DTCC 2010-2022

第十三届中国数据库技术大会

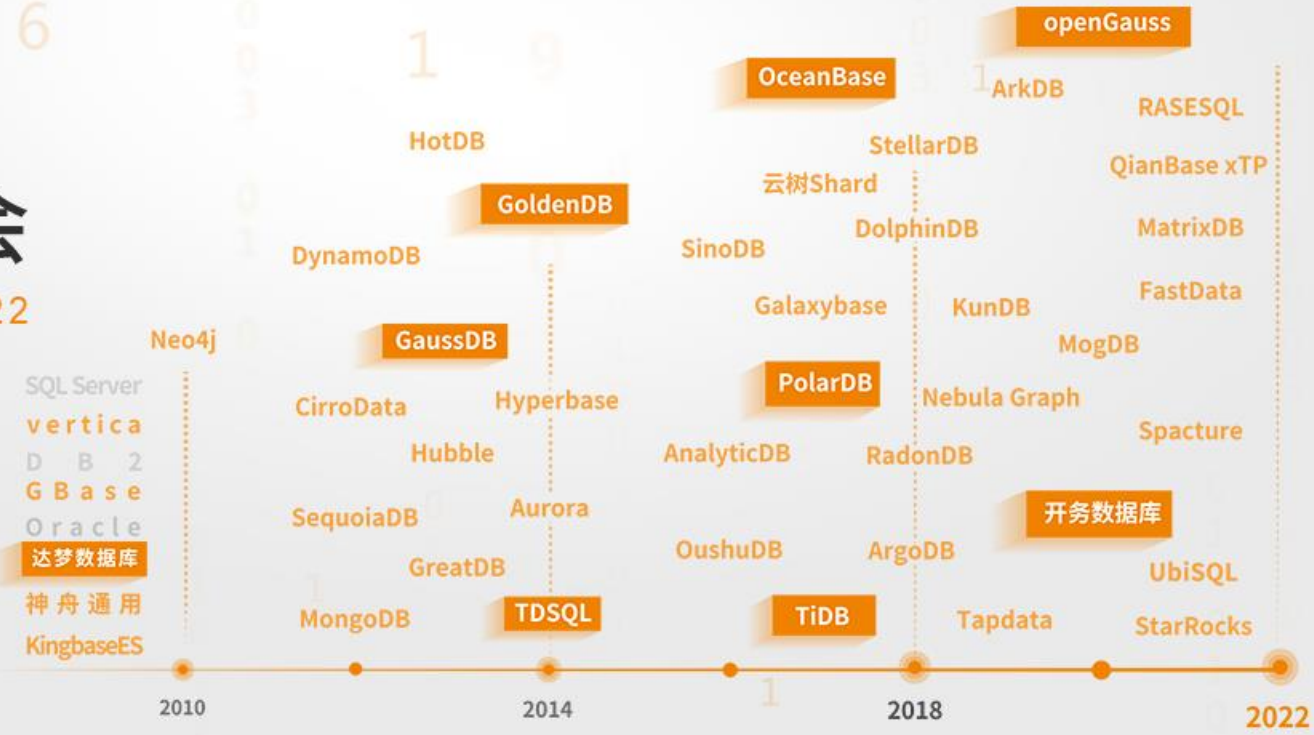
DATABASE TECHNOLOGY CONFERENCE CHINA 2022

