

3.2 分布式数据库相关概念

- **实例：**用户实际使用的一个最小单位的数据库服务集合；一般来说，用户使用数据库可独立的享有数据库实例 IP、端口和账号；实例之间资源（权限、性能、数据空间）完全隔离；实

例可能仅存于 1 台服务器上，也可能在不同的物理服务器上。一个数据库实例可以包含一个或多个用户创建的数据库等。

- **分片 (Sharding)** : 在分布式数据库中，由数据库引擎组成的物理实例，通常多个分片组成一个逻辑上的实例。
- **物理节点组 (SET)** : 搭建数据集群的若干物理节点，通常这些节点基于数据库主从协议联结成若干组，这些物理节点组的统称叫做 SET。需要注意时，如果没有采用多租户（虚拟化技术），通常一个分片就等于一个 SET，如果采用多租户（虚拟化）技术，物理节点组（SET）中可能有多个分片。
- **数据库引擎、版本** : 每个数据库实例运行一个数据库引擎，数据库引擎是用于存储、处理和保护数据的核心服务，通常我们说的 MySQL、SQLServer、Oracle 就是引擎的叫法；每个数据库引擎又包括不同的软件版本，不同的数据库引擎版本都有自己支持的功能和特性。但请注意，innodb 等在本文中被叫做存储引擎，与数据库引擎并非相同概念。



说明 :DCDB 当前支持的 MySQL 协议默认选择 Percona 或 MariaDB 分支(避免 Oracle 授权影响)。

- **集群** : 一个独立网络区域的 DCDB 服务器集合集群；一般来说，先有集群才能在集群上分配实例。而集群中的设备互为冗余，集群通常包括主数据库服务器、多个备份数据库服务器、网络设备、数据备份集群等。通常多个集群可以共享一套管理系统。
- **实例规格** : 是定义数据库的使用大小、性能的一种综合指标。规格使用计算能力、内存、存储容量等多种指标进行定义。由于云计算的弹性能力，实例规格实际上是可以按需伸缩的，用户可以选择在合适时机选择对规格进行扩容或缩小，以保障数据库引擎有足够的空间来写入内容和日志。
- **DDL** :数据库模式定义语言(Data Definition Language) ,主要的命令有 CREATE、ALTER、

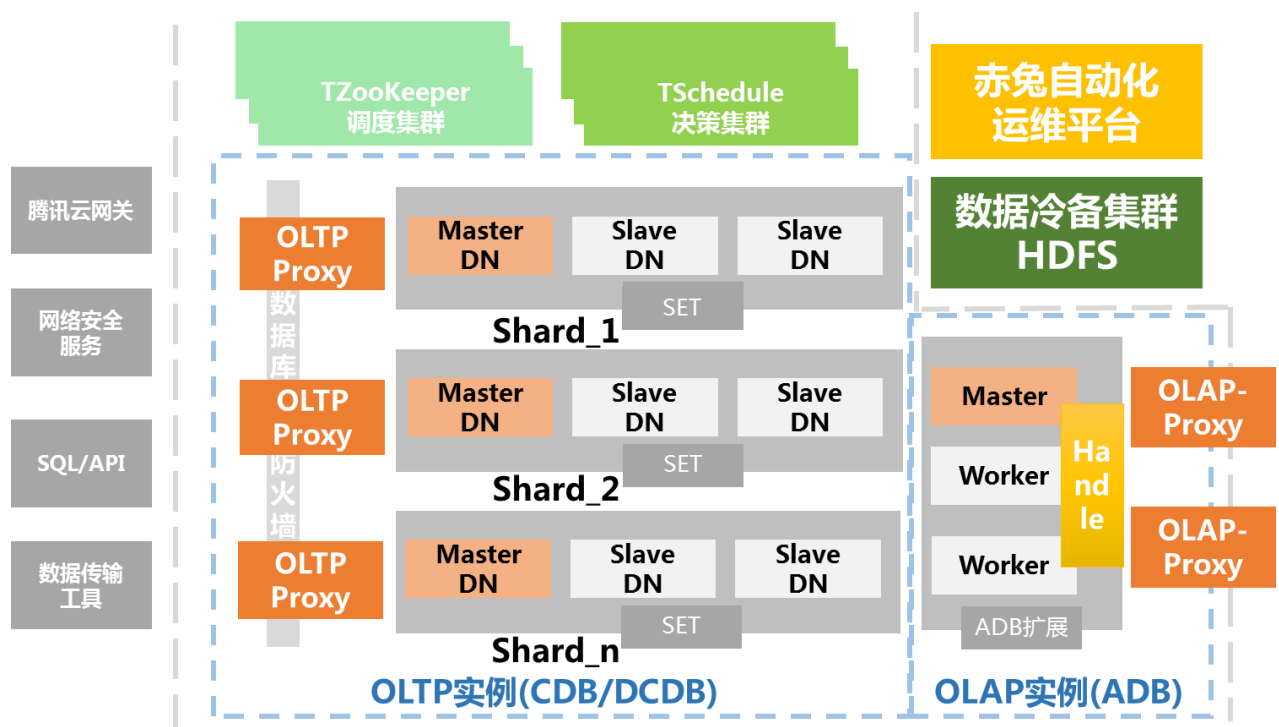
DROP 等。

- **DML**：数据库操纵语言 (Data Manipulation Language)，命令是 SELECT、UPDATE、INSERT、DELETE。
- **OLTP**：联机事务处理 (On-Line Transaction Processing)，是传统的关系型数据库的主要应用，主要是基本的、日常的事务处理，例如银行交易。
- **OLAP**：联机分析处理 (On-Line Analytical Processing)，是数据仓库系统的主要应用，支持复杂的分析操作，侧重决策支持，并且提供直观易懂的查询结果。

4 产品架构

4.1 分布式数据库系统架构

DCDB 采用集群架构，一套完整的腾讯分布式数据库 DCDB 系统由若干个分布架构子系统组成的集群，整个集群架构简图如下：



其中，DCDB 最核心的五个主要模块是：调度集群 (Tschedule)、物理节点组 (SET) 和接入网关集群 (OLTP-Proxy)，决策集群 (TzooKeeper)，赤兔自动化运维平台组成

