dbaplus

Newsletter

2021年1月版

目录

DB-Engines 数据库排行榜

以下取自 2020 年 1 月的数据，具体信息可以参考 http://db-engines.com/en/ranking/，数据仅供参考。

361 systems in ranking, January 2021

Rank Jan 2021	Rank Dec 2020	Rank Jan 2020	DBMS	Database Model	Score Jan 2021	Score Dec 2020	Score Jan 2020
1.	1.	1.	Oracle	Relational, Multi-model	1322.93	-2.66	-23.75
2.	2.	2.	MySQL	Relational, Multi-model	1252.06	-3.40	-22.60
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	1031.23	-6.85	-67.31
4.	4.	4.	PostgreSQL	Relational, Multi-model	552.23	+4.65	+45.03
5.	5.	5.	MongoDB	Document, Multi-model	457.22	-0.51	+30.26
6.	6.	6.	IBM Db2	Relational, Multi-model	157.17	-3.26	-11.53
7.	7.	↑ 8.	Redis	Key-value, Multi-model	155.01	+1.38	+6.26
8.	8.	↓ 7.	Elasticsearch	Search engine, Multi-model	151.25	-1.24	-0.19
9.	9.	↑ 10.	SQLite	Relational	121.89	+0.21	-0.25
10.	10.	↑ 11.	Cassandra	Wide column	118.08	-0.76	-2.59
11.	11.	↓ 9.	Microsoft Access	Relational	115.33	-1.41	-13.24
12.	12.	↑ 13.	MariaDB	Relational, Multi-model	93.79	+0.18	+6.34
13.	13.	↓ 12.	Splunk	Search engine	87.66	+0.66	-1.01
14.	14.	↑ 15.	Teradata	Relational, Multi-model	72.59	-1.24	-5.70
15.	↑ 16.	↑ 25.	Microsoft Azure SQL Database	Relational, Multi-model	71.36	+1.87	+43.16
16.	↓ 15.	↓ 14.	Hive	Relational	70.43	+0.16	-13.81
17.	17.	↓ 16.	Amazon DynamoDB	Multi-model	69.14	+0.01	+7.11
18.	18.	↑ 20.	SAP Adaptive Server	Relational	54.61	-0.27	+0.02
19.	19.	↑ 22.	Neo4j	Graph	53.79	-0.85	+2.13
20.	↑ 21.	↓ 17.	Solr	Search engine	52.48	+1.24	-4.08
21.	↓ 20.	↓ 19.	SAP HANA	Relational, Multi-model	50.87	-1.63	-3.82
22.	22.	↓ 18.	FileMaker	Relational	47.39	-0.31	-7.72
23.	23.	↓ 21.	HBase	Wide column	46.28	-0.64	-7.06
24.	24.	↑ 26.	Google BigQuery	Relational	36.00	+0.24	+9.24
25.	25.	↓ 24.	Microsoft Azure Cosmos DB	Multi-model	32.97	-0.57	+1.46

DB-Engines 排名的数据依据 5 个不同的因素：

- ✓ Google 以及 Bing 搜索引擎的关键字搜索数量
- ✓ Google Trends 的搜索数量
- ✓ Indeed 网站中的职位搜索量
- ✓ LinkedIn 中提到关键字的个人资料数
- ✓ Stackoverflow 上相关的问题和关注者数

新闻资讯

1、2020 年 11 月 24 日，国际权威调研机构 Gartner 公布 2020 年度全球数据库魔力象限评估结果，作为中国科技公司代表，阿里云首次挺进全球数据库第一阵营——“领导者”象限，这也是中国数据库 40 年来首次进入全球顶级数据库行列；此外，腾讯云、华为云进入了“特定领域者”象限。

2、2020 年 12 月 2 日，Kubernetes 官方发布公告，宣布自 v1.20 起放弃对 Docker 的支持，届时用户将收到 Docker 弃用警告，并需要改用其他容器运行时。但 Docker 作为容器镜像构建工具的作用将不受影响，用其构建的容器镜像将一如既往地在集群中与所有容器运行时正常运转。

3、2020 年 12 月 8 日，CentOS 项目宣布，CentOS 8 将于 2021 年年底结束维护，为其接班的是 CentOS Stream，CentOS Stream 作为 RHEL 的上游（开发）分支在 CentOS 8 结束维护后会继续更新。而 CentOS 7 也将在其生命周期结束后停止维护。换而言之，“免费”的 RHEL 以后没有了。

4、2021 年 1 月 15 日，Elastic 公司宣布即将变更 Elasticsearch 和 Kibana 的其中一项开源许可协议——将 Apache License 2.0 变更为双授权许可，即 SSPL + Elastic License。在不违反许可协议的前提下，变更后用户仍可自由选择使用满足自己需求的源代码或发行版。此次变更只影响使用到 Apache License 2.0 的源代码，与 Apache License 2.0 无关的部分像过去一样保持不变。例如 Elasticsearch 和 Kibana 的默认发行版会继续在 Elastic License 许可下发布，用户可继续免费下载和使用。

RDBMS 家族

Oracle 将推出 21c，是全球唯一的融合数据库

根据Oracle的产品策略， 2021年将会推出Oracle 21c，21c重要新特性在于进一步强化融合数据库理念。强调3M功能——多模、多工作负载、多租户，是全球唯一的融合数据库（在本地与云端均能部署）。

一、Oracle 21c重要新特性摘要

1、原生的区块链支持 - Native Blockchain Tables，原生区块链表类似于标准表，允许SQL插入且插入的行以加密方式链接。用户可以选择对行数据进行签名，来杜绝身份欺诈。轻松集成到应用中参与其它表的事务和查询。

2、新的多租户Data Guard - 可插拔数据库（PDB）级别的灾难保护，可以进行应用级别的读写分离和容灾切换，支持PDB级别的ADG。

3、AutoML：自动化的机器学习 - 非专家级用户也可以充分利用机器学习的优势。AutoML将推荐最合适的算法，自动选择特性并调优超参数，从而大幅提升模型准确性；以前是收费选件，现在免费。

4、支持持久化内存，提供微秒级I/O响应。绕过网络和IO软件堆栈，将延迟降低10倍。在系统崩溃时不会丢失数据。

5、JSON加速与灵活性：支持二进制格式的JSON；更新密集型或扫描密集型操作速度数倍增长。

6、In-Memory支持混合查询，在19c之前数据如果分布在列存储区和行存储区，将无法使用In-Memory特性。在21c中，支持混合查询，查询性能超过10倍提升。

7、自动化的In-Memory管理 - 通过数据库内置的机器学习算法将自动判断需要将哪些对象加入或驱逐出In-Memory的列式存储，并且数据库可以进一步的自动压缩较少访问的内存列数据。

8、Sharding增强：能从多个现有数据库创建分片数据库，简化基于同一应用程序模式的多个数据库到分片架构的迁移，能够运行跨分片查询的联邦数据库。

二、2020年甲骨文重要发布回顾

在2020年甲骨文发布了最新的Exadata X8M，Oracle Exadata X8M是业内首个采用了英特尔®傲腾™DC持久性内存和创新的数据库RDMA技术，与上一版本相比I/O吞吐量提高了2.5倍，I/O延迟降低了90% 。Exadata X8M目前拥有70多项独有功能，其他任何能够运行Oracle数据库的硬件都不具备这些功能。

Oracle提供了多种方式使客户可以使用到Exadata的服务，包括了Exadata数据库一体机，Exadata公有云服务以及Exadata公有云私有化部署（ExaCC）。

另外，甲骨文还发布了 Oracle 数据库保护的最佳实践——ZDLRA 最新版本 X8，零数据丢失恢复一体机数据保护解决方案（简称恢复一体机）。经过多年的发展，很多企业都使用上 ZDLRA 来保护 Oracle 数据库。

MySQL 8.0 2020 年度重大更新

一、2020 年度 MySQL 总体回顾

2020 年，MySQL 8.0 发布了 4 个版本 8.0.19~8.0.22，在功能和稳定性方面已经具备一定的用户基础和优势，可以考虑在生产环境试用/使用。

另外，MySQL 5.6 将于 2021 年 2 月停止更新，结束其生命周期（EOL），也就是说，2021 年 2 月以后，MySQL 团队将不会再为 5.6 系列版本的 MySQL 提供任何补丁，反之留给 MySQL 5.7 的时间也较紧张，目前截止时间是 2023 年 10 月。

在功能方面，自 MySQL 8.0.18 版本发布 hash join，到 8.0.20 版本已经可以使用 hash join 代替 BNL，现在支持半连接、反连接、外连接，改进很明显，对 MGR 的细节也进行了很多功能改造，在安全性、可用性方便有较大提升。

尤其是在 12 月 2 日 MySQL 团队发布了 MySQL Database Service with Analytics Engine，将 OLTP/OLAP 处理合二为一，同时在性能、安全和成本方面有了很大提升，让 MySQL 饱受诟病的 OLAP 领域带来转机。

数据库 90%的性能问题由于 SQL 引起，线上 SQL 的执行快慢，直接影响着系统的稳定性。以下是 MySQL 8.0 年度重大版本的特性更新盘点和总结：

二、MySQL 8.0 针对业务研发关心的特性更新盘点

优化器改进

1、子查询 update/delete 采用 Semi Join 半连接作了优化

MySQL 的子查询一直以来以性能差著称，所以解决的方案是用 join 关联查询代替子查询。

通常情况下，我们希望由内到外，先完成内表里的查询结果，然后驱动外查询的表，完成最终查询，但是 MySQL 5.5 会先扫描外表中的所有数据，每条数据将会传到内表中与之关联，如果外表很大的话，那么性能上将会很差。

MySQL 5.6 版本里，子查询终于有了强劲的优化，这意味着不改变原有 SQL 的情况下，通过 MySQL 内部的优化器把子查询改写成为 join 关联查询。

但是，Semi Join 半连接优化仅仅是针对 select 查询的，针对 update/delete 性能仍旧很差。

下面通过实验来验证下，执行语句 explain extended update t1 set name='aa' where id in (select id from t2);

从下图可以看到 Semi Join 半连接优化失效了，仍旧还是先查外表再关联内表。

```
mysql> explain extended update t1 set name='aa' where id in (select id from t2 );
+----+--------------------+-------+-----------------+---------------+---------+---------+------+------+----------+-------------+
| id | select_type        | table | type            | possible_keys | key     | key_len | ref  | rows | filtered | Extra       |
+----+--------------------+-------+-----------------+---------------+---------+---------+------+------+----------+-------------+
|  1 | PRIMARY            | t1    | index           | NULL          | PRIMARY | 4       | NULL | 2    |   100.00 | Using where |
|  2 | DEPENDENT SUBQUERY | t2    | unique_subquery | PRIMARY       | PRIMARY | 4       | func | 1    |   100.00 | Using index |
+----+--------------------+-------+-----------------+---------------+---------+---------+------+------+----------+-------------+
2 rows in set (0.00 sec)
```

从 MySQL 8.0.21 开始，单表 UPDATE 或 DELETE 语句，支持 Semi Join 半联接查询优化。

从下图可以看到 Semi Join 半连接优化生效了。

```
mysql>
mysql> explain  update t1 set name='aa' where id in (select id from t2 );
+----+-------------+-------+------------+--------+---------------+---------+---------+------------+------+----------+-------------+
| id | select_type | table | partitions | type   | possible_keys | key     | key_len | ref        | rows | filtered | Extra       |
+----+-------------+-------+------------+--------+---------------+---------+---------+------------+------+----------+-------------+
|  1 | UPDATE      | t1    | NULL       | ALL    | PRIMARY       | NULL    | NULL    | NULL       |    4 |   100.00 | NULL        |
|  1 | SIMPLE      | t2    | NULL       | eq_ref | PRIMARY       | PRIMARY | 4       | test.t1.id |    1 |   100.00 | Using index |
+----+-------------+-------+------------+--------+---------------+---------+---------+------------+------+----------+-------------+
2 rows in set, 1 warning (0.00 sec)

mysql> select version();
+-----------+
| version() |
+-----------+
| 8.0.22    |
+-----------+
1 row in set (0.00 sec)
```