数据智能 价值创新

IT168.com ChinaUnix.net ITPUB

线上直播 | 2022/12/14-16

数据来源：数据库产品上市商用时间



金融分布式数据库的应用与实践

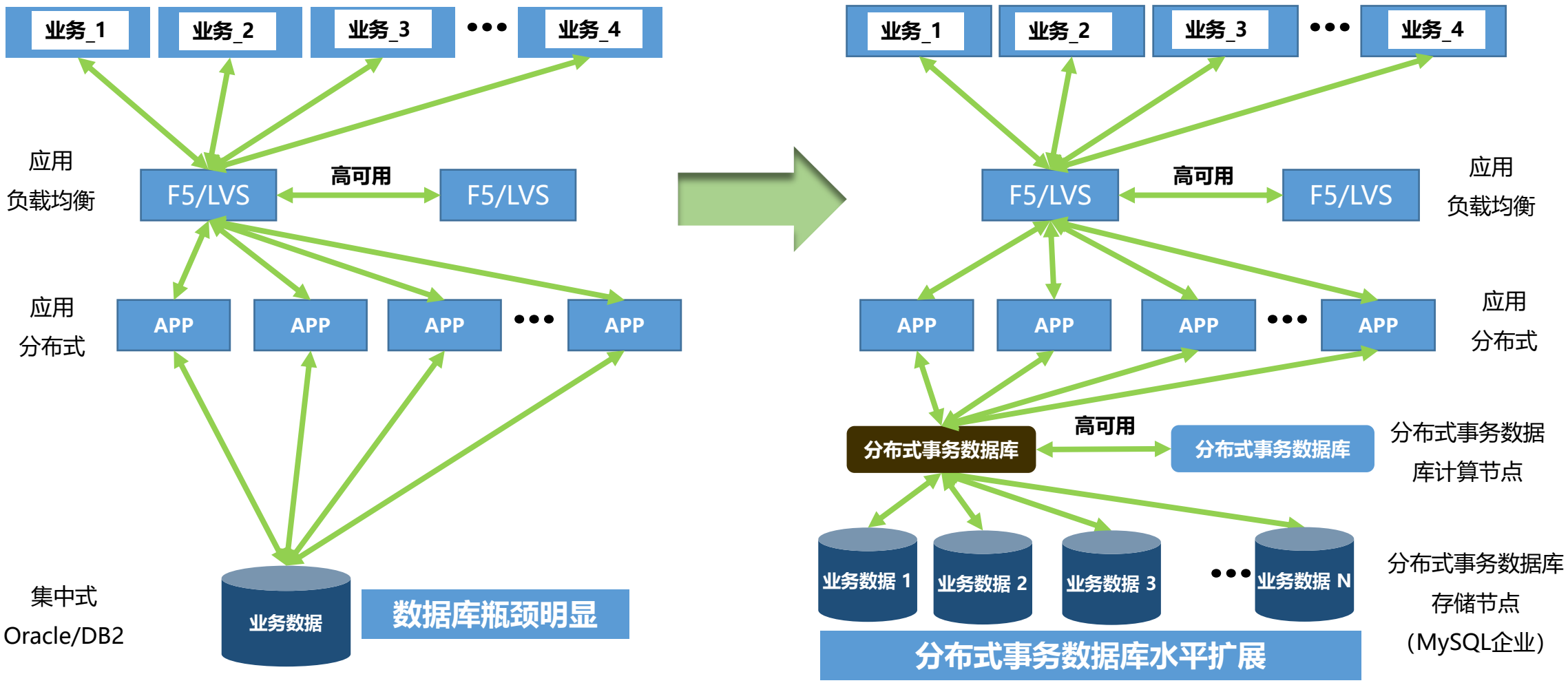
林 春

太平洋保险数智研究院

首席数据库专家

QQ:1819442969

分布式数据库对金融业的意义

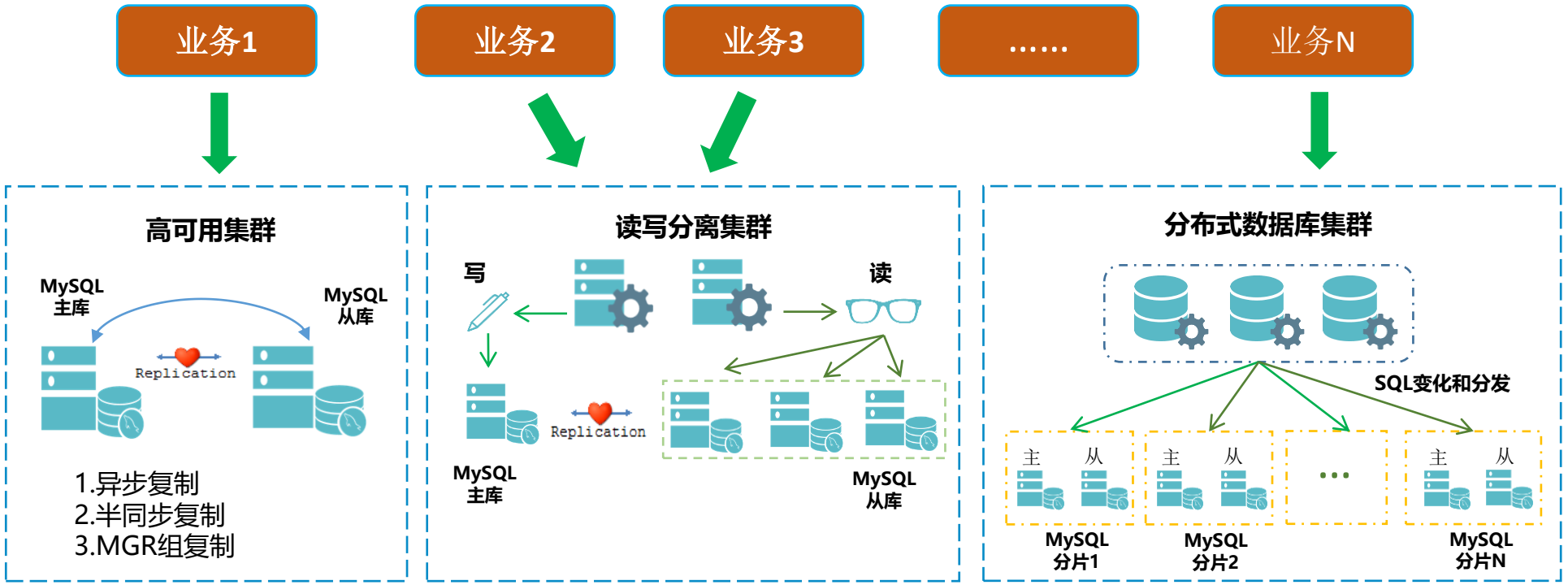


Oracle/DB2为多进程模式，**MySQL**为单进程多线程模式，分布式架构更好支撑海量连接、海量并发、海量吞吐的业务场景。

分布式数据库产品现状

- **分布式数据库产品尚需金融业务场景打磨，尚无100%成熟的分布式数据库产品。**主要体现在数据库产品BUG数量较多、金融业非功能性需求适配、数据库开发及运维管理平台不够友好和数据库周边工具功能不足等方面。存量Oracle数据库迁移改造存在痛点，主要体现在应用适配评估工具需提升、存储过程改造工作量较高、迁移工具性能及稳定性需提升等。
- **不同的分布式数据库各有优劣，需根据具体业务场景适配。**面上主流的分布式数据库分为基于proxy的分布式数据库和原生分布式数据库。原生分布式数据库产品具有对应用侵入性少等优点，但是比起基于proxy的分布式数据库需要有更多的代码研发和改动，数据库产品可靠性方面需要更多的验证；而基于proxy的分布式数据库也面临如何将底座集中式数据库新版本的新特性整合的问题。
- **分布式数据库产品处于可用向好用的演进中间状态。**后期分布式数据库厂商的服务及支持、数据库生态、数据库产品底层掌控能力、软硬件成本、服务成本及应用改造成本需要关注。
- **分布式数据库产品呈现多条技术路线同时快速发展的态势。**在分布式数据库选型上和使用上需要采取高度兼容的策略，结合业务场景需求，在选择通用性兼容及生态较好的分布式数据库产品后，需要考虑在应用开发中简化、优化、标准化SQL，在适配分布式数据库特点、提升应用性能的同时，做到技术产品灵活可控。

分布式数据库适用条件



- ✓数据量较小，单库控制在1TB以内
- ✓单表记录数控制在5000W以内

- ✓数据量较小，单库控制在1TB以内，单表记录数控制在5000W以内
- ✓读写比差别大，可达到5:1以上

- ✓数据量较大，数据库单库数据量1TB以上，单表记录数超过1亿条
- ✓高并发写入，要求较高TPS

分布式数据库选型考量

分布式数据库的兼容性及可移植性
支持国产芯片、国产操作系统、国产处理器、国产服务器及国产中间件

分布式数据库的功能性
