# OceanBase分布式应用

## **分布式存储**

OceanBase是分布式数据库，支持表的横向分区存储；由于分区存储数据库事务管理复杂、表关联查询容易出现性能问题，因此建议尽量不进行分区存储。

但是，对于数量超大的系统，采用合理的分区存储可以提高数据库性能，突破单库或单表的性能瓶颈，解决数据库的单表瓶颈、读写并发量高等问题；另外，OceanBase不支持sharding-jdbc等分表组件访问；因此迁移到OceanBase后需要采用分区存储方式。同时OCeanBase采用分区存储后，可以实现OB集群中的每个Zone都提供数据读写服务，提升数据库性能，提高资源利用率。

## **分区介绍**

分区是将一个表或索引物理地分解为多个更小、更可管理的部分，实现单个操作在单个分区或多个分区并行执行，实现将单个操作任务分解多个小次操作并行执行效果，已提高效率。通常用来在业务明确对数据进行某个维度的管理，或者并发对多分区进行分析查询的时候采用。

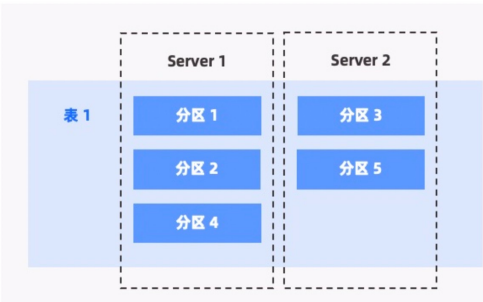
分区缺点:

1. 跨分区事务管理复杂，跨分区事务需要协调多个分区进行操作，事务一致性、原子性、隔离性和持久性需要同步不同分区，管理复杂。
2. 表之间的关联查询风险，采用分区表的数据打散在不同的分区，在进行表关联时，会存在对全分区表进行扫描、汇聚计算的情况，导致分区之间进行大量数据交互、汇聚后占用大量空间，失去分区的意义，引起数据库整体性能下降。

## **OceanBase分区介绍**

#### （1）基本原理

OceanBase采用分区表方式，实现对分区存储；如下图所示，一张表被划分成了 5个分区， 分布在2台机器上；每个分区只能存在于一个节点内部，分区表的不同分区可以分散在不同节点上。



同时，支持二级分区表，即：分区表的每个分区还能按照一定的规则再折分成多个分区。

OceanBase通过分区键，即：数据表中每一行中用于计算这一行属于哪一个分区的列的集合。分区键必须是主键或唯一键的子集；由分区键构成的用于计算这一行属于哪一个分区的表达式叫做分区表达式，目前分区表达式支持，范围(RANGE )、列表(LIST )、哈希(HASH )、组合四种形式。

对于同一个租户下的非分区表作为一个独立的分区表对待。

#### **（2）分区管理**

为避免出现分区带来的缺点，OceanBase采取如下方案处理。

* 分区事务管理

1. 分布式事务跨机执行时，OceanBase通过多种机制保证ACID
2. 原子性：依赖两阶段提交协议保证分布式事务的原子性。
3. 一致性：保证主键唯一等一致性约束，全局快照 - 单租户GTS服务，1秒钟内能够响应获取全局时间戳的调用次数超过200万次。
4. 隔离性：采用MVCC进行并发控制，实现read-committed的隔离级别；所有修改的行加互斥锁，实现写-写互斥；读操作读取特定快照版本的数据，读写互不阻塞；
5. 持久性：Redo-Log使用Paxos协议做多副本同步。

2、OceanBase全局一致性快照技术

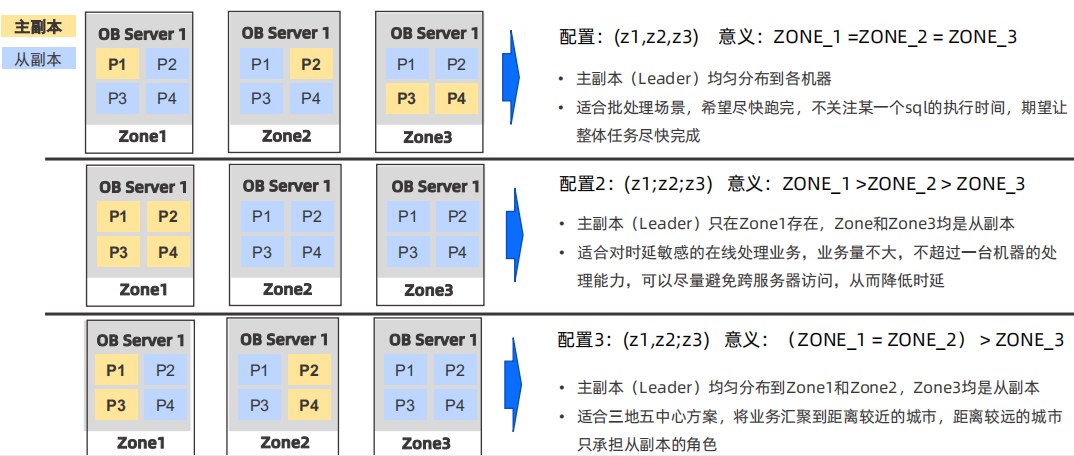
OceanBase数据库是利用一个集中式服务来提供全局一致的版本号。事务在修改数据或者查询数据的时候，无论请求源自哪台物理机器，都会从这个集中式的服务处获取版本号，OceanBase则保证所有的版本号单调向前并且和真实世界的时间顺序保持一致。