2、新增 Anti Join 反连接优化

在 MySQL 8.0.18 版本里，支持对 NOT IN/EXISTS 子查询语句优化，优化器内部将查询自动重写为 Anti Join 反连接查询 SQL 语句。

让我们看一个例子

```
explain select * from t1 where id not in (select id from t2);
```

```
mysql> explain select * from t1 where id not in (select id from t2);
+----+-------------+-------+------------+--------+---------------+---------+---------+------------+------+----------+--------------------------------------+
| id | select_type | table | partitions | type   | possible_keys | key     | key_len | ref        | rows | filtered | Extra                                |
+----+-------------+-------+------------+--------+---------------+---------+---------+------------+------+----------+--------------------------------------+
|  1 | SIMPLE      | t1    | NULL       | ALL    | NULL          | NULL    | NULL    | NULL       |    4 |   100.00 | NULL                                 |
|  1 | SIMPLE      | t2    | NULL       | eq_ref | PRIMARY       | PRIMARY | 4       | test.t1.id |    1 |   100.00 | Using where; Not exists; Using index |
+----+-------------+-------+------------+--------+---------------+---------+---------+------------+------+----------+--------------------------------------+
2 rows in set, 1 warning (0.00 sec)

mysql> show warnings;
| Level | Code | Message |
| Note  | 1003 | /* select#1 */ select `test`.`t1`.`id` AS `id`,`test`.`t1`.`name` AS `name` from `test`.`t1` anti join (`test`.`t2`) on((`test`.`t2`.`id` = `test`.`t1`.`id`)) where true |
1 row in set (0.00 sec)

mysql>
```

优化器内部，是将 not in 子查询重写为下面的语句

```
explain select t1.* from t1 left join t2 on t1.id=t2.id where t2.id is null;
```

通过下面两条 SQL 的执行计划对比，可以看到结果是一样的。

```
mysql> explain select t1.* from t1 left join t2 on t1.id=t2.id where t2.id is null;
+----+-------------+-------+------------+--------+---------------+---------+---------+---------+------+----------+-----------------------------------------------+
| id | select_type | table | partitions | type   | possible_keys | key     | key_len | ref     | rows | filtered | Extra                                         |
+----+-------------+-------+------------+--------+---------------+---------+---------+---------+------+----------+-----------------------------------------------+
|  1 | SIMPLE      | t1    | NULL       | ALL    | NULL          | NULL    | NULL    | NULL    |    4 |   100.00 | NULL                                          |
|  1 | SIMPLE      | t2    | NULL       | eq_ref | PRIMARY       | PRIMARY | 4       | test.t1.id |    1 |   100.00 | Using where; Not exists; Using index          |
+----+-------------+-------+------------+--------+---------------+---------+---------+---------+------+----------+-----------------------------------------------+
2 rows in set, 1 warning (0.00 sec)

mysql> explain select * from t1 where id not in (select id from t2);
+----+-------------+-------+------------+--------+---------------+---------+---------+---------+------+----------+-----------------------------------------------+
| id | select_type | table | partitions | type   | possible_keys | key     | key_len | ref     | rows | filtered | Extra                                         |
+----+-------------+-------+------------+--------+---------------+---------+---------+---------+------+----------+-----------------------------------------------+
|  1 | SIMPLE      | t1    | NULL       | ALL    | NULL          | NULL    | NULL    | NULL    |    4 |   100.00 | NULL                                          |
|  1 | SIMPLE      | t2    | NULL       | eq_ref | PRIMARY       | PRIMARY | 4       | test.t1.id |    1 |   100.00 | Using where; Not exists; Using index          |
+----+-------------+-------+------------+--------+---------------+---------+---------+---------+------+----------+-----------------------------------------------+
2 rows in set, 1 warning (0.00 sec)
```

3、新增 Hash Join 优化

在 MySQL 8.0.18 中，增加了 Hash Join 新功能，它适用于未创建索引的字段，做等值关联查询。在之前的版本里，如果连接的字段没有创建索引，查询速度会是非常慢的（尤其是大表，执行频率很高的话，直接会将数据库打死），优化器会采用 BNL（块嵌套）算法。

Hash Join 算法是把一张小表数据存储到内存中的哈希表里，并逐行去匹配大表中的数据，计算哈希值并把符合条件的数据，从内存中返回客户端。

我们用 explain 命令可以查看到已经使用到 hash join 算法。（在 MySQL 8.0.20 和更高版本中，可以使用 EXPLAIN，省略 FORMAT = TREE）

```
mysql> explain select count(*) from t1 left join t2 on t1.name=t2.name;
+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+----------------------------------------------------+
| id | select_type | table | partitions | type | possible_keys | key  | key_len | ref  | rows | filtered | Extra                                              |
+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+----------------------------------------------------+
|  1 | SIMPLE      | t1    | NULL       | ALL  | NULL          | NULL | NULL    | NULL |    4 |   100.00 | NULL                                               |
|  1 | SIMPLE      | t2    | NULL       | ALL  | NULL          | NULL | NULL    | NULL |    5 |   100.00 | Using where; Using join buffer (hash join)         |
+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+----------------------------------------------------+
2 rows in set, 1 warning (0.00 sec)
```

注：这里的字段 name 未创建索引。

哈希联接可用的内存量受 join_buffer_size 的值限制，生产可适当调大。

总结：以上三个特性，在不改变 SQL 语句的情况下，MySQL 自身就将其优化，减少了开发日常的工作，避免了生产事故的发生。

InnoDB

1、instant add column 亿级大表毫秒级加字段

加字段是痛苦的，需要对表进行重建，尤其是对亿级别的大表，DBA 会反馈开发，表太大，加不了了，那么开发就得重新调整业务逻辑，势必增加了工作量。

虽然 Online DDL 可以避免锁表，但如果在主库上执行耗时 30 分钟，那么再复制到从库上执行，主从复制就出现延迟。使用 instant ADD COLUMN 特性，只需弹

下烟灰的时间，字段就加好了，享受 MongoDB 那样的非结构化存储的灵活方便，无形中减少了开发的工作量。

限制

- 如果指定了 AFTER，字段必须是在最后一列，否则需要重建表；
- 不适用于 ROW_FORMAT = COMPRESSED；
- DROP COLUMN 需要重建表；
- modify 修改字段属性需要重建表。

系统稳定性提升

1、资源组有效解决慢 SQL 引发 CPU 告警

资源组的作用是资源隔离（你可以理解为开通云主机时勾选的硬件配置），将线上的慢 SQL 线程 id 分配给 CPU 一个核，让它慢慢跑，从而不影响 CPU 整体性能。

注：资源组启动需开启 CAP_SYS_NICE 功能

开启步骤如下：

```
# setcap cap_sys_nice+ep /usr/local/mysql/bin/mysqld
# getcap /usr/local/mysql/bin/mysqld
/usr/local/mysql/bin/mysqld = cap_sys_nice+ep
```

2、Query Rewrite 插件重写 DML 语句

MySQL 8.0 Query Rewrite 支持 SELECT INSERT UPDETE DELETE REPLACE 语句重写这个功能要点赞，比如开发上线时，有个 SQL 查询字段索引忘记加了，直接把线上 CPU 打满，此时，你可以将 SQL 重写，让业务先报错，别打死数据库，然后马上通知开发回滚，等加完索引后再上线。

三、使用场景选型测试

2019 年 1 月 24 日 Percona 公司写了一篇博文，是通过单机 TiDB 与 MySQL 的性能压测对比：https://www.percona.com/blog/2019/01/24/a-quick-look-into-tidb-performance-on-a-single-serve/

随着时间的推进，目前 TiDB 发布了 4.0 GA 版本，接下来再次验证一下新版本所带来的性能提升与场景选择建议。

TiDB 的安装同样选择单机版，即一台机器（1 个 PD，1 个 TiDB，1 个 TiKV，其他的无）。

众所周知 MySQL 是 OLTP 产品，那么接下来同样采用 sysbench 做基准测试，基于主键和索引的增删改查操作。

由于机器条件有限，为一台云主机，4 核 8GB 配置，上面分表部署了 MySQL 8.0（innodb_buffer_pool_size 设置 1GB）和 TiDB 4.0，sysbench 分别仅生成 1 张表（1000 万行数据）。

```
#                                                                    sysbench
--test=/usr/share/sysbench/tests/include/oltp_legacy/oltp.lua
--oltp_tables_count=1                          --mysql-table-engine=innodb
--oltp-table-size=10000000
--max-requests=0          --report-interval=1          --max-time=100
--num-threads=16
--mysql-host=127.0.0.1       --mysql-port=4000       --mysql-user=root
--mysql-password=123456
--mysql-db=test  --db-driver=mysql  run
```

压测环境为先性能压测 MySQL，然后重启服务器，再压测 TiDB，对比数据。

MySQL 压测数据，如下图所示：

```
[ 80s ] thds: 16 tps: 1051.73 qps: 20980.66 (r/w/o: 14698.26/4179.94/2102.47) lat (ms,95%): 27.17 err/s: 0.00 reconn/s: 0.00
[ 81s ] thds: 16 tps: 1091.17 qps: 21856.32 (r/w/o: 15301.32/4371.66/2183.33) lat (ms,95%): 24.83 err/s: 0.00 reconn/s: 0.00
[ 82s ] thds: 16 tps: 1049.95 qps: 21010.95 (r/w/o: 14691.27/4219.79/2099.90) lat (ms,95%): 27.66 err/s: 0.00 reconn/s: 0.00
[ 83s ] thds: 16 tps: 1074.97 qps: 21476.38 (r/w/o: 15039.55/4286.89/2149.93) lat (ms,95%): 26.68 err/s: 0.00 reconn/s: 0.00
[ 84s ] thds: 16 tps: 1055.90 qps: 21171.05 (r/w/o: 14820.64/4238.62/2111.80) lat (ms,95%): 27.17 err/s: 0.00 reconn/s: 0.00
[ 85s ] thds: 16 tps: 989.12 qps: 19750.38 (r/w/o: 13825.67/3946.48/1978.24) lat (ms,95%): 30.26 err/s: 0.00 reconn/s: 0.00
[ 86s ] thds: 16 tps: 988.04 qps: 19764.72 (r/w/o: 13825.50/3963.14/1976.07) lat (ms,95%): 30.26 err/s: 0.00 reconn/s: 0.00
[ 87s ] thds: 16 tps: 1031.81 qps: 20647.21 (r/w/o: 14460.34/4123.24/2063.62) lat (ms,95%): 27.66 err/s: 0.00 reconn/s: 0.00
[ 88s ] thds: 16 tps: 1040.09 qps: 20784.81 (r/w/o: 14556.26/4148.36/2080.18) lat (ms,95%): 26.68 err/s: 0.00 reconn/s: 0.00
[ 89s ] thds: 16 tps: 1001.99 qps: 20055.75 (r/w/o: 14033.82/4017.95/2003.97) lat (ms,95%): 30.26 err/s: 0.00 reconn/s: 0.00
[ 90s ] thds: 16 tps: 1069.98 qps: 21389.51 (r/w/o: 14969.66/4279.90/2139.95) lat (ms,95%): 24.83 err/s: 0.00 reconn/s: 0.00
[ 91s ] thds: 16 tps: 1006.16 qps: 20118.30 (r/w/o: 14092.31/4013.66/2012.33) lat (ms,95%): 31.37 err/s: 0.00 reconn/s: 0.00
[ 92s ] thds: 16 tps: 1064.00 qps: 21334.99 (r/w/o: 14914.00/4293.00/2128.00) lat (ms,95%): 25.74 err/s: 0.00 reconn/s: 0.00
[ 93s ] thds: 16 tps: 995.87 qps: 19864.46 (r/w/o: 13924.22/3948.49/1991.75) lat (ms,95%): 28.16 err/s: 0.00 reconn/s: 0.00
[ 94s ] thds: 16 tps: 1013.83 qps: 20232.78 (r/w/o: 14150.77/4054.34/2027.67) lat (ms,95%): 25.74 err/s: 0.00 reconn/s: 0.00
[ 95s ] thds: 16 tps: 1084.08 qps: 21680.61 (r/w/o: 15180.14/4332.32/2168.16) lat (ms,95%): 26.20 err/s: 0.00 reconn/s: 0.00
[ 96s ] thds: 16 tps: 911.84 qps: 18273.72 (r/w/o: 12794.70/3656.34/1822.67) lat (ms,95%): 31.94 err/s: 0.00 reconn/s: 0.00
[ 97s ] thds: 16 tps: 1024.39 qps: 20507.80 (r/w/o: 14353.46/4104.56/2049.78) lat (ms,95%): 28.16 err/s: 0.00 reconn/s: 0.00
[ 98s ] thds: 16 tps: 1009.06 qps: 20212.26 (r/w/o: 14137.88/4056.25/2018.13) lat (ms,95%): 28.67 err/s: 0.00 reconn/s: 0.00
[ 99s ] thds: 16 tps: 1097.01 qps: 21883.28 (r/w/o: 15333.20/4356.06/2194.03) lat (ms,95%): 26.68 err/s: 0.00 reconn/s: 0.00
[ 100s ] thds: 16 tps: 991.01 qps: 19771.14 (r/w/o: 13816.10/3976.03/1979.01) lat (ms,95%): 30.26 err/s: 0.00 reconn/s: 0.00
SQL statistics:
    queries performed:
        read:                            1426152
        write:                           407472
        other:                           203736
        total:                           2037360
    transactions:                        101868 (1018.53 per sec.)
    queries:                             2037360 (20370.63 per sec.)
    ignored errors:                      0      (0.00 per sec.)
    reconnects:                          0      (0.00 per sec.)
```