主要考察分布式数据库的基础功能、 SQL 语法标准、数据库对象支持情况、单库事务及分布式事务支持情况、数据切片算法等

分布式数据库的可扩展性
自动化的扩缩容操作、水平扩容对业务的无感知、扩容支持自动化的数据重分布

分布式数据库的性能
分布式数据库读写性能、TPCC、多表关联 JOIN 查询的性能等

分布式数据库的维护性
故障分析、性能监测、故障自治自愈

分布式数据库的可靠性
7*24 小时稳定性测试、模拟故障节点测试、断电断网测试等

分布式数据库的易用性
运维管理支持可视化方式、自动化运维及智能化分析、自动化巡检等

分布式数据库的安全性
数据库鉴权、访问控制、安全审计、高危 SQL 拦截、数据传输加密、备份加密及等保等级

信创数据库工作推进思路

1. **建设相关工具、平台**
   实现信创数据库效率大幅提升降低应用改造难度与成本。
2. **建设信创数据库专业队伍**
   聚焦信创数据库核心攻坚，培养全栈能力，建设专业队伍。
3. **知识库体系化建设**
   建设信创数据库知识库体系，固化我司最佳实践
4. **制定信创数据库培训策略**
   降低信创数据库学习曲线，快速提升内部人员相应技能
5. **制定信创数据库技术支持策略**
   提供覆盖架构、开发、运维的全生命周期的数据库技术支持
6. **行业动态研究**
   跟进行业最新动态，研究新技术在我行应用的可行性

信创数据库知识体系构建

建**立一套完备的、先进的信创数据库知识结构体系，是培养太保信创数据库人才队伍、构建数据库服务能力的重中之重。**构建时需要兼顾太保信创整体目标，同时又要融入攻坚过程积累的实战经验 。

	迁移规划 迁移规划和准备	迁移验证 迁移改造和验证	上线切换 切割上线和风险保障	运维保障 维护系统稳定运行
架构设计类	• 架构及产品特点深度剖析 • 数据库选型最佳实践 • 存储引擎高级技术	——	• 项目实战谈数据迁移方案	——
交付技术类	• 项目实战谈架构设计	• 信创流程详解 • 信创最佳实践 • 数据迁移工具及最佳实践 • PL引擎的设计和最佳实践	• 开发规范及最佳实践 • 兼容性改造最佳实践 • 上线流程详解	• 参数最佳配置 • 安全及权限
性能优化类	• 内存原理 • 执行计划原理及最佳实践 • 索引的设计及最佳实践	• 性能优化与案例分享 • SQL引擎、事务高级技术 • 以项目实战谈信创最佳实践	——	• 分布式事务原理 • 内存管理及内存分配
运维技术类	• 高可用高级技术 • 容灾、备份恢复高级技术	——	• 回退的方法及实践	• 运维技术及最佳实践 • SQL监控和优化 • 运维、监控与异常处理 • 高级日志分析技术