1. OceanBase两阶段提交协议
2. 协调者不写日志，变成了一个无持久化状态的状态机
3. 事务的状态由参与者的持久化状态决定
4. 所有参与者都prepare成功即认为事务进入提交状态，立即返回客户端commit
5. 每个参与者都需要持久化参与者列表，方便异常恢复时构建协调者状态机，推进事务状态
6. 参与者增加clear阶段，标记事务状态机是否终止
7. 分布式事务底层优化
8. 单分区事务：不走2PC ，直接写一条日志即可完成事务提交
9. 单机多分区事务→ 优化的两阶段提交
10. 多机多分区事务→完整的两阶段提交 →prepare, commit/abort
11. 分布式事务调优方法
12. 业务数据模型设计原则：尽量避免跨机分布式事务
13. 单sql语句不建议跨机器通过table group、primary\_zone把相关的表的leader放在同一个机器上
14. 慎重选择事务中的第一条语句，因为Obproxy的路由规则

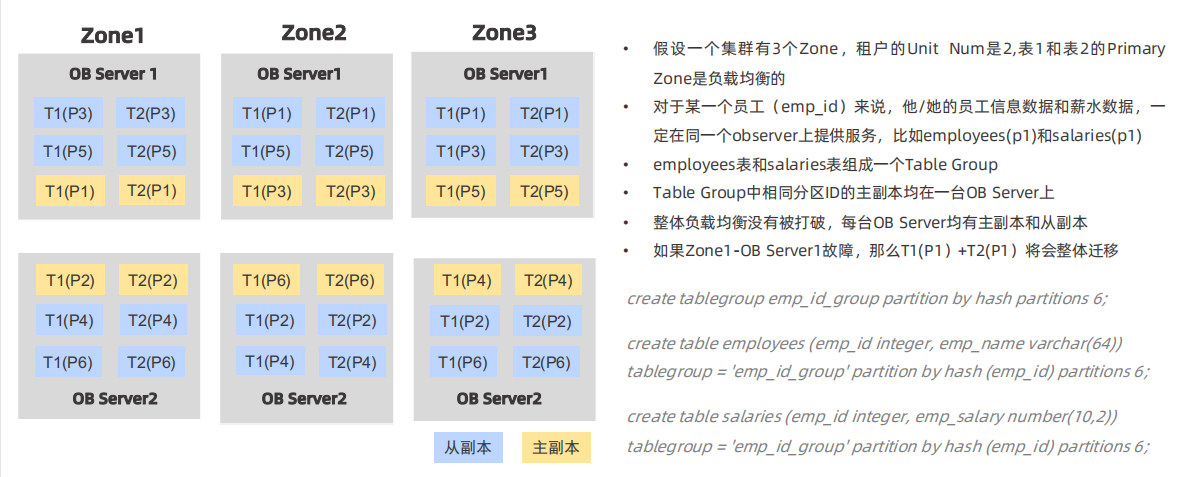
* 分区表关联查询风险

1. 设置Primary Zone，降低跨zone分区关联查询风险

通过设置Primary Zone，将业务汇聚到特定Zone，如下图：



1. Table Group，将多个表的相同分区ID的主副本聚集在一个OBServer中，减少分布式事务、关联查询引入的开销，如下图;



### 3）分区适配步骤

#### （1）分为表确认

对业务场景、表关系和系统数据库操作进行全面梳理，重点明确以下信息：

* 按照表功能将表分类为：业务相关表、配置相关表、日志表等方面；
* 明确各个表的数据量和数据增长速度；
* 明确配置表与业务表的关系，是否存在关联查询场景；
* 根据业务领域，将业务表再进行分类，分成独立的模块
* 对系统数据库操作分析，是否存在跨业务表模块操作、业务表与配置表、日志表与配置表及业务表的关联操作，若存在是否可以进行改造。