TiDB 压测数据，如下图所示：

```
[ 80s ] thds: 16 tps: 231.58 qps: 4549.69 (r/w/o: 3171.20/916.33/462.16) lat (ms,95%): 95.81 err/s: 0.00 reconn/s: 0.00
[ 81s ] thds: 16 tps: 241.48 qps: 4889.66 (r/w/o: 3448.81/948.87/491.97) lat (ms,95%): 84.47 err/s: 0.00 reconn/s: 0.00
[ 82s ] thds: 16 tps: 229.01 qps: 4562.29 (r/w/o: 3185.20/917.06/460.03) lat (ms,95%): 95.81 err/s: 0.00 reconn/s: 0.00
[ 83s ] thds: 16 tps: 245.65 qps: 4882.99 (r/w/o: 3421.09/965.61/496.29) lat (ms,95%): 84.47 err/s: 0.00 reconn/s: 0.00
[ 84s ] thds: 16 tps: 220.25 qps: 4435.94 (r/w/o: 3097.45/896.00/442.49) lat (ms,95%): 90.78 err/s: 0.00 reconn/s: 0.00
[ 85s ] thds: 16 tps: 222.06 qps: 4447.23 (r/w/o: 3116.86/882.24/448.12) lat (ms,95%): 108.68 err/s: 0.00 reconn/s: 0.00
[ 86s ] thds: 16 tps: 242.13 qps: 4865.52 (r/w/o: 3404.77/974.50/486.25) lat (ms,95%): 82.96 err/s: 0.00 reconn/s: 0.00
[ 87s ] thds: 16 tps: 231.17 qps: 4555.34 (r/w/o: 3180.33/909.67/465.34) lat (ms,95%): 94.10 err/s: 0.00 reconn/s: 0.00
[ 88s ] thds: 16 tps: 256.71 qps: 5177.35 (r/w/o: 3619.03/1037.88/520.44) lat (ms,95%): 82.96 err/s: 0.00 reconn/s: 0.00
[ 89s ] thds: 16 tps: 246.01 qps: 4879.18 (r/w/o: 3421.13/960.04/498.02) lat (ms,95%): 80.03 err/s: 0.00 reconn/s: 0.00
[ 90s ] thds: 16 tps: 231.95 qps: 4673.06 (r/w/o: 3277.34/927.81/467.91) lat (ms,95%): 94.10 err/s: 0.00 reconn/s: 0.00
[ 91s ] thds: 16 tps: 266.04 qps: 5343.83 (r/w/o: 3734.58/1072.17/537.08) lat (ms,95%): 77.19 err/s: 0.00 reconn/s: 0.00
[ 92s ] thds: 16 tps: 264.54 qps: 5301.86 (r/w/o: 3727.57/1038.21/536.08) lat (ms,95%): 71.83 err/s: 0.00 reconn/s: 0.00
[ 93s ] thds: 16 tps: 261.39 qps: 5162.75 (r/w/o: 3599.40/1037.56/525.79) lat (ms,95%): 77.19 err/s: 0.00 reconn/s: 0.00
[ 94s ] thds: 16 tps: 245.07 qps: 4910.38 (r/w/o: 3444.97/971.27/494.14) lat (ms,95%): 90.78 err/s: 0.00 reconn/s: 0.00
[ 95s ] thds: 16 tps: 244.96 qps: 4963.20 (r/w/o: 3472.44/997.84/492.92) lat (ms,95%): 87.56 err/s: 0.00 reconn/s: 0.00
[ 96s ] thds: 16 tps: 253.06 qps: 5042.22 (r/w/o: 3526.86/1009.25/506.12) lat (ms,95%): 80.03 err/s: 0.00 reconn/s: 0.00
[ 97s ] thds: 16 tps: 234.00 qps: 4627.99 (r/w/o: 3237.99/921.00/469.00) lat (ms,95%): 99.33 err/s: 0.00 reconn/s: 0.00
[ 98s ] thds: 16 tps: 233.97 qps: 4737.48 (r/w/o: 3320.64/941.90/474.95) lat (ms,95%): 82.96 err/s: 0.00 reconn/s: 0.00
[ 99s ] thds: 16 tps: 236.00 qps: 4701.03 (r/w/o: 3290.02/936.01/475.00) lat (ms,95%): 87.56 err/s: 0.00 reconn/s: 0.00
[ 100s ] thds: 16 tps: 223.02 qps: 4453.48 (r/w/o: 3122.34/882.10/449.05) lat (ms,95%): 99.33 err/s: 0.00 reconn/s: 0.00
SQL statistics:
    queries performed:
        read:                            335510
        write:                           95446
        other:                           48344
        total:                           479300
    transactions:                        23965  (239.51 per sec.)
    queries:                             479300 (4790.20 per sec.)
    ignored errors:                      0      (0.00 per sec.)
    reconnects:                          0      (0.00 per sec.)
```

结论：对比上述数据，MySQL 8.0 的性能基准测试数据是 TiDB 4.0 的 4 倍+，吻合 Percona 官方压测结果，适用于并发较少，数据量较少（或者你的硬盘足够大，以支撑业务数据增长量）的 OLTP 使用场景，MySQL 仍旧是单机版之王，地位难以撼动。

TiDB 对于快速查询（即通过主键和索引点查）和写入的效率较低。较小的数据规模、并发并不是 TiDB 所擅长的场景，架构上就会有开销。TiDB 更偏向于 OLAP 使用场景，在复杂的 SQL 查询时有优势。

最后附上官方压测结论：
Short version: TiDB supports parallel query execution for selects and can utilize many more CPU cores - MySQL is limited to a single CPU core for a single select query. For the higher-end hardware - ec2 instances in my case - TiDB can be 3-4 times faster for complex select queries (OLAP workload) which do not use, or benefit from, indexes. At the same time point selects and writes, especially inserts, can be 5x-10x slower. Again, please note that this test was on a single server, with a single TiKV process.

PostgreSQL 获得“2020 年度数据库”称号

一、PostgreSQL 以最高增长速度获得 DB-Engines 2020 年度数据库称号

二、PostgreSQL 13 正式版的提升与优化

PostgreSQL 全球开发组在 2020 年 9 月 24 日宣布 PostgreSQL 13 正式发布，也是目前的最新版本。

PostgreSQL 13 在索引和查找方面进行了重大改进，有利于大型数据库系统，同时包括索引的空间节省和性能提高，使用聚合或分区的查询时的更快响应，使用增强的统计信息时更优化的查询计划，以及很多其他改进。

PostgreSQL 13 除了具有强烈要求的功能（如并行清理和增量排序）外，还为不同大小的负载提供了更好的数据管理体验。此版本针对日常管理进行了优化，为应用程序开发人员提供了更多便利，并增强了安全性。

1、持续的性能提升：

有效地处理标准数据库索引 B-tree 中的重复数据。这降低了 B-tree 索引所需的总体使用空间，同时提高了整体查询性能。

引入了增量排序，其中查询中来自较早步骤的已排序数据可以加快后续步骤的排序。

使用扩展的统计信息（通过 CREATE STATISTICS 访问）来创建增强带有 OR 子句和列表中的 IN/ANY 查找的查询计划。

更多类型的聚合和分组可以利用 PostgreSQL 的高效哈希聚合功能，因为具有大聚合的查询不必完全放在内存中。

带有分区表的查询性能得到了提高，因为现在有更多情况可以修剪分区并且可以直接连接分区。

2、管理优化

引入索引的并行清理来继续改进清理系统。数据插入现在还可以触发自动清理过程。

复制槽(Replication slots)用于防止预写日志（WAL）在备库收到之前被删除，可以指定要保留的 WAL 文件的最大数量，有助于避免磁盘空间不足的错误。

增加了更多管理员可以监视数据库活动的方式，包括从 EXPLAIN 查看 WAL 使用情况的统计信息，基于流的备份进度，以及 ANALYZE 命令的进度。

可以使用新的 pg_verifybackup 命令来检查 pg_basebackup 命令输出的完整性。

3、便利的应用程序开发

在 SQL/JSON 路径支持中添加了 datetime()函数，该函数将有效的时间格式（例如 ISO 8601 字符串）转换为 PostgreSQL 本地类型。

UUID v4 生成函数 gen_random_uuid()现在可以直接使用而无需安装任何扩展。分区表完全支持逻辑复制和 BEFORE 行级触发器。

FETCH FIRST 语法现已扩展为可包含 WITH TIES 子句。

4、安全增强

添加了“可信扩展”的概念，该概念允许数据库用户使用安装超级用户标记为“受信任”的扩展。某些内置扩展默认情况下标记为受信任，包括 pgcrypto、tablefunc、hstore 等。

允许客户端在使用 SCRAM 身份验证时要求通道绑定，postgres_fdw 现在可以使用基于证书的身份验证。

三、PostgreSQL 常用插件动态

1、分布式插件 citus 发布 9.5.1

citus 是 PostgreSQL 的一款比较流行的 sharding 插件，agpl 开源协议，目前为微软所拥有，国内苏宁有较大量使用案例。

https://github.com/citusdata/citus

2、地理信息插件 postgis 3.1 正式版本发布

PostGIS 是专业的时空数据库插件，在测绘、航天、气象、地震、国土资源、地图等时空专业领域应用广泛。同时在互联网行业也得到了对 GIS 有性能、功能深度要求的客户青睐，比如共享出行、外卖等客户。

官网地址：http://postgis.net/

3、时序插件 timescaledb 发布 2.0.0

timescale 是 PostgreSQL 的一款时序数据库插件，在 IoT 行业中有非常好的应用。github star 数目前有 1 万多，是一个非常火爆的插件。