- **物理节点组 (SET)**: 由 MySQL、监控和信息采集 (TAgent) 组成, 通常情况下:
 1. SET 默认采用一主多从架构, 通常部署在跨机架物理服务器中;
 2. 每个节点 (DataNode) 都部署 TAgent, 并实时向决策集群上报, 提供决策依据;
- **调度集群 (TScheduler)**: 帮助 DBA 或者数据库用户自动调度和运行各种类型的作业, 比如数据库备份、收集监控、生成各种报表或者执行业务流程等等, DCDB 把 Schedule、zookeeper、OSS (运营支撑系统) 结合起来通过时间窗口激活指定的资源计划, 完成数据库在资源管理和作业调度上的各种复杂需求。
- **决策集群 (TzooKeeper)**: 它是 DCDB 提供配置维护、选举决策、路由同步等, 并能支撑数据库节点组 (分片) 的创建、删除、替换等工作, 并统一下发和调度所有 DDL (数据库模式定义语言) 操作, 通常决策集群需要采用奇数台, 实际部署的时候应大于等于 3 组并跨机房部署。
- **赤兔自动化运维平台 (CHITU)**: 基于 DCDB 定制开发的一套综合的业务运营和管理平台, 同时也是真正融合了数据库管理特点, 将网络管理、系统管理、监控服务有机整合在一起。
- **接入网关集群 (OLTP-Proxy)**: 在网络层连接管理 SQL 解析、分配路由。(请注意, OLTP-Proxy 并非腾讯云网关 TGW 集群)。
 1. OLTP-Proxy 通常与 MySQL 混合部署, 也可以部署在不同物理设备中;
 2. 从配置集群 (TzooKeeper) 拉取数据库节点 (分片) 状态, 提供分片路由, 实现透明读写;
 3. 记录并监控 SQL 执行信息, 分析 SQL 执行效率, 记录并监控用户接入信息, 进行安全性鉴权, 阻断风险操作;
 4. OLTP-Proxy 通常可以直接访问, 但仍然建议前端部署需部署可提供负载均衡能力网关,

并由网关对用户提供唯一虚拟 IP 服务。

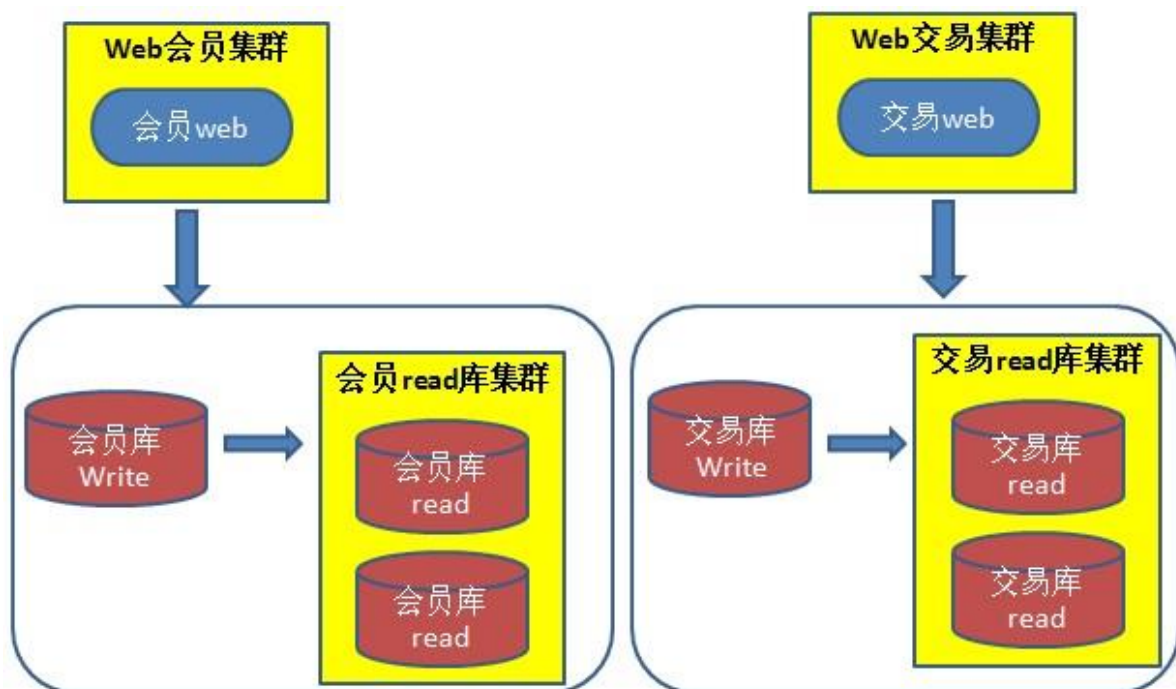
这种集群架构极大简化了各个节点之间的通信机制，也简化了对于硬件的需求，这就意味着即使是简单的 x86 服务器，也可以搭建出类似于小型机、共享存储等一样稳定可靠的数据库。

4.2 分库与分表

在高性能并发互联网架构中，性能瓶颈往往出现在数据库服务器，特别是当业务(用户)达到百万级用户规模以后，通过在数据层进行合理的数据切分（shard），可以有效解决数据库性能、可伸缩等问题。数据库切分同样是从两个维度考虑：垂直切分(按功能切分)和水平切分。

4.2.1 分库

垂直切分(通常也叫做“分库”)也就是按功能切分数据库，这种切分方法跟业务紧密相关，实施思路也比较直接，比如“京东 JD”等电商平台，一个原有一个数据库实例，按功能切分为会员数据库、商品数据库、交易数据库、物流数据库等多个数据库实例，共同承担业务压力。

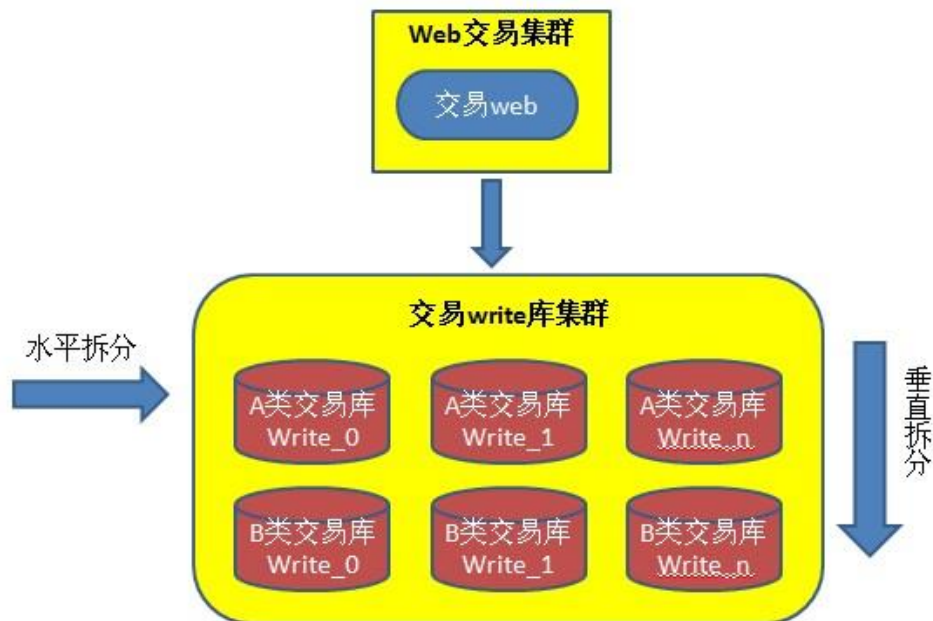


4.2.2 分表

有时候，垂直拆分并不能彻底解决压力问题，因为单台数据库服务器的负载和容量也是有限

的，随着业务发展势必也会成为瓶颈，解决这些问题的常见方案就是水平切分了。

水平切分（又叫做“分表”、横向拆分等）是按照某种规则，将一个表的数据分散到多个物理独立的数据库服务器中，这些“独立”的数据库“分片”；多个分片组成一个逻辑完整的数据库实例。一般来说，分表的前提是分库。