*红色字体部分为高级技术

太保OB应用改造评估工具-“指南针”

能够依托知识库沉淀对存储过程代码进行扫描分析，并初步给出问题原因、代码位置。

评估项全面、有效，提升了项目组问题排查的效率，从而降低应用改造的成本。

能够辅助识别冗余大表、冗余索引，有助“数据库减负”。

分布式数据库迁移前数据库画像脚本功能

- 前20位大表每日更新、插入、删除记录数（按更新记录数排序）、记录总数、平均记录长度、表空间大小、索引空间总大小
- 更新排名前20位表每日更新、插入、删除记录数、记录总数、平均记录长度、表空间大小、索引空间总大小

- 用以抓取高负载时间段大事务SQL，便于迁移到OB或TD前提前优化
- 以小时为维度，一天什么时间段日志负载最大
- 以周为维度，一周周几最繁忙
- 以月为维度，一个月那天最繁忙

- 用于识别未使用索引，进行瘦身
- 实例启动以来，从未使用过的索引
- 超过5个索引的表识别
- 未带主键表识别
- 全局索引识别

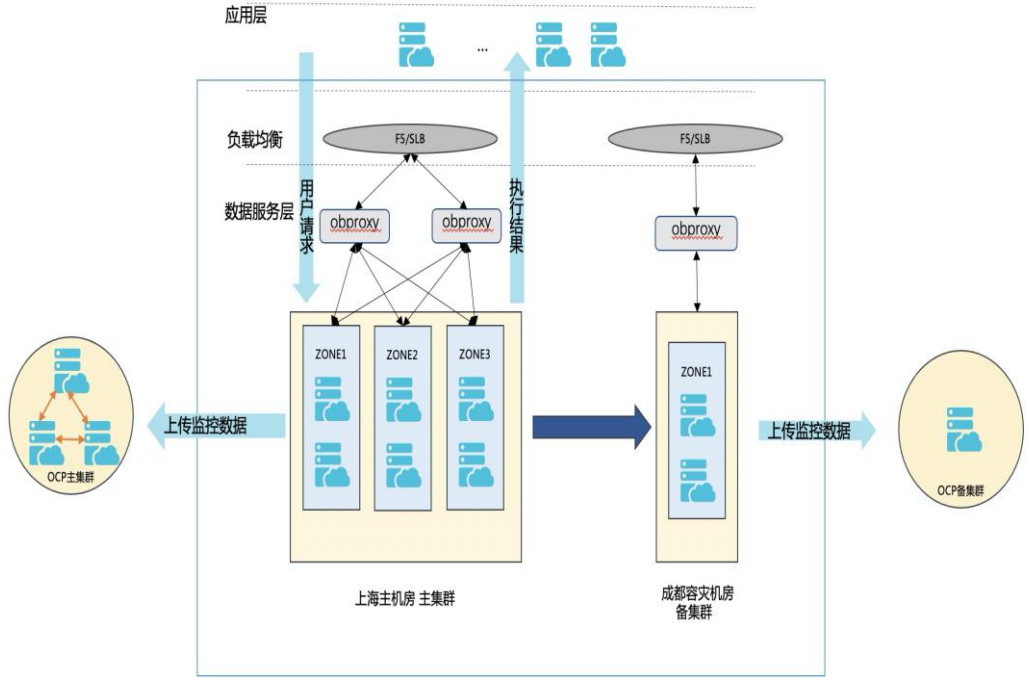
- 无效索引识别
- 失效对象识别

- 长查询画像
- 高CPU开销SQL画像
- 高IO开销SQL画像
- 高排序开销SQL画像

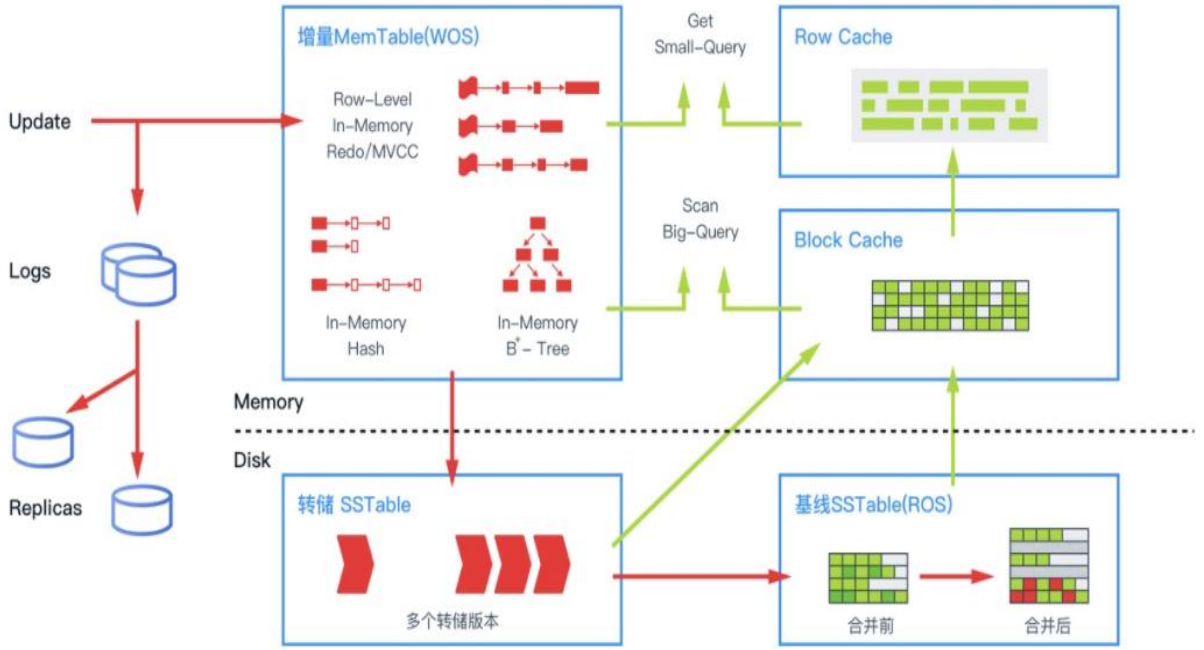
- 识别实例启动以来未修改过的表
- 识别实例启动以来未访问过的表
- 识别仅有插入、没有更新、修改的大表

Oceanbase架构及性能优化关键点

- OCP集群可以管理多个OB集群，管理OBserver节点总数上限约为200个。单个OB集群OBserver节点总数不建议过多，通常不超过20个。OceanBase数据副本是以分区为单位。