2.0.0 的主要功能增强包括：

- 添加对分布式表的支持
- 添加对用户定义的 actions
- 添加持续聚合 API
- 重新设计信息视图
- 将企业功能移动到社区版

官网地址：https://github.com/timescale/timescaledb

4、Oracle 兼容插件 orafce 发布 3.13.4

orafce 是 PostgreSQL 的一款增强 Oracle 兼容性的插件。

官网地址：https://api.pgxn.org/src/orafce/

四、PostgreSQL 知名衍生产品动态

1、gpdb 发布 6.13.0 版

gpdb 是兼容 PostgreSQL 的开源 mpp 数据库，适合 OLAP 场景。

https://github.com/greenplum-db/gpdb

OceanBase 最新 2.2.7 版本有重要更新

2020 年，OceanBase 数据库一共迭代了 3 个版本，最新发布的 2.2.7 版本在高可用性、兼容性、安全性等方面进行了重要的更新：

一、更灵活的高可用性

OceanBase 数据库在高可用特性方面，除了提供原有的基于多副本的高可用形态之外，又提供了基于事务日志复制技术的高可用特性—主备库，创新性的将日志和复制技术引入到了原生分布式数据库领域，为客户提供更加灵活的高可用和容灾能力。主集群通过向备用集群发送事务日志的方式来实现数据同步，从而确保生产集群能够在遇到数据损坏、灾难等情况下仍然可以快速恢复业务。当 OceanBase 数据库生产集群出现计划内或者计划外的不可用情况时，主备库可以通过将某一个备用集群的角色切换为主集群，从而保证系统的持续运行，最大限度地降低服务的停机时间。另外，OceanBase 数据库在物理备份恢复方面也提供了更多的功能。

二、支持主备库

OceanBase 的主备库高可用架构是 OceanBase 高可用能力的重要补充。当主集群出现计划内或计划外（多数派副本故障）的不可用情况时，备集群可以接管服务，并且提供无损切换（RPO = 0）和有损切换（RPO > 0）两种容灾能力，最大限度降低服务停机时间。

主备库可以解决 OceanBase 集群不可用情况下的容灾问题，提供数据保护和灾难恢复能力，典型场景包括同城双中心、两地双中心、两地三中心等。当双机房部署在一个地域时，主备库可以满足机房级容灾的需求；主集群部署在应用主机房，多副本架构，满足机器级容灾能力；备集群部署在应用备机房，可以选择单副本，降低部署成本，也可以选择多副本；当应用主机房故障时，备集群可以 failover 成新主集群，接管服务，从而提供机房级容灾能力。同理，对于两地两中心和两地三中心部署场景，当主集群的多数派所在的地域故障时，主备库可以提供地域级容灾能力。

主备库功能主要包括：

- 支持配置一个主集群和一个或多个备集群。
- 支持最大保护模式、最大性能模式，两种保护模式可相互切换。
- 提供 Switchover 和 Failover 两种集群角色的切换能力。
- 并且 OCP 也已经与主备库进行集成。

三、支持物理备份与恢复

备份恢复作为数据库高可用性的基础功能，也是数据保护的最后一道防线。OceanBase 数据库已经实现了基于物理块拷贝的物理备份，也提供了更多的功能和性能优化，主要包括：

- 支持全量备份、增量备份、日志归档三种类型的备份；
- 支持 NFS 和 OSS 两种备份目的地；
- 支持指定备份数据的保存时间策略。

四、更高的 Oracle 兼容性

1、SQL 语法

- 新增支持正则表达式
  OceanBase 数据库支持 5 个正择表达式，REGEXP_LIKE、REGEXP_REPLACE、REGEXP_SUBSTR、REGEXP_INSTR、REGEXP_COUNT；
- DML 语句支持 returning into 子句
  支持在 DML 语句中通过 returning into 子句将涉及到的数据返回到变量或者复杂数据类型当中；
- 提供 select .... for update 语法
  支持在运行查询的时候可以根据需要锁定查询到的数据；
- 支持 oracle 模式下外键约束
  支持 enable/disable、validate/novalidate、rely/norely 选项；
- 新增支持更多窗口/聚合函数
  支持 CORR（相关系数）、COVAR_POP（总体协方差）等概率相关的函数、REGR_INTERCEPT、REGR_AVGX 等线性回归函数；
- 支持逻辑 Rowid
  在 Oracle 模式中为有主键表提供逻辑 rowed 伪列；
- 支持从 OceanBase 数据库 Oracle 租户到 Oracle 数据库的数据库链接。

2、支持 PL/SQL

- 支持创建函数、存储过程、包，自定义类型等大部分的高级特性；
- 并且支持 dbms_lob、dbms_output、dbms_metadata 等常见系统包，以及和安全相关的系统包。

3、支持更复杂的查询

- 支持层次查询的start with子句和where子句中出现子查询，支持查询join表；
- 实现层次查询的hash执行方式，提升执行性能。

4、提供更强的并行处理能力

- 提供提前解除行锁的能力来降低锁冲突，提升系统性能。事务解锁的时机由日志持久化成功，提前到拷贝batch buffer完成，使得操作同一行的事务能够并发执行。

五、更高的安全性

1、提供安全审计功能

为了防止用户登录信息泄露或者访问权限被滥用，加强企业对数据安全、合规等方面的要求，数据库需要记录有权限用户的操作。审计是跟踪用户行为最主要的工具。

OceanBase 数据库在 Oracle 模式下提供了审计功能，语法与 Oracle 基本保持一致。审计的作用范围是租户，也就是说，租户 A 下发起的审计只对租户 A 下的用户生效。 审计配置只有审计用户 ORAAUDITOR 可以进行，符合三权分立的原则。

安全审计支持以下功能：

- 以租户为单位，租户 A下发起的审计只对租户 A 下用户操作有效，其它租户无效；
- 审计配置只有审计员 ORAAUDITOR 可以进行，以此满足等保三级中三权分立要求；
- 审计语法与 Oracle 标准审计基本保持一致，相关表视图与 Oracle 保持一致；
- 标准审计中支持OStatement/Object 审计类型，待支持 Privilege 审计，不支持 Network 审计和 FGA 审计；
- 审计结果支持写文件/写内部表，不支持写 xml/syslog；
- 如果写文件，文件较大时会自动切割，文件路径不支持用户配置；
- 如果写表，会占用系统表空间，系统表表空间满则用户语句阻塞住。

2、支持数据存储加密

数据存储加密是指对数据和 CLOG 等磁盘中的数据进行透明加密（以下简称 TDE），数据在写入存储设备之前会实时自动进行加密，并在从存储设备中读取时进行解密，这个过程对用户是透明的，黑客和恶意用户无法从数据文件，数据库备份或磁盘中读取到敏感数据.对于传输通信加密，则可以规避明文信息传播过程中潜在的窃听、篡改、冒充风险。OceanBase 为了方便客户使用 TDE 特性，在实现上也尽量兼容 Oracle 的实现方式，数据的加密单位为 tablespace。不过需要说明的是，OceanBase 并不是一个多文件的数据库系统，表空间是为了兼容 TDE 而产生的一个逻辑概念，可以简单的理解为表空间是一组表的集合。

3、支持数据传输加密

OceanBase 在数据库保护方面提供了数据全连路加密和存储透明加密功能，保护用户的数据安全。OceanBase采用在原有的TCP通信上扩展支持SSL/TLS协议解决通信加密的问题、确保客户端于数据库之前的数据通信、数据库节点之间的通信都能够被加密。

为了能给用户提供更大的便利性，OceanBase为每个user提供不同SSL认证机制，例如：SSL单向认证、X509双向认证、指定加密算法认证、指定发行方认证等。

其他新特性：

- 支持range，list，hash，key类型的分区，支持模版化、非模版化二级分区、支持对分区的添加、删除等多种维护操作；
- 针对单机、分布式事务的性能提升；
- 数据文件动态调整大小。

NoSQL 家族

Redis 2020 年度功能特性更新亮点

从 2019 年 12 月 19 日 6.0 RC 到 2020 年 12 月 14 日 6.2 RC，长达 1 年的时间中，Redis 提供了许多喜人的功能和特性，虽然在性能上没有质的提升，但是在功能性、易用性和可维护性上面还是有许多值得期待和关注的亮点：

一、新增 ACL 权限控制

从 Redis 6 开始支持 ACL，该功能通过限制对命令和 key 的访问来提高安全性。ACL 的工作方式是在连接之后，要求客户端进行身份验证（用户名和有效密码）；如果身份验证阶段成功，则连接与指定用户关联，并且该用户具有限制。同时，该功能与旧版本是向后兼容的，完全不用担心兼容性问题。

ACL 的用法：

```
ACL <subcommand> arg arg ... arg. Subcommands are:
1) LOAD                            -- Reload users from the ACL file.
2) SAVE                            -- Save the current config to the ACL
file.
3) LIST                             -- Show user details in config file
format.
4) USERS                           -- List all the registered usernames.
5) SETUSER <username> [attribs ...]    -- Create or modify a user.
6) GETUSER <username>              -- Get the user details.
7) DELUSER <username> [...]          -- Delete a list of users.
8) CAT                              -- List available categories.
9) CAT <category>                   -- List commands inside category.
10) GENPASS [<bits>]                -- Generate a secure user password.
11) WHOAMI                         -- Return the current connection
username.
12) LOG [<count> | RESET]            -- Show the ACL log entries.
```

同时，Redis6 中 auth 命令在 Redis 6 中进行了扩展，因此现在可以在两个参数的形式中使用它：

```
#before Redis 6
AUTH <password>
#Redis 6
AUTH <username> <password>
```

二、新增 SSL 支持

Redis 6 开始支持 SSL，连接更加的安全。

三、多线程 IO

众所周知，Redis 以高性能著称，主要得益于单进程单线程的工作方式，多路 I/O 复用技术让单个线程高效的处理多个连接请求，使得 Redis 能够获得优异的性能。多线程 IO 的引入，主要用来处理网络数据的读写和协议解析的性能，但是命令的执行仍然是单线程，这样设计主要是不想因为多线程而使逻辑变得复杂，需要去处理 key、lua、事务，LPUSH/LPOP 等等的并发问题。

四、无盘复制

rdb 作为 Redis 持久化方式之一，默认生成的 rdb 文件会存储在本地的磁盘上，开启无盘复制后，如果不再有用，将会被立即被删除。

```
repl-diskless-sync no
repl-diskless-sync-delay 5
repl-diskless-load disabled
```

同时，对 RDB 的加载也做了优化，RDB 文件的加载速度比之前变得更快了。根据文件的实际组成（较大或较小的值），大概可以获得 20-30%的性能提升。

五、Redis-benchmark 支持集群

六、Redis 引入集群代理

从 Redis 6 开始引入一个集群代理，可以为客户端抽象 Redis 集群，使其像正在于单个实例进行交互一样，可以简化客户端的操作。同时在简单且客户端仅使用简单命令和功能时执行多路复用。值得注意的是，这个集群代理跟 Redis 不在同一个 repo 中。其 Repo 如下：

https://github.com/artix75/redis-cluster-proxy

七、新增命令或命令优化

- 添加 SMISMEMBER 命令：原子批量的 sismember；
- 添加 ZMSCORE 命令：原子批量 zscore；
- 添加 LMOVE 和 BLMOVE 命令：用来代替 RPOPLPUSH（6.2 后被废弃）；
- 添加 RESET 命令可以重置客户端连接状态；
- 添加 COPY 命令：复制 key；
- 添加 ZDIFF 和 ZDIFFSTORE 命令；
- 添加 ZINTER 和 ZUNION 命令；
- 添加 GEOSEARCH/GEOSEARCHSTORE 命令：GEO 可以基于方形搜索（之前是基于半径）；

- 将 GET 参数添加到 SET 命令，以获得更强大的 GETSET(GETSET 在 6.2 后被废弃)；
- 向 XPENDING、X[REV]RANGE 添加专有范围查询；
- 为 ZADD 添加 GT 和 LT 选项以进行条件分数更新；
- client info 和 list 接口支持指定 ids；
- 在 XPENDING 命令中添加 IDLE 参数；
- 将本地地址添加到 CLIENT LIST 和 CLIENT KILL 过滤器；
- 将 NOMKSTREAM 选项添加到 XADD 命令；
- 将命令 command 添加到 Sentinel；
- 添加 SENTINEL MYID 子命令等等。