水平拆分的方案，实际上是分布式架构最直接体现，他与 RAC 等方案的最大不同点是，每个计算节点都参与计算和数据存储，每个计算节点都仅存储一部分数据。因此，分布式数据库从架构上来讲，性能是可以线性增长的。

4.2.3 DCDB 分表原理简介

关系型数据库是一个二维模型，数据的切分通常就需要找到数据库表中某一字段作为拆分键 (shardkey) 以确定拆分维度，再通过定义规则来实现数据库的拆分，几种常见的分表规则如下：

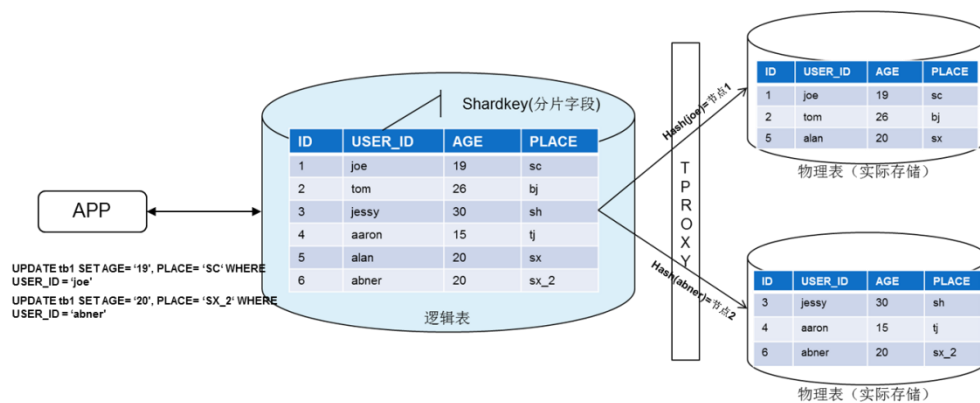
- 基于日期顺序(Time)，如取 shardkey 为时间字段，按年拆分，2015 年一个分片，2016 年一个分片。
- 基于某字段划分范围 (Range)，如取 shardkey 为用户 ID，0~1000 一个分片，

1001~2000 一个分片。

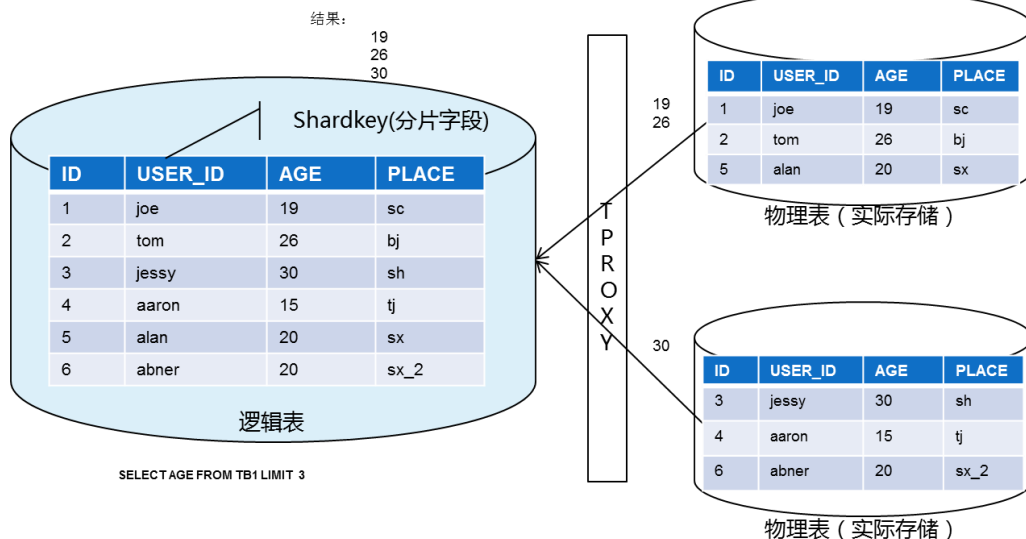
- 基于某字段求模 (HASH) , 如取 shardkey 为订单 ID , 将求模 (HASH) 后按照 Range 方式分散到不同分片中。

无论是 Time、Range 都有个主要缺点就是可能导致严重数据倾斜, 即多个分片之间负载和数据容量严重不均衡。例如, 在大部分数据库系统中, 数据有明显的冷热特征——显然当前的订单被访问的概率比半年前的订单要高的多——而单纯采用 Time 分表或 range 分表, 就意味大部分热数据将容易被路由在少数几个分片中, 而存储冷数据的设备性能却被浪费掉了。

因此, DCDB 默认采用某个字段求模 (HASH) 的方案进行分表。因为 HASH 算法能够基本保证数据相对均匀的分散在不同的物理设备中。具体过程即记录(SQL)请求时被发起时, DCDB 会理解 SQL 语句的含义, 然后按照拆分键的值和执行策略将 SQL 路由到对应分表进行执行, 如下图所示, 先通过 hash 算法计算, 再路由到各个节点上。



而如果一个查询 SQL 语句的数据涉及到多个分表, 此时 SQL 会被路由到多个分表执行, DCDB 会将各个分表返回的数据按照原始 SQL 语义进行合并, 并将最终结果返回给用户。



读取数据时 (如果有明确 shardkey 值):

- 业务发送 select 请求中含有 shardkey 时, 网关通过对 shardkey 进行 hash
- 不同的 hash 值范围对应不同的分表
- 数据根据分表算法, 将数据从对应的分表中取出

读取数据时 (如果没有明确 shardkey 值):

- 业务发送 select 请求没有 shardkey 时, 将请求发往所有分表
- 各个分表查询自身内容, 发回 Proxy;
- Proxy 根据 SQL 规则, 对数据进行聚合, 再答复给网关