- OceanBase 数据库存储引擎基于 LSM-Tree 架构，基线数据和增量数据分别保存在磁盘（SSTable）和内存（MemTable）中。对数据的修改都是增量数据，只写内存。读的时候，数据可能会在内存里有更新过的版本，在持久化存储里有基线版本，需要把两个版本进行合并，获得一个最新版本。



Oceanbase系统架构



Oceanbase存储架构

Oceanbase架构及性能优化关键点（续）

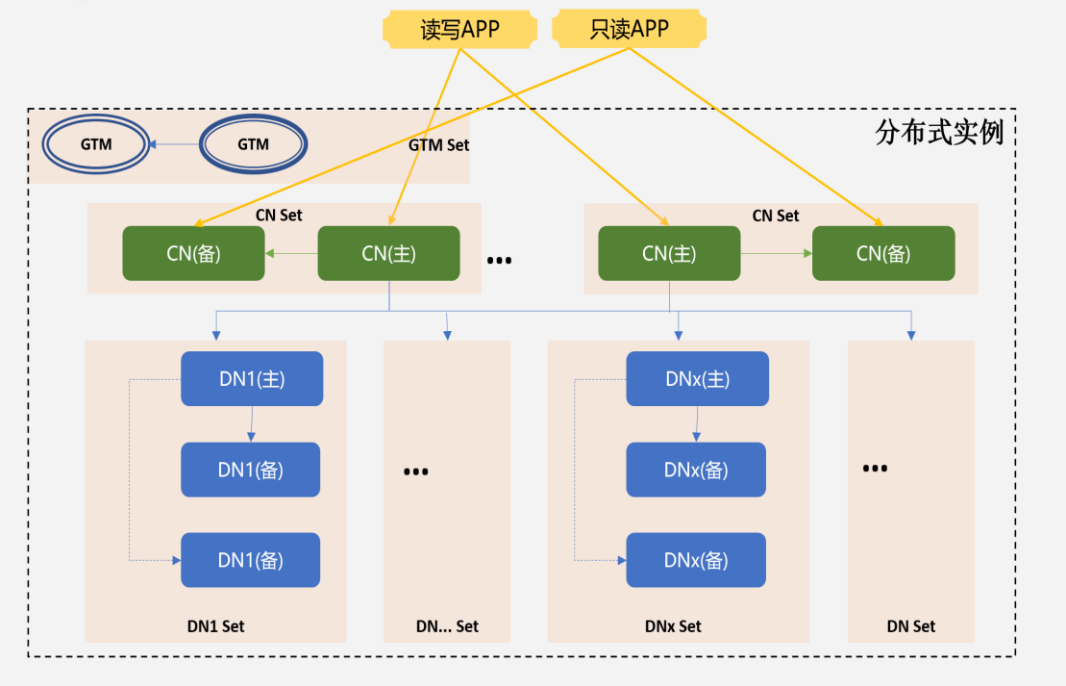
- 设计集群时核心业务系统集群应独立部署，控制集群租户数量、合理利用分区优化数据生命周期管理，以最小化容灾、备份数据集，减少转储次数进而减少非计划合并次数，避免性能毛刺现象。
- 非关键、低容量、**没有频繁DML操作**的数据库可共用一个共享集群，租户互相隔离，集群采用高配以节省软件成本。
- freeze_trigger_percentage决定转储的内存阈值，minor_freeze_times参数决定租户的转储多少次触发合并，在集群规划时，租户的负载、memstore内存大小需要考虑，以避免由于个别租户频繁转储导致整个集群合并，导致其他租户成为受害者，尽可能避免写放大对性能影响。
- OceanBase应用总设计原则是尽可能让数据访问、操作在内存中完成，减少转储造成读延迟毛刺，每天定期合并释放memstore内存及删除记录空间，最大限度减少写放大、读放大。租户转储频度需要巡检中检查，并评估内存大小、内存阈值设置、慢SQL开销，从gv$memstore可以查询租户转储相关信息。
- 优化SQL避免对不必要数据操作，否则会导致转储增多从而读取时延出现明显增大；优化SQL避免请求读取过多SSTable；尽可能避免对相同记录做频繁更新，否则查询会在事务链表读取更多事务节点，目前OB底层读操作时，当链表的长度大于18时，写操作时，当链表的长度大于6时会触发compact。