八、Info 字段改进

Info 字段的改进，更加的方便采集和监控集群的运行状态指标。比如：

- info 中添加无盘复制进度信息；
- 将主线程 cpu 时间添加到 INFO 字段；
- 将 total_forks 添加到 INFO STATS 字段；
- 将 maxclients 和 cluster_connections 添加到 INFO CLIENTS 字段；
- 在客户端列表中添加跟踪 bcast 标志和客户端重定向标志；
- 修复 INFO client_recent_max_input_buffer 字段中的相关问题；
- 添加 total_reads_processed 和 total_writes_processed 方便计算读写比。

九、功能或性能优化改进

- EXISTS 命令不会更改 LRU；
- object 命令不触发过期（LOOKUP_NOTOUCH|LOOKUP_NONOTIFY）；
- 解决 rehash 在大量 key-value 下内存增加导致的数据剔除；
- CONFIG REWRITE 变为原子且更安全的，但需要对配置文件的文件夹具有写访问权；
- 使用新的增量逐出机制，可减少逐出峰值的延迟：详见 maxmemory-eviction-tenacity 配置；
- 使用命令行参数启动 Redis 时，不重置 save 所保存的配置；
- 在加载期间更新 INFO 的内存指标；
- 改善 SELECT 和 MOVE 的数据库 ID 范围检查；
- 如果 Redis 超出内存限制，则 GEORADIUS[BYMEMBER]可以返回 -OOM 错误。

十、客户端优化改进

对 Redis 客户端缓存进行了重新设计放弃了缓存槽（caching slot）方法而只使用 key 的名称。并且引入了广播模式（broadcasting mode），在使用广播模式时，服务器端不再尝试记住每个客户端请求的 key。 取而代之的是，客户端订阅 key 的前缀：每次修改匹配前缀的 key 时，这些订阅的客户端都会收到通知。这意味着会产生更多的消息（仅适用于匹配的前缀），但服务器端无需进行任何

内存操作。

更多特性，请参阅：

- 6.0 RELEASENOTES：
  https://raw.githubusercontent.com/redis/redis/6.0/00-RELEASENOTES
- 6.2 RELEASENOTES：
  https://raw.githubusercontent.com/redis/redis/6.2/00-RELEASENOTES

RocksDB 6.x 大版本迭代更新回顾

从 2019 年 11 月 25 日 6.6.0 到 2020 年 12 月 22 日 6.15.2，长达 1 年的时间中，主要在 6.x 这个大版本上迭代，增加一些新特性及性能方面的优化。

一、新特性：

1. sst_dump 的时候新加了 readahead_size 参数，可以指定 scan 时候读取数据的大小。

2. 增加了 Options.file_checksum_gen_factory，用户可以指定 checksum 的函数，在 sst 文件写完之后会把 checksum 和对应的函数名存储在 ExternalSstFileInfo。

3. 在 MultiGet 的时候增加了 value_size_soft_limit,用户可以指定读取 Value 的累积值，当超过 Limit 之后忽略后面值的读取。

4. 在调用 IngestExternalFiles 的时候增加了 checksum 的校验，用户可以通过 verify_file_checksum 这个参数来指定，为了向后兼容可以不启用 checksum 的校验。

5. 对 BlockBasedTableBuilder 增加了 Pipeline 和并行压缩优化的支持，把 Block Build/Block Compression/Block Append 进行 Pipeline，并通过多线程来加速整个处理过程，目前还在验证阶段。

6. 提供了 memkink 的 Block Cache 分配机制，可提供通过 DRAM 所能达到的超大高速缓存(大小可达几 TB)。

7. sst_dump 中增加了对不同 compression level 压缩文件大小和耗时的统计及显示。

8. 通过 DB::GetMapProperty 拿到一些表级别的统计信息。

9. 支持通过 timestamp 的 CompactRange 及 GetApproximateSizes。

10. 通过配置化的方法来更好的管理 serialize 及 compare，以方便将来更好的

扩展，同时支持一些自定义扩展，如 TableFactory 等。

11. 增加了当 IO 出错时，自动恢复功能。当 Flush 或者 WAL 写入出错时，DB 进入只读模式并尝试自动恢复，当在处理压缩出时，DB 还是可以正常读写，并进行自动恢复。

12. 当使用 BlobDB 时，可以对 Base DB 开启定期的 compact 操作。

13. 通过在 BlobDBOptions 中将 enable_garbage_collection 设置为 true 开启对 BlobDB 非 TTL Blob 数据的垃圾回收。在 Compaction 期间遇到的最旧的 N 个文件中的所有有效 Blob（其中 N 是非 TTL Blob 文件的数量乘以 BlobDBOptions :: garbage_collection_cutoff 的值）将重定位到新的 Blob 文件，对一些不再需要的 Blob 文件，做删除操作。

14.MultiGet（）可以使用 IO Uring 并行化从同一 SST 文件读取的内容。默认不开启此功能，可以使用环境变量 ROCKSDB_USE_IO_URING 启用。

二、性能优化：

1. 在 Level iterator 中更多的使用前缀查找及前缀布隆过滤器来提升性能。

2. 通过 DB Option 来指定预读取及其大小，默认值为 512KB，可以通过 options.log_readahead_size 来指定。

3. 读取及解压缩 block 的时候减少多于的内存拷贝，同时在 Direct IO 模式下，减少读取 sst table 和 blob db 时的内存拷贝。

4. 通过内部 cache 机制来解决启动时大量计算 block 大小导致重启比较慢的问题。

5. 通过将块缓存查找共享到适用的过滤器块，以提升带分区过滤器的 MultiGet 的性能。

6. 减少基于块的表中的随机访问期间的键比较。

7. 重用全局统计线程，以减少多实例下线程个数。

8.通过设置 KCompactionStyleLevel 来减少写放大。

更多特性，请参阅：
https://github.com/facebook/rocksdb/blob/master/HISTORY.md

Cassandra 4.0 版本重大更新预览

Cassandra 是连续 8 年 DB-Engines 排名第一的宽表数据库，支持类 SQL 语法 CQL，开发体验接近 MySQL。采用分布式、无中心架构，可扩展 PB 级存储及千万 OPS 读写能力。在互联网、社交、金融、智能制造、AIoT 等行业应用广泛。

2017 年 6 月 23 日发布了 3.11 版本，最新版本是 2020 年 11 月 4 日发布的 3.11.9。预计 2021 年 Q1 将发布打磨了三年多的 4.0 版本。其中零拷贝串流（Zero-copy SSTable streaming）、Netty 节点间通信、虚拟表（Virtual Tables ）、增量式修复、审计日志等功能都十分值得期待。

阿里云于 2020 年商业化发布了云 Cassandra 数据库，目前也在积极跟进 4.0 版本的测试发布工作。

功能预览：

- 零拷贝串流：在串流时无需将数据读到内存后再写入到网络，可直接通过网络收发数据。从而显著提升性能（3-5 倍），大大减少内存和 CPU 占用。可应用在多种场景，如缩短故障节点的恢复时间，降低多个节点同时处于不可用状态的概率；加速节点扩容时数据迁移速率，缩短扩节点时间等；
- 增量式修复：在 2.1 版本中以实验性功能推出，4.0 版本发布用于生产。增量式修复将数据分为“已修复”和“未修复”两个部分，每次修复时无需修复已修复过的数据，仅需修复“未修复”部分。从而减少修复时间，仅需几分钟即可完成；
- Netty 节点间通信：在 4.0 版本中，节点间通信改成了 Netty。I/O 是非阻塞的，不再按节点分配线程。Netflix 通过测试 192 个节点的 Cassandra 集群，对比 4.0 和 3.0 版本，发现 4.0 版本的延迟平均值减少 40%，99 分位的延迟减少了 60%，吞吐量提升约 2 倍；
- 虚拟表（Virtual Tables ）：目前是只读的，基于 Cassandra 内部 API 实现。每一个虚拟表都是每个节点所特有的，也就是说虚拟表是 local 的。可以把虚拟表当作 Cassandra 的一个接口。有了虚拟表，我们可以不用 JMX，通过 CQL 来进行查询虚拟表，从而获取 Cassandra 的系统状态和当前配置。使数据库系统更加安全可控；
- 审计日志：将数据库所有操作记录到一个本地文件，包括 authentication，所有的 CQL 请求，不论成功与否都会被记录下来。用途广泛，如利用记录来 debug 线上问题，也可以辅助测试。4.0 版本中同步推出的 full query logger 就是用审核日志来实现的，这些由 full query logger 生成的记录可用来做回放测试。另外，也可以帮助企业做合规管理，可作为企业的审核依据。

NewSQL 家族

TiDB 2020 年度重大更新及技术要点分析

一、TiDB 4.0 版本关键特性

2020 年，TiDB 总共更新了 35 个版本，在稳定性、性能、安全、高可用、备份与恢复、数据迁移、部署运维和功能方面进行大量的改进，其中 4.0 版本已有大量用户在使用。 TiDB 4.0 版本关键特性如下：

1、性能

与 3.0 相比，TPC-C TpmC 提升 50%，TPC-H 部份查询性能提升 50%，主要通过 IndexMerge、统一线程池、存储层缓存查询结果、Follower read、列存储引擎（TiFlash）等特性提升性能。

2、稳定性

优化器新增 SPM 和 16 种 SQL Hint 确保执行查询语句的稳定性；将查询中间结果写入磁盘，避免系统 OOM；新增 AutoRandom Key 解决大批量数据写入产生的热点问题。

3、备份与恢复

新增分布式备份与恢复工具，备份性能的高达 TiKV 节点 * 150+ MB/s，且随 TiKV 节点数线性扩展，备份与恢复支持全量、Database、Table 级别备份与恢复，可将备份的数据存储到 NFS、AWS S3 等外部存储系统。

4、安全

支持 TDE；所有组件之间的通信息支持 TLS；支持对系统输出的日志脱敏，防止信息泄露。

5、支持悲观事务、事务的大小限制从 100MB 放开到 10GB。

6、部署与运维

TiUP 提供基于命令行的部署、运维工具，秒级部署一套 TiDB、一条命令完成版本升级、扩容、缩容 ，极大的提升部署、运维 TiDB 的效率；支持可视化的诊断、排查 TiDB 的性能的问题，功能包括：Key Visualizer、Statement、Slow Query、Diagnostic Report、Log Search & Download 等。

二、TiDB 5.0-rc 版本最值得关注的特性

TiDB 的发布策略有一个不成文的小规律，一个大版本专注于功能，下一个大版本专注于提升稳定性和性能，在 2020 年中发布的 TiDB 4.0 版本是一个功能偏多的版本，包括第一次引入了 TiFlash，TiUP 以及 Dashboard 等组件和功能，于是在 2021 年初发布的 5.0 版本的重点在于提升整体的性能和稳定性。

随着 TiDB 在金融级客户中落地场景越来越多，也越来越深入，对于性能和稳定性的要求就很明确：满足核心场景的需求。可以在下面的特性中看到 5.0 相比 4.0 在性能和稳定性（尤其引入了长时间高压力测试，对于性能抖动的监控）上有很大的提升。第二个亮点是 TiFlash （TiDB 的 Real-time OLAP 组件）引入了第一个版本的 MPP 实现，这极大扩展了 TiDB 对于实时分析场景的适应度，对于大表 JOIN 的场景，将 OOM 错误降到最低。

另外在 5.0 版本中，TiDB 的事务模型做了一个比较大的改进：在多数据中心异地部署方案中引入本地时钟取代过去集中的时钟，这在有明显地域访问特点的业务场景中，将极大的改善事务的延迟，对于异地多活的场景是一个非常重要的改进。 TiDB 5.0-rc 版本最值得关注的特性如下：

1、性能

开启聚族索引功能，性能与 4.0 相比 TPC-C TpmC 提升了 39%；开启异步提交事务(Async Commit)功能，Sysbench oltp-insert latency 降低了 37.3%。