从上述原理来看, 查询 SQL 中含有 shardkey 值比不含 shardkey 值效率将会更高。

4.2.4 选择拆分键 (shardkey)

拆分键是在水平拆分过程中用于生成拆分规则的数据表字段。DCDB 建议拆分键要尽可能找到数据表中的数据在业务逻辑上的主体, 并确定大部分 (或核心的) 数据库操作都是围绕这个主体的数据进行, 然后可使用该主体对应的字段作为拆分键, 进行分表 (该分表方案叫做 groupshard), 如下图:



基于 Groupshard 的分表方案，确保可以确保某些复杂的业务逻辑运算，可以聚合到一个物理分片内。例如，某电商平台订单表和用户表都是基于用户维度（UserID）拆分，平台就可以很容易的通过联合查询（不会存在跨节点 join，或分布式事务）快速计算某个用户近期产生了多少订单。

下面的一些典型应用场景都有明确的业务逻辑主体，可用于拆分键：

- 面向用户的互联网应用，都是围绕用户维度来做各种操作，那么业务逻辑主体就是用户，可使用用户对应的字段作为拆分键；
- 电商应用或 O2O 应用，都是围绕卖家/买家维度来进行各种操作，那么业务逻辑主体就是卖家/买家，可使用卖家/买家对应的字段作为拆分键；但请注意，某些情况下几个超大卖家占到绝大多数交易额，这种情况会导致某几个分片的负载和压力明显高于其他分片，我们会在后面章节予以说明。
- 游戏类的应用，是围绕玩家维度来做各种操作，那么业务逻辑主体就是玩家，可使用玩家对应的字段作为拆分键；
- 物联网方面的应用，则是基于物联信息进行操作，那么业务逻辑主体就是传感器/SIM 卡，可使用传感器、独立设备、SIM 卡的 IMEI 作为对应的字段作为拆分键；
- 税务/工商类的应用，主要是基于纳税人/法人的信息来开展前台业务，那么业务逻辑主体就是纳税人/法人，可使用纳税人/法人对应的字段作为拆分键。

以此类推，其它类型的应用场景，大多也能找到合适的业务逻辑主体作为拆分键的选择。

4.2.5 同表不同拆分建的场景

在上面选择 shardkey 的例子都是基于一个维度查询，但如果是电商订单表，肯定即需要从用户维度查询，又需要从商户维度查询，那么如果这个系统会有多个主要的查询维度时，就会存在采用同表不同拆分建的场景；解决这类查询有两种方案

- 采用全表扫描，虽然这种场景会带来较大的读负载，但由于存在读写分离策略，此种方案也是可行的。
- 即使用异构索引表，其本质就是利用数据库多源同步工具，将原表内的每一次更新，都同步到数据库中的另一张结构相同，但拆分建不同的新的表中。这时，你只需要简单映射一下，异构索引表的作用就基本等同于传统数据库中的索引概念。但此处由于经过了多源同步工具的，即新的表数据更新可能会有一定延迟。

4.2.6 拆分键选择的限制

普通的分表创建时必须在最后面指定 shardkey 的值，该值为表中的一个字段名字，会用于后续 sql 的路由选择：

```
mysql> create table test1 ( a int, b int, c char(20),primary key (a,b),unique key u_1(a,c) )
shardkey=a;
```

由于在 DCDB 下，shardkey 对应后端数据库的分区字段，因此必须是主键以及所有唯一索引的一部分，否则没法创建表，如下错误语句：`mysql> create table test1 ( a int, b int, c char(20),primary key (a,b),unique key u_1(a,c),unique key u_2(b,c) ) shardkey=a ;`此时有一个唯一索引 u_2 不包含 shardkey，没法创建表，会报如下错误：``ERROR 1105 (HY000): A UNIQUE INDEX must include all columns in the table's partitioning function``。因为主键索

引或者 unique key 索引意味着需要全局唯一，而要实现全局唯一索引则必须包含 shardkey 字段。

除了上面的限制外，shardkey 字段还有如下要求：

- a) shardkey 字段的类型必须是 int, bigint, smallint/char/varchar。
- b) shardkey 字段的值不应该有中文，网关不会转换字符集，所以不同字符集可能会路由到不同的分区。
- c) 不要 update shardkey 字段的值。
- d) shardkey=a 放在 sql 的最后面。
- e) 访问数据尽量都能带上 shardkey 字段，这个不是强制要求，但是不带 shardkey 的 sql 会路由到所有节点，消耗较多资源。

如果是广播表，此时该表在所有 set 中都是全量数据，这个主要方便于跨 set 的 join 操作，同时通过分布式事务保证修改操作的原子性，使得所有 set 的数据是完全一致的：

```
mysql> create table global_table ( a int, b int key) shardkey=noshardkey_allset;
```

如果是单表，语法和 mysql 完全一样，此时该表的数据全量存在第一个 set 中，所有该类型的表都放在第一个 set 中：

```
mysql> create table noshard_table ( a int, b int key);
```



注意：更多详情详见腾讯云官网 DCDB 开发指南。

4.3 分布式数据库技术特征

4.3.1 自动水平拆分

只需在建表的时候设定 shardkey，即支持对数据库中大表自动水平拆分（分表），系统将基于 Hash 方案自动将写入数据均匀的分布到不同物理分片中，查询也自动聚合返回；分表而对业务系统透明，业务实际所见为一张逻辑完整的表，而无需感知后端的物理架构。

4.3.2 高度兼容 MySQL 语法

DCDB 兼容绝大多数常用的 MySQL 语法，包括 MySQL 的语言结构、字符集和时区、数据类型、常用函数、预处理协议、排序、联合（join）、存储过程、索引、分区、事务、预处理协议、控制指令、等常用的 DDL、DML、DCL 语句语法。

4.3.3 补齐分布式架构的不足

因分布式架构的特殊性，在数据库高级功能与性能之间不能两全；为此 DCDB 提供三种建表方案，提供（分布式）事务特性，提供全局唯一数字序列，支持 JSON 等能力，有效的弥补了分布式架构的不足，为开发者提供更加灵活的开发方案。

4.3.4 不停机弹性扩展

目前单一片最大可支持 6TB 存储 如果性能或容量不足以支撑业务发展时，在控制台点击，即可自动升级完成。升级过程中，您无需关心分布式系统内的数据迁移，均衡和路由切换。升级完成后访问 IP 不变，仅在自动切换时存在秒级闪断，您仅需确保有重连机制即可。

4.3.5 基于数据强一致的高可用

DCDB 默认采用主从架构，可确保 99.95%以上可用性；系统支持强同步复制已提供数据强一致，业务系统写入数据后，只有当从机同步后才给予应用事务应答，确保主从数据完全一致，不会因故障导致数据丢失、错乱，目前强同步复制性能已基本等于异步复制。