根据以上信息，确认迁移到OceanBase后，分区表、非分区表（单独使用一个分区）、复制表的分类。

#### （2）分区键确认及系统改造

对分区表的分区键进行确认，主要明确分区键、分区表达式，确认系统改造量，主要包括一下内容：

* 根据表之间的关联关系、常用查询条件、数据分布等方面综合考虑确认分区键（分区键可以是多个列），不同业务模块的分区键、分区表达式可以不同；
* 分区表达式类型：

1. 范围(RANGE )分区：分区表达式的值位于一个给定的连续区间内。一般使用这种分区方式大都是对连续的值进行分区，如：按年份，日期进行分区。
2. 列表(LIST )分区：列值在一个离散值集合中的某个值来进行选择，如：机构编码、渠道编码等。
3. 哈希(HASH )：对分区键应用一个散列函数，得出数据应该放在n个分区中的哪一个分区。如：ID、保单号等。
4. 组合分区，采用一、二级分区方式。

特别说明：HASH分区不支持动态修改（需要重新建表调整分区），RANGE、LIST分区支持动态修改。

分区键确认后，需要对系统进行改造，主要改造涉及以下方面：

1. 禁止非分区表与分区表之间的关联操作，可拆分两个查询依次执行方式；
2. 分区键类型不同的表，禁止关联查询操作；
3. 分区表的关联操作必须使用含有分区键条件；
4. 尽量进行单表操作，避免表关联操作。

#### （3）分区优化设置

由于数据分散在不同分区，在分区汇聚操作、非分区表关联操作时，会出现分区数据搬移情况，当数据量较大时，会大量消耗数据库内存及网络带宽，降低数据库整体性能。因此可以利用OceanBase数据库的Primary Zone和table group性质降低分区数据搬移。具体使用场景如下：

* Primary Zone可以将所有的主副本放在一个OB server中，避免OceanBase数据库的跨OB server的数据搬移。适用于系统表关系复杂，无法实现分区表之间关联操作均使用分区键，存在较多的数据统计汇总计算的场景。
* Table group可以实现将相同分区键值的不同表，分派到同一个分区；避免了关联表之间的跨分区查询。适用于系统表关系不太复杂，可以实现分区表之间关联操作均使用分区键，分区键在多个表中，且关联操作较多的情况。

#### （4）完善数据库脚本

采用分区存储表、复制表、Primary Zone、table group等创建的语法在oracle数据库中不存在，另外表设置与系统关系紧密，因此需要系统负责人完善数据库脚本，并供DBA执行；执行完毕后在进行数据导入。

#### （5）功能测试及性能测试

除正常功能测试外，更要关注数据分区存储的分布是否与分区键的规则相同，日常功能测试时系统反映是否正常，事务处理是否正常，超长事务是否正常，还需关注是否存在未发现的跨分区操作；另外，对系统重要功能、存在的已知的跨分区操作，要进行重点性能测试，以保证分区存储后，数据库性能满足系统要求。

#### （6）生产数据库建立

在确定数据库脚本后，就可以申请生产数据库，并将当时生产库的全量数据导入的新的OceanBase数据库，验证分区存储是否满足数据迁移要求。同时，在原有数据库和新库直接部署OMA，进行原有数据库对象及sql的兼容性进行评估，并记录生产数据库执行的sql语句，未后续的sql功能测试和性能测试做准备。

#### （7）生产性能压测

利用OMA捕获的原有数据库一段时间内（一周、两周或一个月等）的sql语句，在新的OceanBase数据库上进行回放，回放时可以将执行时间进行压缩，如：原有一天的sql语句，在OceanBase上1小时内执行完毕等；进行性能测试，对评估回放结果，主要是执行较多的SQL、耗时较长sql进行优化（代码优化和数据存储优化）；通过多次回访，使数据库性能达到最优。

#### （8）上线切换

OceanBase数据库完成性能测试后，需要原有数据库清空，重新创建数据库，已完成生产数据迁移。数据迁移方式建议采用：全量+校验+增量+全量校验+反向的方式，具体操作为:

1. 全量：即全量迁移，将原有数据库的某个时间节点的数据全量迁移到OceanBase数据库上；
2. 校验：校验OceanBase数据库与原数据库数据是否一致；
3. 增量：即增量迁移，上线当天，停止原数据库服务，将原数据库从全量迁移后的数据到上线当天的数据，增量迁移到OceanBase数据库上。
4. 全量校验：增量迁移后，校验OceanBase数据库与原数据库数据是否一致；
5. 方向：利用OMS工具将OceanBase数据库的新数据同步到原数据库中，防止OceanBase数据库出现意外情况，无法提供服务事，系统可以切换至原数据库，继续应用。

特别说明，上线前要制定如下内容：

①制定迁移方案、回切方案

②提前进行生产演练

③建议制定切换的行事历

④制定交付手册

以保证系统顺利切换。

#### （9）持续监控

监控OceanBase数据库是否满足系统要求，特别是慢SQL情况；同时监控OceanBase数据库与原数据库的数据同步效果。OceanBase持续运行一段时间，无问题后，下架原有数据库。