2、稳定性

通过提升优化器的稳定性及限制系统任务对 I/O、网络、CPU、内存等资源的占用，降低系统的抖动，长期测试 72 小时，衡量 Sysbench TPS 抖动标准差的值从 11.09% 降低到 3.3.6%。

3、可用性

引入 Raft Joint Consensus 算法，确保 Region 成员变更时，确保在有节点故障时系统任旧可用。

4、备份与恢复

通过备份文件到 AWS S3、Google Cloud GCS 或者从 AWS S3、Google Cloud GCS 恢复到 TiDB，确保企业数据的可靠性。

5、导入与导出数据

提升从 AWS S3 或者 TiDB/MySQL 导入导出数据的性能，帮忙企业在云上快速构建应用，例如：导入 1TiB TPC-C 数据性能提升了 40%，由 254 GiB/h 提升到 366 GiB/h；DM 2.0 提供 dm-master、dm-worker 高可用的能力，确保部分节点故障后服务可用。

6、部署、运维

优化 EXPLAIN 功能、引入不可见索引等功能帮助提升 DBA 调试及 SQL 语句的效率。

SequoiaDB 首创专利级 STP（分布式序列时钟协议）

2020 年 10 月，巨杉数据库正式发布了 SequoiaDB v5.0 版本，其中三大黑科技：跨引擎事务一致性、原生分布式金融级容灾、多云多平台，帮助企业提高在线交易、数据中台、历史数据管理、内容管理的运维效率，降低业务开发复杂性。

在 SequoiaDB v5.0 中，巨杉数据库团队首创专利级 STP（分布式序列时钟协议），各 STP Server 之间通过 Raft 保证逻辑时钟的强一致性，每个数据库节点通过本地的 STP Client 与主 STP Server 保持逻辑时钟一致性。通过专利 STP 技术确保逻辑时钟持续向前推进，防止出现冲突，使得各数据库节点进行分布式事务处理时，只需要通过本地获取时钟即可，避免了强制所有参与节点从 GTM 中获取唯一 ID 的步骤，大大降低延迟。当前，巨杉数据库基于这一技术已经可以支持超过 4096 节点，数十 PB 级别的海量数据扩展。

另外，v5.0 版本实现了在分布式数据库中罕见的“RR（Repeatable Read）级事务隔离”，在要求 ACID 绝对保障的高并发环境下性能显著提升，在多次不同环境的客户压力测试环境下，相比原有版本性能提升近 3 倍。

发布文档详见：
https://http://doc.sequoiadb.com/cn/sequoiadb-cat_id-1432190607-edition_id-500

发布 Blog 详见：
https://mp.weixin.qq.com/s/o4ZzKzzLo3TlyPjRNAViaA

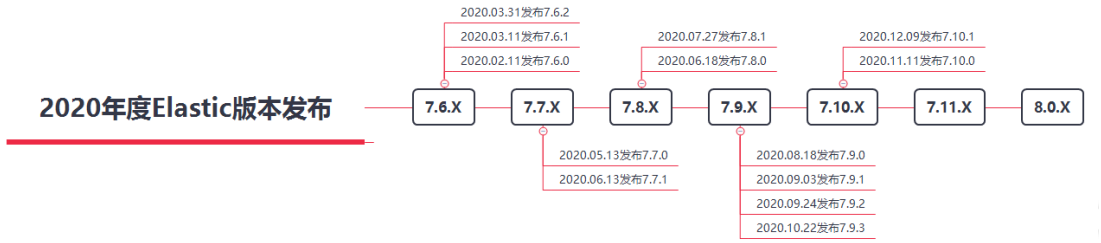
大数据生态圈

Elastic 2020 年度重大更新及技术要点分析

Elastic.co 2020 年共发布过 5 次大的版本，依然围绕 7.X 延续，从 7.6.X 到 7.10.X，至于令人期待的 8.X 依然没有明确发布日期，中间还有个 7.11.X 待正式发布，预计 2021 年第三季度会发布 8.X。

Elastic.co 版本发布有个特点，大版本发布一般是革新的跨度，中间版本发布是新功能特性持续引入，小版本发布是中间版本的修复版本。

Elastic.co 为了便于 6.X 迁移到 7.X，同时也发布了多个 6.8.X 版本，此版本为中间过渡型，不在以下整理范围之内；整理内容代表笔者个人认知观点与喜好，不代表社区或者官方。

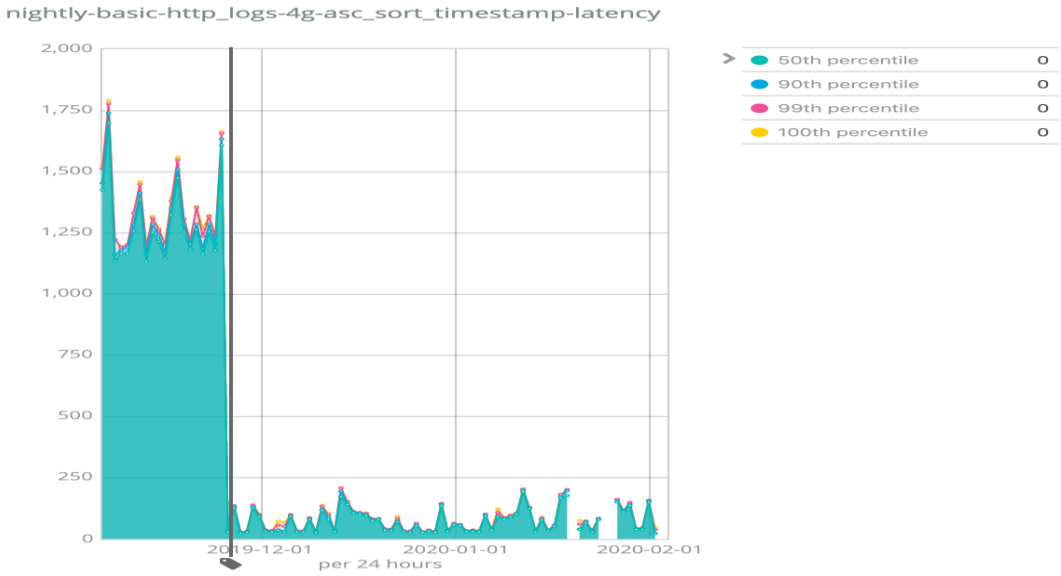


一、Elastic 7.6.x 版本

1、性能优化

1）Block-Max WAND 排序算法

- 引入了 Black-Max WAND 算法，此算法在 7.0 版本就已经集成，只是应用在分值计算领域，现已经随着 7.6.0 正式发布，应用在日期与数值类型排序领域，经测试特定性能提升 35 倍之多，非常值得大规模运用；
- 算法原理详细参考 Wiki 相关论文。



图示：官方性能测试

2）GEO 检索算法

- Elastic 比较完整的支持 GEO 地理位置检索相关特性，早期版本中，不能的检索类型采用的是不同的检索算法，在 7.6.0 版本中，将 geo-shape 多边形检索算法从 quadtree 替换到 BKD tree，性能大幅度提升；
- Elasitc 提供的 GEO 检索能力非常丰富且强大，再结合自身海量数据的倒排检索能力，值得你大规模应用在地理位置混合检索领域；
- quadtree 四叉树与 BKD tree 算法原理详细参考 Wiki 相关论文。

2、新特性

1）histogram 数据字段类型

- 直方图在大数据统计领域经常用到，特别是在基础指标监控中，经常需要记录一组数据指标，传统上会采用记录多条数据的方式；
- histogram 直方图字段类型可以更加便利的记录指标统计数据，占用空间同比大大减少，无需创建索引。

```
#创建直方图字段的索引
PUT my_index
{
  "mappings": {
    "properties": {
      "my_histogram": {
        "type" : "histogram"
      },
      "my_text" : {
        "type" : "keyword"
      }
    }
  }
}
#填充直方图字段数据
PUT my_index/_doc/1
{
  "my_text" : "histogram_1",
  "my_histogram" : {
      "values" : [0.1, 0.2, 0.3, 0.4, 0.5],
      "counts" : [3, 7, 23, 12, 6]
    }
 }
```

2）string stats aggregation 字符串聚合

新增字符串统计聚合函数，便于直接统计字段的字符串信息，无需原有通过脚本方式。

```
#字符串聚合统计函数
POST /twitter/_search?size=0
{
    "aggs" : {
        "message_stats" : { "string_stats" : { "field" : "message.keyword" }
}
    }
}
#聚合返回结果展示
{
    ...

    "aggregations": {
        "message_stats" : {
            "count" : 5,
            "min_length" : 24,
            "max_length" : 30,
            "avg_length" : 28.8,
            "entropy" : 3.94617750050791
        }
    }
}
```

3）dense vector 向量字段

向量检索可应用于很多领域，特别是图像识别，语音识别，智能问答，文本嵌入等，Elastc 早期引入 Vector 字段类型，仅仅限于实验性阶段，7.6.0 版本正式发布，对于有海量向量检索的应用项目多了一个选择。

```
#创建向量字段索引
PUT my_index
{
  "mappings": {
    "properties": {
      "my_vector": {
        "type": "dense_vector",
        "dims": 3
      },
      "my_text" : {
        "type" : "keyword"
      }
    }
  }
}
#填充向量字段数据
PUT my_index/_doc/1
{
  "my_text" : "text1",
  "my_vector" : [0.5, 10, 6]
}
#填充向量字段数据
PUT my_index/_doc/2
{
  "my_text" : "text2",
  "my_vector" : [-0.5, 10, 10]
}
```

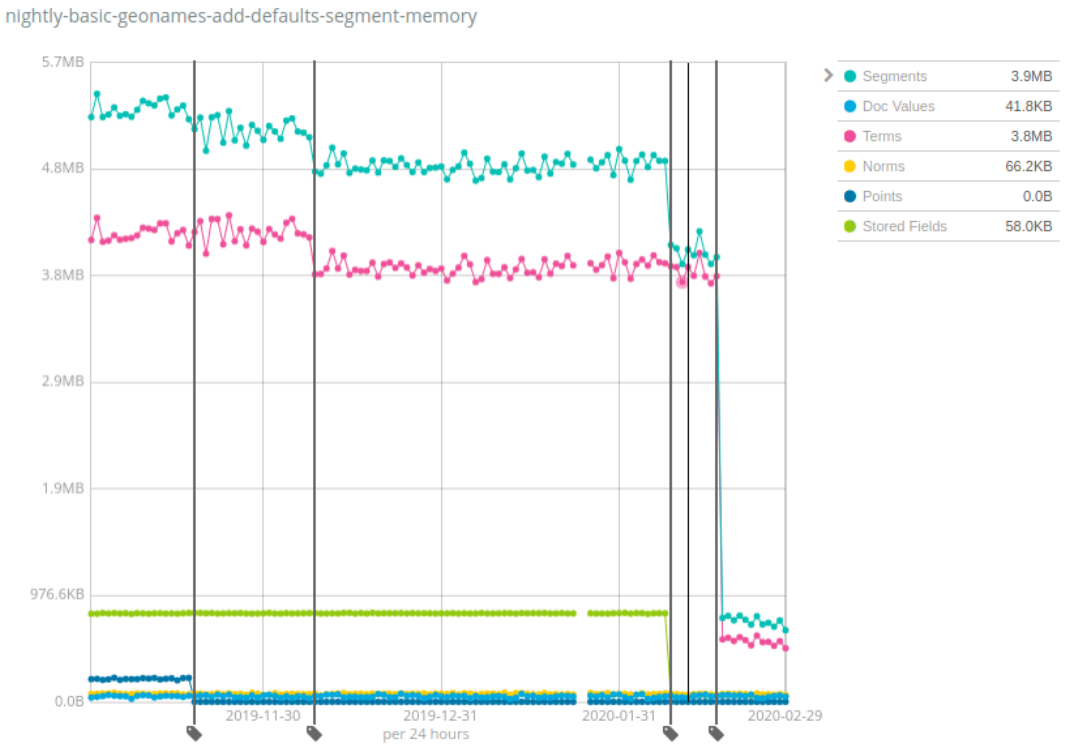
二、Elastic 7.7.X 版本推出多项新特性

Elastic 7.7.X 推出了很多新特性，非常值得掌握并运用，可非常有效提升应用效率与性能。

1、性能优化

1）off heap 堆外内存

Elasitc.co 7.7.0 版本性能优化最大的应该是 off-heap 堆外内存，将数据_id 已到堆外内存，大大降低了 JVM 堆内存占用。据官方特定数据集测试，JVM 堆内存利用率提升了 7 倍，同比之前版本可大大提升单节点挂载数据量，意味着服务器单位成本可大幅度降低。