4.3.6 超高性能

DCDB 深度定制开发 MySQL 内核,性能远超基于开源 MySQL ;支持三种方案的读写分离,有效提供读扩展的同时提供开发灵活性;对线程池调度算法进行了优化,在重负载时表现更佳;并配置 PCI-E SSD 的硬盘,提供高于 SATA 三倍以上 IO 配置,帮助您更轻松满足业务性能需求。

4.3.7 易于使用的托管部署

只需在腾讯云 DCDB 管理控制台中单击几下,即可在几分钟内启动并连接到一个可以立即投入生产的 DCDB 数据库。控制台提供常见的数据库运维操作,为您精细管理数据库提供便利;提供常见的系统监控数据和性能分析数据,祝您迅速识别运行异常的数据库。

4.3.8 OLAP 扩展

DCDB 是一个 OLTP 数据库,但为解决用户复杂 OLAP 需求,DCDB 支持向 ADB 扩展,通过自研 OLAP-Proxy 和 spark 能力,无需增加额外存储,即可为用户提供一站式数据分析能力;(邀请测试中)

5 产品优势

5.1 高可用与强同步

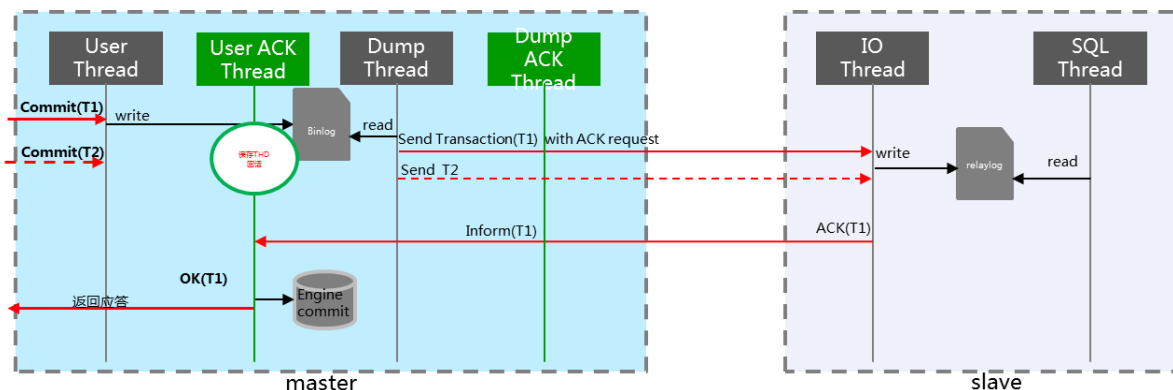
在生产系统中,通常都需要用高可用方案来保证系统不间断运行;数据库作为系统数据存储和服务的核心能力,其可用要求高于计算服务资源。由于数据库中记录了数据,想要在三台数据库中切换,数据必须是同步的,所以数据同步技术是数据库高可用方案的基础。

腾讯自主研发了基于 MySQL 协议的异步多线程强同步复制方案 (Multi-thread Asynchronous Replication MAR), 相比于 Oracle 的 NDB 引擎, Percona XtraDB Cluster 和 MariaDB Galera Cluster, 其性能、效率和适用性更据优势。简单来说, MAR 强同步方案强同

步技术具有以下特点：

- 一致性的同步复制，保证节点间数据强一致性；
- 对业务层面完全透明，业务层面无需做读写分离或同步强化工作；
- 将串行同步线程异步化，引入线程池能力，大幅度提高性能
- 支持自动成员控制，故障节点自动从集群中移除；
- 支持自动节点加入，无需人工干预；
- 每个节点都包含完整的数据副本，可以随时切换；
- 无需共享存储设备，成本更低

腾讯强同步技术，只有当备机数据同步后，才由主机向应用返回事务应答，示意图如下：



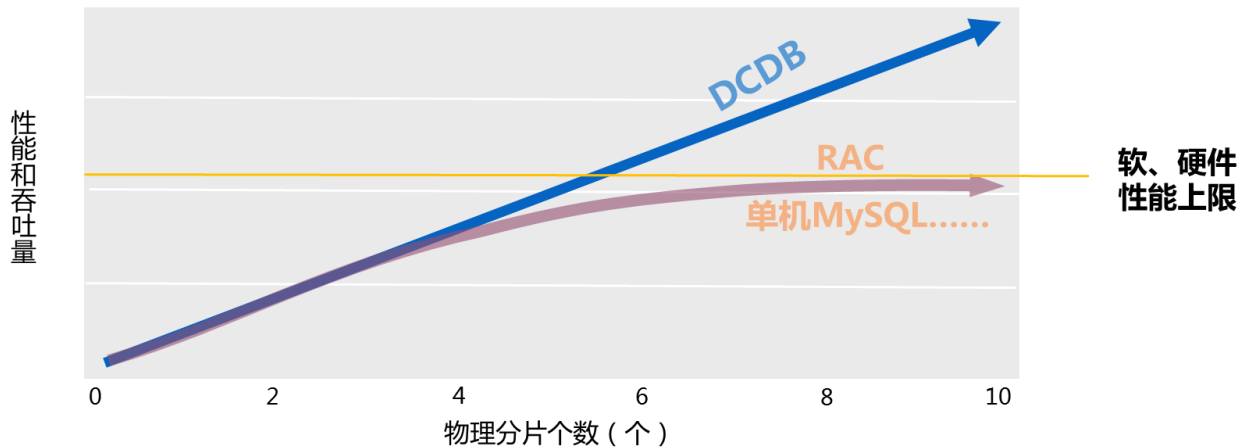
腾讯强同步从性能上优于其他主流同步方案，通过对比在跨可用区(IDC 机房，延迟约 10~20ms)同样的测试方案下，我们发现其 MAR 技术性能优于 MySQL 5.6 半同步约 5 倍，优于 MariaDB Galera Cluster 性能 1.5 倍（此处测试使用 sysbench 标准用例测试）。

同步方案（跨 IDC 测试）	最大 QPS（100 并发水平）	平均耗时（ms）
DCDB MAR 强同步	486004	26
MySQL 5.7 半同步	386513	32

MySQL 5.6 半同步	10720	42
DCDB 异步同步	486004	13
MySQL 5.7 异步同步	418186	12

5.2 性能与容量线性增长

腾讯分布式数据库 DCDB 是天然的 MPP (Massively Parallel Processing, 大规模并行处理系统)架构, 采用分布式架构是随着分片 (SET) 的增加, 每个分片各自承担一部分分布式任务, 意味着并发性能、处理能力、存储容量将线性增长。