Oceanbase架构及性能优化关键点（续）

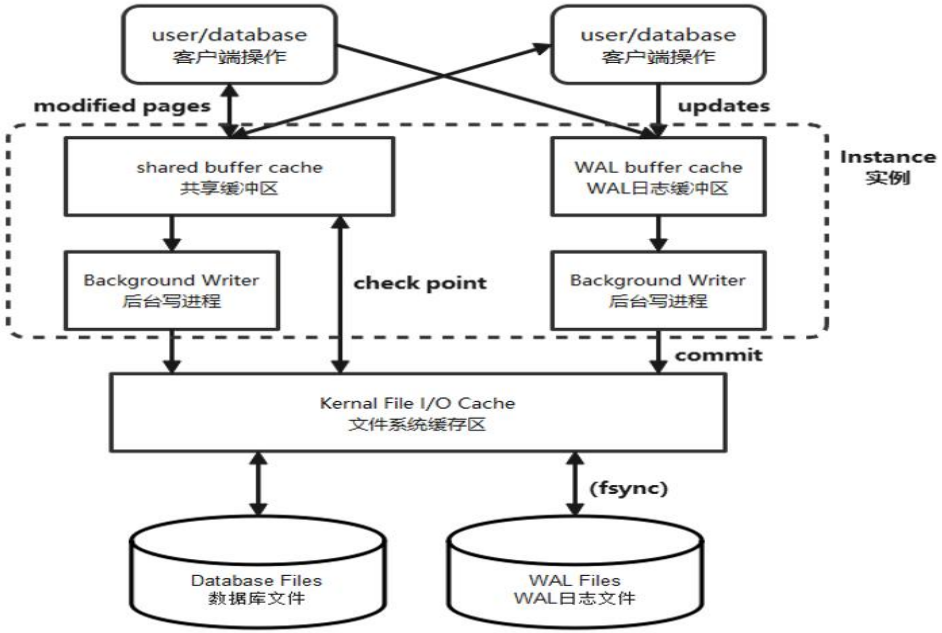
- OceanBase合并是以集群为维度，需要控制集群容量规模，以避免合并耗时过长，MemStore内存释放不及时，导致MemStore满而数据写入失败。
- major_freeze_duty_time参数控制定时合并时间，在关键业务日期或者重大变更时段，需要调整定时合并时间窗口或者暂停合并。
- OMS迁移数据前，建议先做合并，以最大腾出memstore内存空间用于迁移。
- 设计时需考虑尽可能同一租户的表或分区数据放在相同OBserver上，数据访问或修改尽可能避免跨OBserver。
- OceanBase备份是以整个集群为维度，不能单独对某个租户，因此需要集群规划需要考虑备份时间窗口能否对集群完成备份，如果备份时间过长，需要在白天业务时间备份，需要考虑对备份进行流控，例如：使用backup_concurrency参数控制数据备份的并发参数，使用以下语句控制备份流量backup net limit for whole clusterRange: [0M, max)；。
- 目前OceanBase分布式死锁仅能通过平台手工检测死锁，死锁无法通过图算法自动改出死锁，需要依赖ob_trx_lock_timeout参数，默认参数为-1，**建议生产环境设置该参数为10~30秒**。
- OceanBase大事务需要做拆分。对相同记录频繁更新后，查询需要有额外操作，会有一定读延迟。

TDSQL架构及性能优化关键点

- CN节点负责数据的分发和查询规划，每个CN节点都提供相同的数据库视图;DN节点处理存储本节点相关的元数据，每个节点还存储业务数据的分片。此外，DN节点负责完成执行协调节点分发的执行请求;GTM负责管理集群事务信息，同时管理集群的全局对象，比如序列等。TDSQL-PG通过GTM保证读一致性，TDSQL-PG使用逻辑时钟替代快照优化了带宽和CPU开销。
- TDSQL-PG的分布键选择对性能非常关键，交易场景不带分布键会检索所有DN节点数据，性能会有明显下降，TDSQL-PG分布键需要尽可能避免跨节点检索、操作数据,通常从主键、表连接字段、where条件中筛选度较好的字段中选取。