图示：官方堆外内存测

2）jdk14 运行环境

Elastic.co 产品基于 Java 语言编写，运行于 JVM 之上，了解过 JVM GC 的同学都会习惯性的更换 G1 垃圾回收器，很多初学者以为 Elaistc 与普通 Java 应用一样，都需要配置 GC，调优 JVM，其实不然。

G1 垃圾回收器出道已经多年，在普通应用中常用，但在 ES 里面却不是，ES 团队在 JDK14 版本以前测试出过 G1 的致命 BUG，所以官方一直没有承认 G1 的可靠性，此次 JDK 版本为 14，意义重大，等同于官方认同了 jdk14 之后 G1 的可靠性。

此特性官方并未大张旗鼓的说明，原因不详。

关于 G1 在早期 BUG，可详细查阅 ES 官方博客说明。

```
#以下内容截取jvm.options
## GC configuration
8-13:-XX:+UseConcMarkSweepGC
8-13:-XX:CMSInitiatingOccupancyFraction=75
8-13:-XX:+UseCMSInitiatingOccupancyOnly

## G1GC Configuration
# NOTE: G1 GC is only supported on JDK version 10 or later
# to use G1GC, uncomment the next two lines and update the version on the
# following three lines to your version of the JDK
# 10-13:-XX:-UseConcMarkSweepGC
# 10-13:-XX:-UseCMSInitiatingOccupancyOnly
14-:-XX:+UseG1GC
14-:-XX:G1ReservePercent=25
14-:-XX:InitiatingHeapOccupancyPercent=30
```

2、新特性

1）node.transform 节点角色

新增一个节点角色，将原有数据转换的职责分离出来，设定独立的转换节点，使得可大规模应用于 OLAP 项目应用中，将实时需求与离线需求分离。

```
#设定节点数据转换角色
node.transform: true
xpack.transform.enabled: true
```

2）async search 异步检索

- ES 虽然号称检索性能宇宙第一，但在聚合统计或者是复杂检索逻辑时，慢是无法避免的，同时也没有其它产品可替代，传统上只能应用程序端采用异步调度的方式查询或者聚合，费事费力，项目逻辑复杂性增加；ES 引入异步检索之后，可大大减少应用程序端的复杂性，是开发的便利工具，也是 ES 提升性能的重要一环；
- 应用端发起异步检索，若服务端执行快，则立即返回检索结果；若超过设定时间，则返回异步检索元数据，应用端依据异步元数据再次检索即可，服务端会存储异步检索结果，有点类似应用程序的缓存。

```
#发起异步检索
POST /sales*/_async_search?size=0
{
    "sort" : [
      { "date" : {"order" : "asc"} }
    ],
    "aggs" : {
        "sale_date" : {
            "date_histogram" : {
                "field" : "date",
                "calendar_interval": "1d"
            }
        }
    }
}
# 异步检索结果
{
  "id" : "FmRldE8zREVEUzA2ZVpUeGs2ejJFUFEaMkZ5QTVrSTZSaVN3W1NFVmtlWHJsdzoxMDc=",
  "is_partial" : true,
  "is_running" : true,
  "start_time_in_millis" : 1583945890986,
  "expiration_time_in_millis" : 1584377890986,
  "response" : {
    .....
  }
}
```

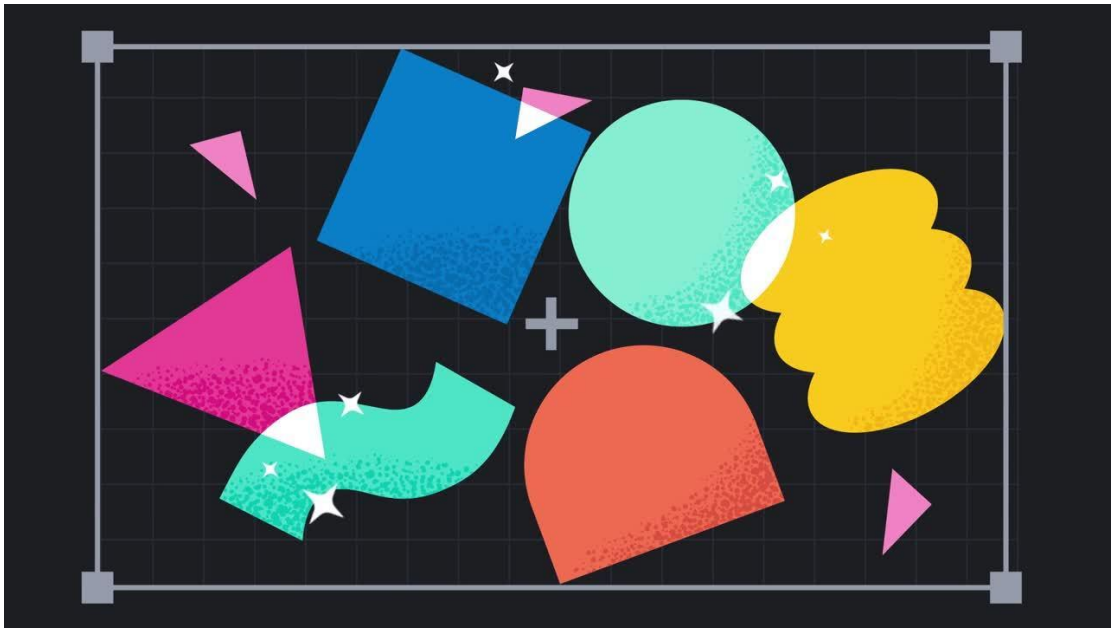
三、Elastic 7.8.x 版本比较稳定

在腾讯课堂开设了 Elastic 系列重磅课程，从课程开始到结束，周期近 4 个月，经过大量的严密测试与实战，未发现什么问题，版本相当稳定。

1、性能优化

1）geo 地理图形检索聚合增强

- 多种地理图形聚合检索更换算法，替换为 BKD tree 算法，检索效率大大提升：
- geo_bounds 聚合检索；
- geotile_grid 聚合检索；
- geohash_grid 聚合检索；
- geo_centroid 聚合检索。



图示

2）聚合内存优化

运用异步检索时，若检索的索引分片数量超过 512 个，内部会采用串型化机制维护聚合汇总结果，大大降低内存消耗。

2、新特性

1）histogram 聚合函数

7.6.X 版本增加了 histogram 数据字段类型，此字段不支持所有检索，仅仅用于内容直接展示，支持的聚合统计函数很少，7.8.X 增加了多个聚合函数，应用领域更加扩大：

- Sum 聚合函数；
- ValueCount 聚合函数；
- Average 聚合函数。

四、Elastic 7.9.x 版本

1、性能优化

1）wildcard 通配符字段

- Text 类型应用在分词检索领域，所有文本都需要先分词，再检索，性能才可以；keyword 类型是不分词，擅长碰撞检索场景，虽然也有推出模糊查询匹配机制，但性能上不上很好；
- wildcard 就是应对上述 2 种情况难全的场景，特别是日志检索领域，即使在不分词情况下，依然可以使用通配符，性能较之前是杠杠，专为性能而生；

- wildcard 字段类型详细原理参考 wiki。

```
#设置wildcard类型
PUT my-index-000001
{
  "mappings": {
    "properties": {
      "my_wildcard": {
        "type": "wildcard"
      }
    }
  }
}
#填充wildcard字段数据
PUT my-index-000001/_doc/1
{
  "my_wildcard" : "This string can be quite lengthy"
}
#查询wildcard字段数据
GET my-index-000001/_search
{
  "query": {
    "wildcard": {
      "my_wildcard": {
        "value": "*quite*lengthy"
      }
    }
  }
}
```

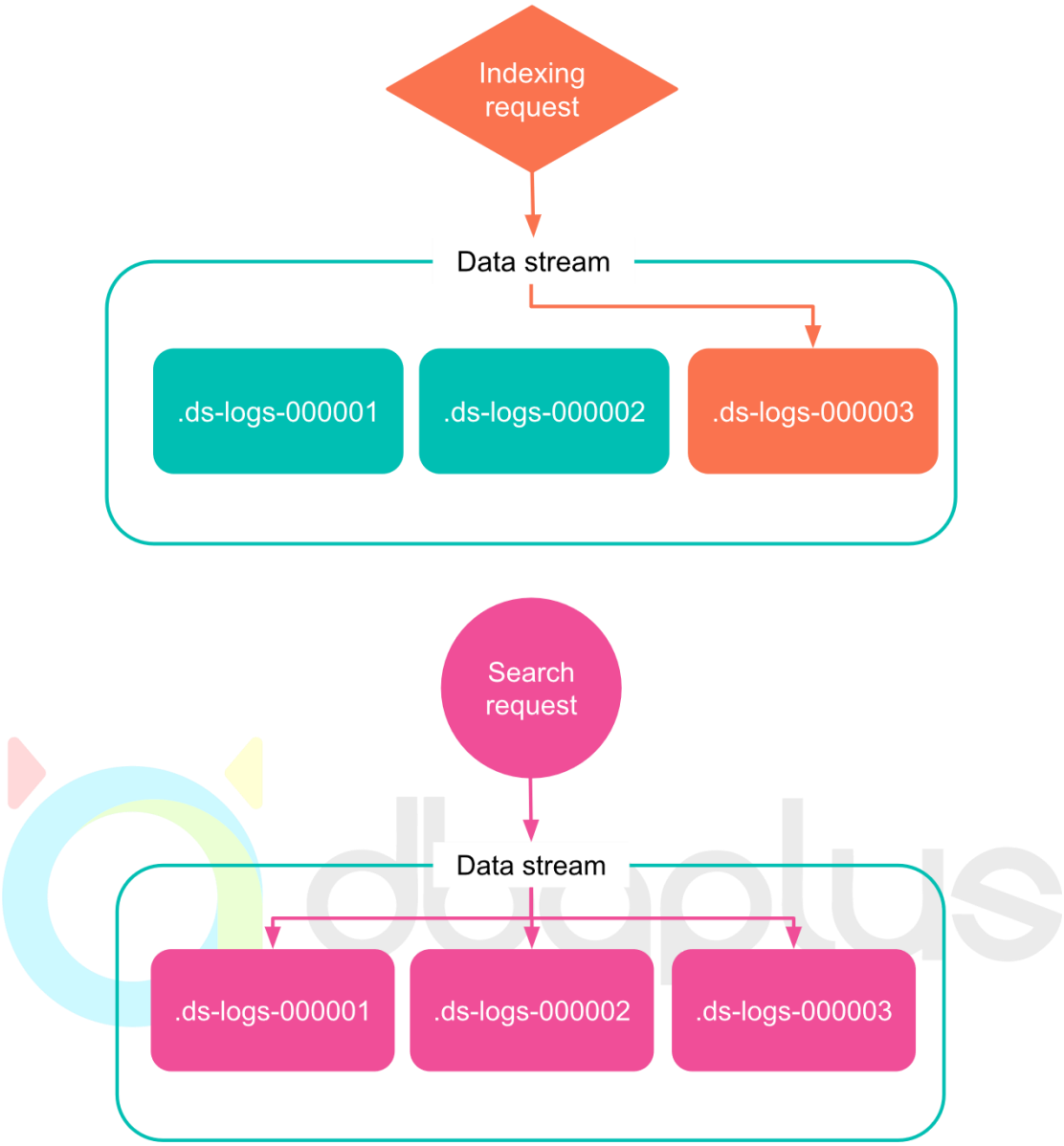
2、新特性

1）data streams 数据流

Elastic.co 虽不是原生的时序类数据产品，但在这个时序应用场景领域，同比绝大多数时序数据库更加优秀，无论是性能还是功能。

时序类应用场景，典型的数据按照时间采集存储，按照时间由近及远查询，为了满足此种诉求，早期都会借助于 Elastic 提供的索引别名特性实现，操作过程稍微复杂一些，Elasitc.co 终于意识到这是个很大的应用场景领域，顺势推出了 data streams 特性，将之前手动的别名绑定机制自动化了。