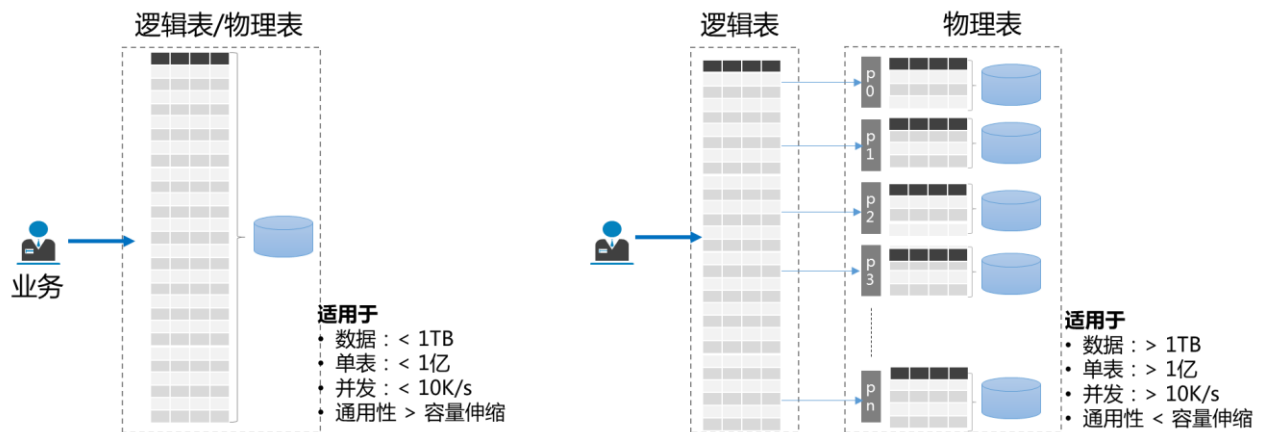


并且 DCDB 默认采用线程池, 且对调度算法进行了优化, 改进当系统内核处于重负载时, 查询和更新请求在线程组之间分布不均衡等极端情况下性能, 并且能够更好地利用计算资源, 减少无谓的线程切换, 减少请求在队列中的等待时间, 及时处理请求。类似的内核优化还有很多, 通过 sysbench 的压力测试, DCDB 单个分片纯写入操作能超过 12 万+TPS, 纯查询操作能超过 48 万 QPS, 是 MySQL5.6 性能的 4 倍, MySQL5.7 的 2 倍以上。

5.3 逻辑表

腾讯分布式数据库 DCDB 对应用来说, 读写数据完全透明, 对业务呈现的表实际上是逻辑表。逻辑表屏蔽了物理层实际存储规则, 业务无需关心数据层如何存储, 只需要基于业务表应该如何

设计。



DCDB 为用户提供了三种类似的表：分表，小表以及单表：

- **分表**：即水平拆分的表，通常适用于表规模大于一亿行、大于 100GB，且快速增长的表；



说明：需要拆分的标准并无一个准确值，通常取决于开发者对业务的判断和对性能的预期。

- **广播表**：即所有操作都将广播到所有逻辑分片（Shard）中，这意味着每个分片都有该表的全量数据，通常适用于配置表，或需要频繁 JOIN 的表，该类表数据更新相对较少，广播表可以两个表的联合查询（JOIN）事务收敛到单节点中，以提高性能；
- **单表**：一些无需分片的表，通常适用于数据量较小的库表；由于单表并未拆分，这意味着单表使用完全兼容 MySQL。

5.4 实时监控告警

通过 DCDB 的监控模块，DCDB 所有的内部运营状态或数据都能实时采集并分析，数据一份上报到调度集群，一份上报到 DCDB 监控系统，如下图：



基于调度集群的自动分析，可以随时自动处理集群任意节点的故障，实现智能调度和自动切换。而通过监控系统和告警系统，数据库管理员（DBA）基本上可以随时获悉集群状态，并通过管理平台进行所有常规的运营操作，而不需要登录设备实际检查。

5.5 高性能分布式事务

腾讯分布式数据库 DCDB 默认支持读写分离能力，架构中的每个从机都能支持只读能力，如果配置有多个从机，将由网关集群（OLTP-Proxy）自动分配到低负载从机上，以支撑大型应用程序的读取流量；我们提供多种读写分离方案供您选择，且您无需关注若干从机是否完全存活，因为系统将根据策略自动调度

- **只读帐号**：您仅需要在创建帐号时，标记为只读帐号，系统将根据策略向将读请求发往从机；
- **/*slave*/注释**：您可以在编程过程中，通过注释/*slave*/，系统将把该条语句发往从机，常用于编程阶段将特殊的读逻辑嵌入代码。
- **只读实例**：如果从机仍无法满足您的读需求，您可以添加只读实例。值得注意的是，只读实例并不参与高可用切换。

由此可为应用提高总的读取吞吐量；通过多种只读方案的组合，您可以配置出复杂的只读方案，以满足各种业务需求和开发的灵活性。

5.6 全局唯一数字序列

数据切分后，原有的关系数据库中的主键约束在分布式条件下将无法使用，因此需要引入外

部机制保证数据唯一性标识，这种保证全局性的数据唯一标识的机制就是全局唯一数字序列（sequence）。

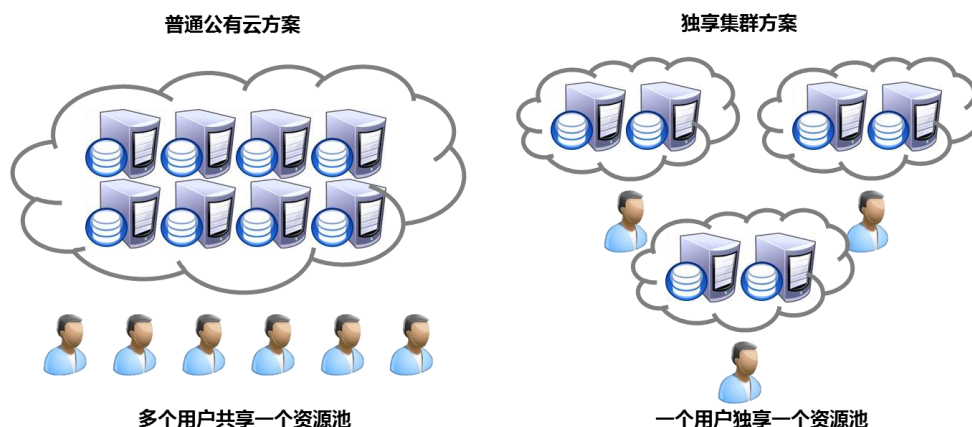
DCDB 全局唯一数字序列（以下简称 sequence，使用的是 unsigned long 类型，8 个字节长），使用方法与 MySQL 的 AUTO_INCREMENT 类似。目前 DCDB 可以保证该字段全局唯一和有序递增，但不保证连续性。