TDSQL-PG系统架构



DN、CN节点数据库系统架构

分布式数据库使用建议

01 SQL类型及关键信息梳理及SQL改造

对数据类型、SQL特殊类型例如hint、外键、触发器、sequence、db link、存储过程等进行梳理；对业务系统中大表、访问频繁的表、表连接条件、where搜索条件进行梳理。

02 表设计的选择

基于proxy的分布式数据库需根据业务类型特点、数据量，选择分表、广播表、单表；原生分布式数据库需注意表组设计、全局索引设计、复制表设计、分区键选择等。

03 分片方案设计

尽可能使分片数据在数据节点均匀分布；避免无差别滥用分片；分片键字段必须是主键以及所有唯一索引的一部分；不要更新分片键字段的值；访问的数据尽量包含分片键字段，否则不带分片键字段的SQL语句会路由到所有节点，将消耗较多资源。

04 数据迁移方案的制定及演练

数据迁移方案制定及数据迁移模拟演练；数据要能实现异构数据库回写；并发对增量数据记录做比对

基于PG的分布式数据库表膨胀优化建议

- PG未使用UnDO表空间实现MVCC，在更新时对原记录做删除标记，然后在相同数据文件再插入新的记录实现。对于有频繁更新的表，需要关注“表膨胀问题”。
- 更新频繁的表如果平均记录长度较长或包含clob字段，可以考虑应用层面对表做拆分。
- 在Oracle数据库迁移到TDSQL前，需要识别前20位大表中更新频繁的表以及更新频繁程度排在前20位的表，在迁移之前将相应表fillfactor设为80（默认100）
- 在PG监控中，监控阈值评估时需要增加对表膨胀的监控（以天为维度）
- 在做了大批量DML操作后，随即执行vacuum analyze操作
- **如果系统存在频繁更新的大表、长查询、长事务，空间回收问题也需要关注**，Vacuum full需要固定较长维护窗口及两倍表的空间。

DML类型	并发tps	维护时间窗口	空间回收清理分析
查询为主，DML较少			不存在“表膨胀”问题
DML主要以insert为主，仅存在少量update			不存在“表膨胀”问题
查询为主，存在一定量DML操作	并发量不大	非7*24应用，有足够维护时间窗口	每天有维护窗口进行VACUUM FULL操作
DML较多，存在长查询、长事务	并发量较大	非7*24应用，有足够维护时间窗口	每天有维护窗口进行VACUUM FULL操作
DML较多，存在长查询、长事务	并发量不大	7*24场景，存在较为空闲的维护窗口	定期在非业务高峰期进行VACUUM FULL，需对数据量及VACUUM FULL时间做评估
DML较多，存在长查询、长事务	并发量较大	7*24场景，存在业务量较小时间段窗口	需要进行评估，避免VACUUM FULL操作影响业务
DML较多且数据量较大，存在长查询、长事务	并发量较大	7*24场景，维护窗口时间较短	需对应用做评估，使用需要对应用做一定改造

分布式数据库优化案例一

问题：Oracle侧19秒，分布式数据库侧超时未跑出结果

陷阱：/*+parallel(6)*/ 分布式数据库侧3分钟跑出结果

```
select  d.comp_contact_cd || '|' || b.ext_policy_no || '|' ||
        a.case_report_no || '|' || a.case_report_no || '|' || c.sms_type_cd || '|' ||
        e.code || '|' || to_char(c.act_start_dttm, 'yyyy-mm-dd hh24:mi:ss') || '|'
  from t_cc_l2_pi_vehicle_claim a
  left join t_cc_l0_srt b  on b.srt_id = a.srt_id
  left join t_cc_l2_sms_send c   on c.rel_obj_id = b.srt_id
  left join t_cc_l2_organ d   on d.organ_party_id = b.sub_company_id
  left join v_4s_info_list e   on e.name = a.fours_name
 where trunc(c.act_start_dttm) = to_date('20220401', 'yyyy-mm-dd');
```