5.7 多租户与独享集群

DCDB 默认采用多租户方式为您提供服务，多租户意味着您的实例并非独占一整套集群，而是与其他租户共用。腾讯云多租户方案保障客户的数据安全隔离，当实例被销毁时，数据也将被完全销毁。

DCDB 也支持独享集群数据库（Database Dedicated Cluster）方案。独享集群，可以让您以独享物理集群（完整设备）资源方式购买、创建数据库，以满足您对资源独享、物理安全、行业监管等需求；购买独享集群后，您可以在其上灵活创建多种自定义规格的云数据库。

独享集群与腾讯云多租户的方案可以用下图表示：



独享集群方案可以享有 DCDB 的全部能力，但在部署架构上有如下区别：

- 数据库服务器：承载数据库的物理主机，是使用数据库的主要部分，您将完全独占。
- 腾讯云综合运营管理系统：部署在腾讯云更高等级的管理网络中，是您提供灵活的运维能力

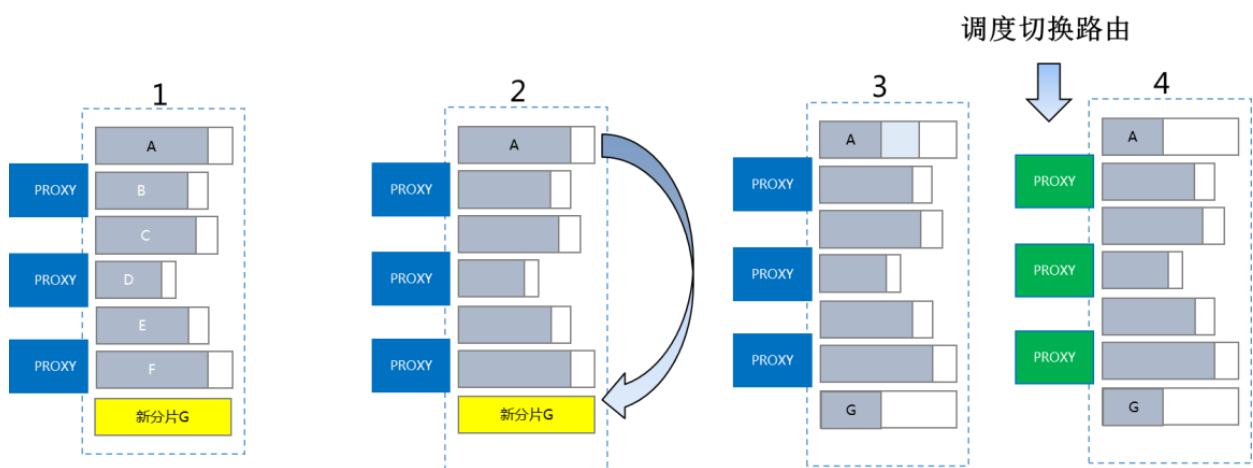
的基础，您将与其他用户共享。

- 腾讯云网关/安全网关：部署在数据库前端的提供智能负载均衡和安全管理虚拟网络设备，为您的数据库提供 VIP（唯一虚拟 IP）、VPC、主备切换、安全防护等能力，您将与其他用户共享。
- 您可能得自留资源冗余，以保障集群的高可用。

5.8 弹性扩展

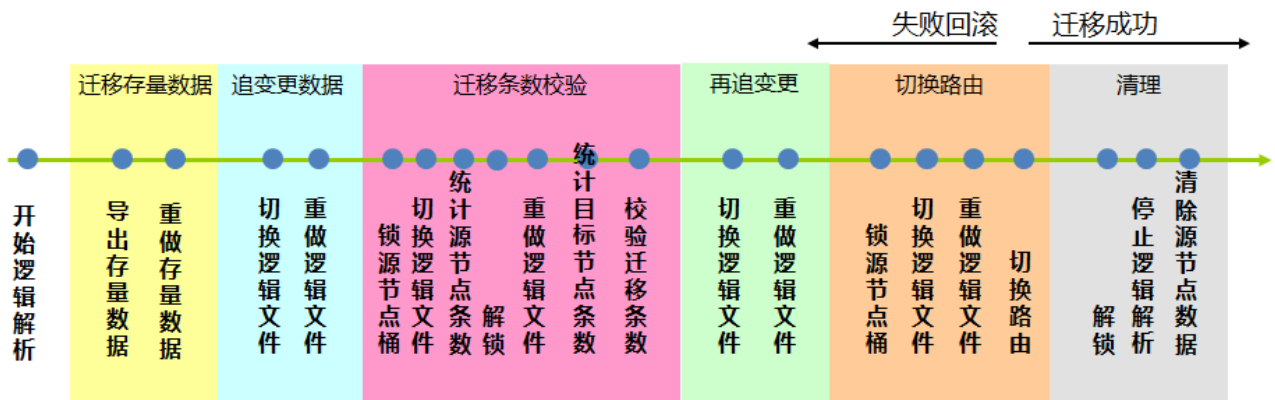
DCDB 支持在线实时扩容分为新增分片和对现有分片扩容两种方式。DCDB 在线扩容过程对业务完全透明，无需业务停机。扩容时仅部分分片存在秒级的只读（或中断），整个集群不会受影响。

DCDB 主要是采用腾讯自研的自动再均衡技术（rebalance）保证自动化的扩容和稳定，以新增分片为例，扩容过程如下下图：



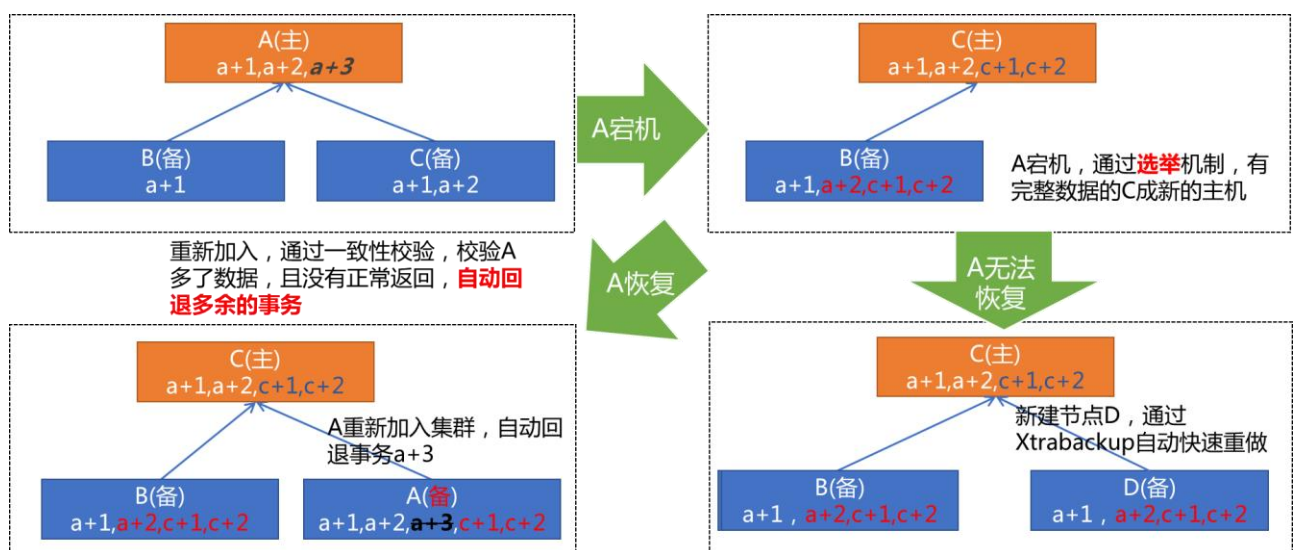
- 1) 控制台点击扩容后，系统根据负载和容量计算出 A 节点（可也手工指定节点）存在瓶颈。
- 2) 根据新加 G 节点配置，将 A 节点部分数据搬迁（备机）到 G 节点。
- 3) 数据完全同步后，AG 校验数据库，（存在 1~几十秒的只读），但整个服务不会停止。
- 4) 调度通知 proxy 切换路由。

为确保业务不停以及数据一致性，DCDB 的整个迁移过程采用移存量数据、迁移增量数据、数据检验、再追增量、切换路由、清理六个步骤循环迭代进行。该能力经过腾讯内外海量业务迁移，至今未发生过一次数据异常错误或全集群停机。



5.9 容灾与恢复

DCDB 的每一个分片都支持基于强同步的一主多从架构（默认为一主一从，异步同步），主数据库故障时备机立即顶替工作，切换过程对用户透明，且不改变访问 IP。并且对数据库和底层物理设备提供 7X24 小时持续监控。发生故障时，DCDB 将自动重启数据库及相关进程；如果节点崩溃无法恢复，将通过备份文件自动重建节点（如下图）。该功能持续保障数据库可用性在 99.99% 以上，以帮助业务持续稳定的运行。



而 DCDB 也支持定时备份您的数据库实例，自动备份保留期可配置，自动备份存储在统一的

安全备份存储内，提供 99.99999% 的数据持久性；在业务出现异常时，自动备份和恢复能力可使您的业务损失减少到最小。

6 应用场景

6.1 电子商务类应用

几乎所有大型电子商务平台基于分布式数据库，其中性能是最重要的考虑；当大流量推广的时候，只有分布式架构的数据库可免受物理硬件性能限制，性能线性扩展。2017 年，鹅漫 U 品上线不足一年，每个月都保持着高速增长，到中期的日均单超过 1 万单时，普通数据库性能和扩展问题就显现出来了，经过 1 个月不到的改造，成功迁移到 DCDB，最后顺利通过双 11，双 12 等各类大型推广活动。

6.2 金融类应用

随着手机银行、网上理财、数字货币等等具有互联网特色的金融业务兴起，单笔交易变小，交易次数变多等情况；传统架构逐渐不足以支撑业务发展，网络安全风险逐渐增加；DCDB 不仅在性能容易扩展，强同步能力也确保数据不错不丢，是国内第一个将分布式事务应用于金融系统的产品，同时支持部署在腾讯金融云，支持物理独享，加密，审计等系列安全方案。

6.3 IoT 类应用

在工业监控和远程控制、智慧城市的延展、智能家居、车联网等物联网场景下，传感监控设备多，采样率高，数据存储要求高，数据规模存储规模问题凸显；DCDB 容量的线性扩展不仅可有效解决容量问题；其支持 MySQL 协议和 JSON 也能让开发者用自己熟悉的协议开发系统；同时可扩展 RocksDB，让数据压缩率低至 20% 以下，二级分区让冷热数据有效快速的分离处理，极大的降低了数据存储成本。

6.4 游戏应用

全区全服的 SNS 游戏,新进 IO 等游戏品类,PCU 很容易到达 10w 甚至 1000W 都是有可能的,所以在设计之初就要考虑所有的功能模块都要具有可平滑扩展的能力。GameSvr、MatchSvr 等无状态服务器扩展相对容易,但记录游戏中全部好友关系链,全部历史战绩,游戏交易记录的 DBSvr 扩展就成为难题。而 DCDB 能比较友好的解决类似全区全服的扩展问题。

6.5 成为去 O 的中坚力量

企业的核心业务系统一般都是 OLTP 为主的应用场景,在这个领域,Oracle 一直是市场的领导者,而开源数据库 MySQL、MariaDB、PostgreSQL 等似乎仅能提供给中小企业或个人站长使用。在互联网领域,以 DCDB 为代表的分布式数据库应用非常广泛,用普通 x86 服务器,轻松支撑起上亿的用户访问,经过验证的好的分布式数据库在性能和稳定性上甚至高于用高端设备搭建的 Oracle RAC。

当然,对于企业而言,由于 Oracle 数据库和上层应用绑定比较紧密,通常会使用到 Oracle 的存储过程、自定义函数、触发器,这就需要涉及到应用迁移,这个工作的工作量和时间周期通常较大,但综合计算下来,即使加上软件改造成本,采用 DCDB 的 TCO 仍然低于使用商业数据库,当前,不管是互联网和传统行业,去 O 的成功案例比比皆是。

6.6 分支业务聚合到总部

由于政务、银行、大型国企的组织架构通常采用总部-分部-分支的架构;因为各种原因,其某些核心 IT 系统建设也采用总部-分部-分支模式。随着业务互通,人员互通,信息互通等需求越来越强烈,业务逐渐向总部聚合。而业务聚合一个重要问题是数据库性能和容量无法承载。以某部委为例,其省级业务系统数据规模和性能已经在用最高端的商业数据库硬件承载。如果聚合到总部,一是设备性能扩无可扩,二是软件费用和硬件成本将会是天价。因此,到现在为止,不少业务也仅能做到数据汇总,而非业务聚合。

DCDB 此类分布式数据库在微信支付、京东等超大规模业务的应用证明了，一个系统承载全国业务的可能性。在 2014 年马年春节，总共 800 万微信用户参与争抢 4000 万个红包，最高峰期间的 1 分钟有 2.5 万个红包被领取。2016 年大年初一每分钟有 55 万个红包被发出、165 万个红包被拆开，增长近百倍。