分布式数据库优化案例一-优化前执行计划

执行计划如下：

```
==========================================================================
|ID|OPERATOR                  |NAME          |EST. ROWS|COST    |
--------------------------------------------------------------------------
|0 |HASH RIGHT OUTER JOIN      |              |21809    |10776678|
|1 | SUBPLAN SCAN              |V_4S_INFO_LIST|1574     |612748  |
|2 |  MERGE GROUP BY           |              |1574     |612724  |
|3 |   NESTED-LOOP JOIN        |              |12329    |611942  |
|4 |    TABLE SCAN             |TV            |1574     |569665  |
|5 |    TABLE SCAN             |EV            |8        |25      |
|6 | HASH RIGHT OUTER JOIN     |              |21809    |10148423|
|7 |  TABLE SCAN               |D             |21849    |8452    |
|8 |  HASH RIGHT OUTER JOIN    |              |21809    |10114605|
|9 |   TABLE SCAN              |C             |758060   |293222  |
|10|   HASH OUTER JOIN         |              |4361791  |7029401 |
|11|    TABLE SCAN             |A             |502323   |194301  |
|12|    PX COORDINATOR         |              |4471269  |3643345 |
|13|     EXCHANGE OUT DISTR    |:EX10000      |4471269  |1729508 |
|14|      PX PARTITION ITERATOR|              |4471269  |1729508 |
|15|       TABLE SCAN          |B             |4471269  |1729508 |
==========================================================================
```

分布式数据库优化案例一

优化思路：

```
select  d.comp_contact_cd || '|' || b.ext_policy_no || '|' ||
      a.case_report_no || '|' || a.case_report_no || '|' || c.sms_type_cd || '|' ||
      e.code || '|' || to_char(c.act_start_dttm, 'yyyy-mm-dd hh24:mi:ss') || '|'
 from t_cc_l2_pi_vehicle_claim a
 left join t_cc_l0_srt b   on b.srt_id = a.srt_id
 left join t_cc_l2_sms_send c   on c.rel_obj_id = b.srt_id
 left join t_cc_l2_organ d   on d.organ_party_id = b.sub_company_id
 left join v_4s_info_list e   on e.name = a.fours_name
where c.act_start_dttm between to_date('20220331', 'yyyy-mm-dd') and to_date('20220402', 'yyyy-mm-dd');
```

分布式数据库优化案例一-优化后执行计划

优化后：分布式数据库侧**3秒**跑出结果，为Oracle侧用时的**15.79%**

```
==========================================================================
|ID|OPERATOR                    |NAME                  |EST. ROWS|COST    |
--------------------------------------------------------------------------
|0 |HASH RIGHT OUTER JOIN       |                      |35470    |11158762|
|1 | SUBPLAN SCAN               |V_4S_INFO_LIST        |1574     |5228166 |
|2 |  HASH GROUP BY             |                      |1574     |5228142 |
|3 |   HASH JOIN                |                      |12329    |5221937 |
|4 |    TABLE SCAN              |TV                    |1574     |569665  |
|5 |    TABLE SCAN              |EV                    |5765961  |2230301 |
|6 | HASH RIGHT OUTER JOIN      |                      |35470    |5906226 |
|7 |  TABLE SCAN                |D                     |21849    |8452    |
|8 |  MERGE JOIN                |                      |35470    |5863532 |
|9 |   TABLE SCAN               |A                     |502323   |194301  |
|10|   SORT                     |                      |36360    |5644101 |
|11|    HASH JOIN               |                      |36360    |5545652 |
|12|     TABLE SCAN             |C(I_T_CC_L2_SMS_SEND_N2)|4293   |16593   |
|13|     PX COORDINATOR         |                      |4471269  |3643345 |
|14|      EXCHANGE OUT DISTR    |:EX10000              |4471269  |1729508 |
|15|       PX PARTITION ITERATOR|                      |4471269  |1729508 |
|16|        TABLE SCAN          |B                     |4471269  |1729508 |
```

分布式数据库优化案例二

问题描述：

hibernate框架版本为3.3.2.GA版本，调用spring-data-jpa进行update和query时访问Oracle系统视图v$session出错。

困难点：

数据库监控工具无法定位问题SQL，应用侧也未在应用代码及框架代码中发现问题SQL位置。

解决思路：

1、在Oracle源库通过sql跟踪，抓取问题SQL如下：

```
SELECT MACHINE, TERMINAL, SYS_CONTEXT('USERENV', 'IP_ADDRESS'),
  SYS_CONTEXT('USERENV', 'OS_USER')
FROM
 V$SESSION A, (SELECT * FROM V$MYSTAT WHERE ROWNUM = 1) B WHERE A.SID = B.SID
  AND ROWNUM = 1
```

2、分析应用SQL目的：是用作审计，将登录当前用户的用户名、ip记录。设法将语义层面报错转换为SQL引擎层面抛出异常。

3、设法根据问题SQL字段，构造同名伪表、伪视图，但是不放任何数据。

4、调用框架，骗过优化器，触发SQL引擎层抛出NO_DATA_FOUND异常同时提供模块名称，定位是一个触发器模块。

5、改写对应触发器逻辑，提供对应解决方案。


openGauss
OceanBase
ArkDB
RASESQL
HotDB
StellarDB
QianBase xTP
云树Shard
GoldenDB
DolphinDB
MatrixDB
DynamoDB
SinoDB
FastData
Galaxybase
KunDB
SQL Server
vertica
D B 2
GBase
Oracle
达梦数据库
Spacture
开务数据库
UbiSQL
神舟通用
TDSQL
TiDB
Tapdata
StarRocks
KingbaseES
2010
2014
2018

THANKS