Elastic.co 产品一直有个特点，使用人员如何节约时间与资源，就集成什么样的功能。在此之前 ，使用人员可能要做很多周边工作，在此之后，仅仅简单配置即可，所以使用 ES 也必须保持着经常更新的思维。



图示：data streams 数据查询示意图

2）EQL 事件查询语言

EQL，全称 Event Query Languge，事件查询语言，专门设计用于安全领域日志检索等，相比 DSL（领域搜索语言），表达能力要更加直接，查询复杂度大大降低，比 SQL 更加流畅，非常值得应用。必须承认 DSL 依然是 Elastic.co 中最强大最全面的查询语言表达式。

```
#EQL查询语法示意
GET /sec_logs/_eql/search
{
 "query": """
  sequence by process.pid with maxspan=1h
   [ process where process.name == "regsvr32.exe" ]
   [ file where stringContains(file.name, "scrobj.dll") ]
  until [ process where event.type == "termination" ]
 """
}
```

3）node.roles 配置变更

Elasitc 支持多种角色，默认单节点具备所有节点角色，也可以依据集群规模与业务需求灵活搭配。7.9.0 版本发布之后，Elastic.co 官方虽然连配置参数都改变了，其实本质上没有什么变化 ，与 7.8.X 版本依然是兼容，在同一集群中可并行运行也不会有问题。

变化带来一些好处，首先语法上要简化一些，其次是为了后面版本推出更多的节点角色，以及具备层次的节点角色。

```
#7.9.X之前节点角色设置方式
node.master: true
node.master: true
node.voting_only: true
#7.9.0之后的节点角色设置方式
node.roles: [ data, master, voting_only ]
```

4）tableau 可视化支持

ELK 三件套融合了 kibana 可视化，Elasticsearch 数据存储，Logstash 数据采集能力，过于火爆误导很多传统 BI 人员以为 Elasticsearch 只支持 Kibana，其余的不支持，很是遗憾。7.9.0 顺势推出了 ODBC 驱动，可以支持 Tableau 可视化，终于可以结合 Tableau 丰富的 BI 能力与 Elasticsearch 的数据检索能力，强强联合。

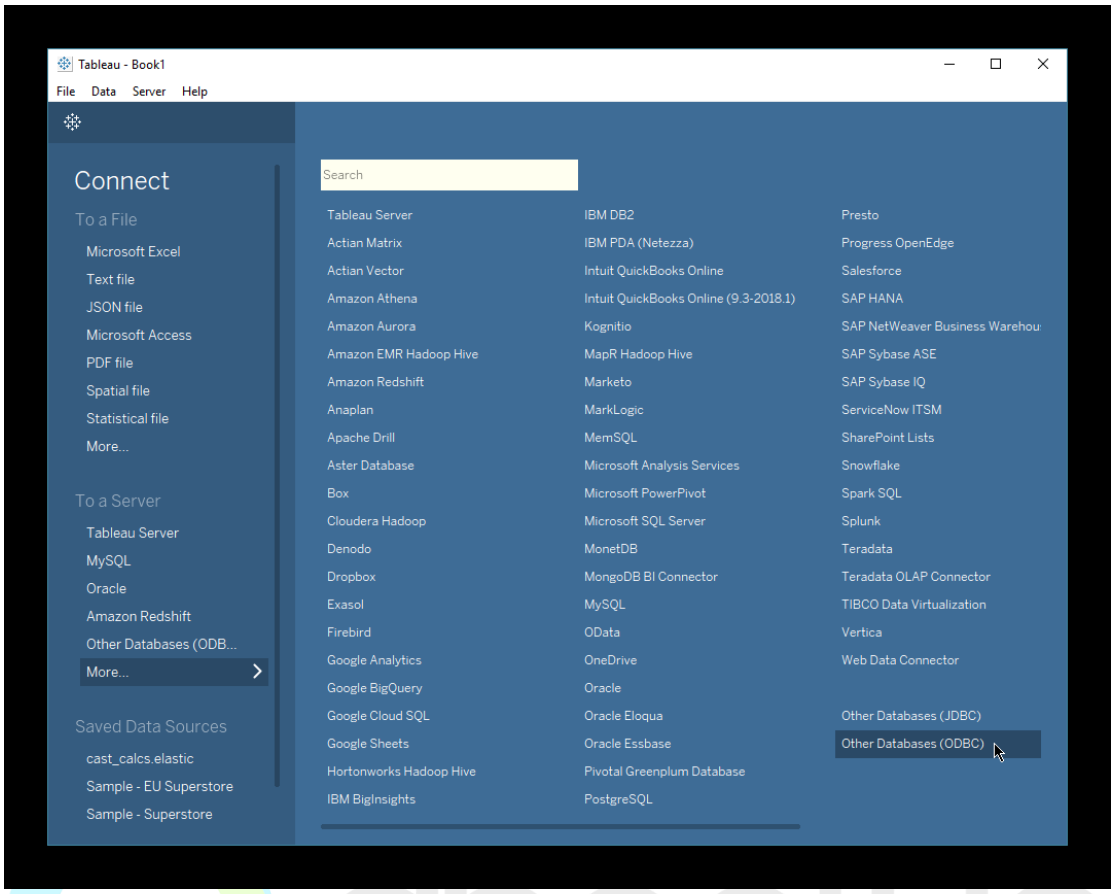


Tableau - Book1
File Data Server Help
Connect
To a File
Microsoft Excel
Text file
JSON file
Microsoft Access
PDF file
Spatial file
Statistical file
More...
To a Server
Tableau Server
MySQL
Oracle
Amazon Redshift
Other Databases (ODB...
More...
Saved Data Sources
cast_calcs.elastic
Sample - EU Superstore
Sample - Superstore
Search
Tableau Server
Actian Matrix
Actian Vector
Amazon Athena
Amazon Aurora
Amazon EMR Hadoop Hive
Amazon Redshift
Anaplan
Apache Drill
Aster Database
Box
Cloudera Hadoop
Denodo
Dropbox
Exasol
Firebird
Google Analytics
Google BigQuery
Google Cloud SQL
Google Sheets
Hortonworks Hadoop Hive
IBM BigInsights
IBM DB2
IBM PDA (Netezza)
Intuit QuickBooks Online
Intuit QuickBooks Online (9.3-2018.1)
Kognitio
MapR Hadoop Hive
Marketo
MarkLogic
MemSQL
Microsoft Analysis Services
Microsoft PowerPivot
Microsoft SQL Server
MonetDB
MongoDB BI Connector
MySQL
OData
OneDrive
Oracle
Oracle Eloqua
Oracle Essbase
Pivotal Greenplum Database
PostgreSQL
Presto
Progress OpenEdge
Salesforce
SAP HANA
SAP NetWeaver Business Warehou
SAP Sybase ASE
SAP Sybase IQ
ServiceNow ITSM
SharePoint Lists
Snowflake
Spark SQL
Splunk
Teradata
Teradata OLAP Connector
TIBCO Data Virtualization
Vertica
Web Data Connector
Other Databases (JDBC)
Other Databases (ODBC)

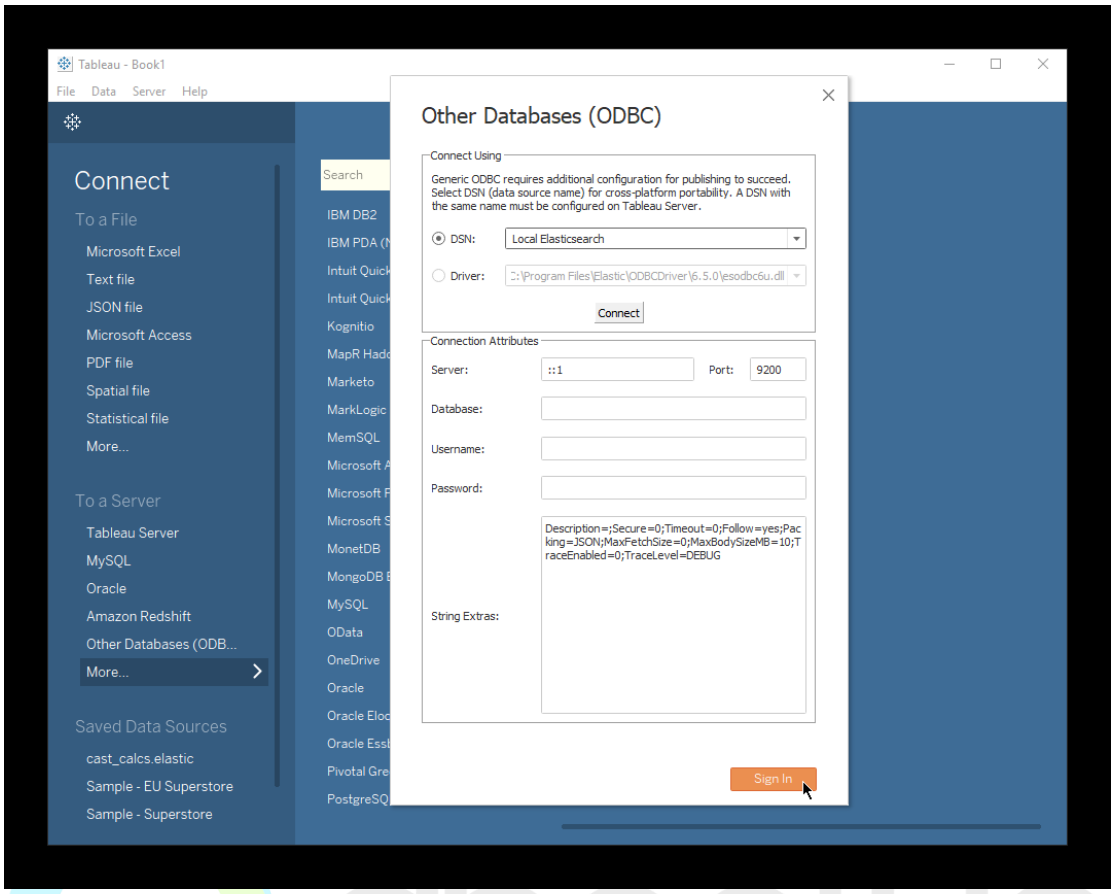


Tableau - Book1
File Data Server Help
Connect
To a File
Microsoft Excel
Text file
JSON file
Microsoft Access
PDF file
Spatial file
Statistical file
More...
To a Server
Tableau Server
MySQL
Oracle
Amazon Redshift
Other Databases (ODB...
More...
Saved Data Sources
cast_calcs.elastic
Sample - EU Superstore
Sample - Superstore
Search
IBM DB2
Other Databases (ODBC)
Connect Using
Generic ODBC requires additional configuration for publishing to succeed. Select DSN (data source name) for cross-platform portability. A DSN with the same name must be configured on Tableau Server.
DSN:
Local Elasticsearch
Driver:
Connect
Connection Attributes
Server:
::1
Port:
9200
Database:
Username:
Password:
String Extras:
Description=;Secure=0;Timeout=0;Follow=yes;Packing=JSON;MaxFetchSize=0;MaxBodySizeMB=10;TraceEnabled=0;TraceLevel=DEBUG
Sign In

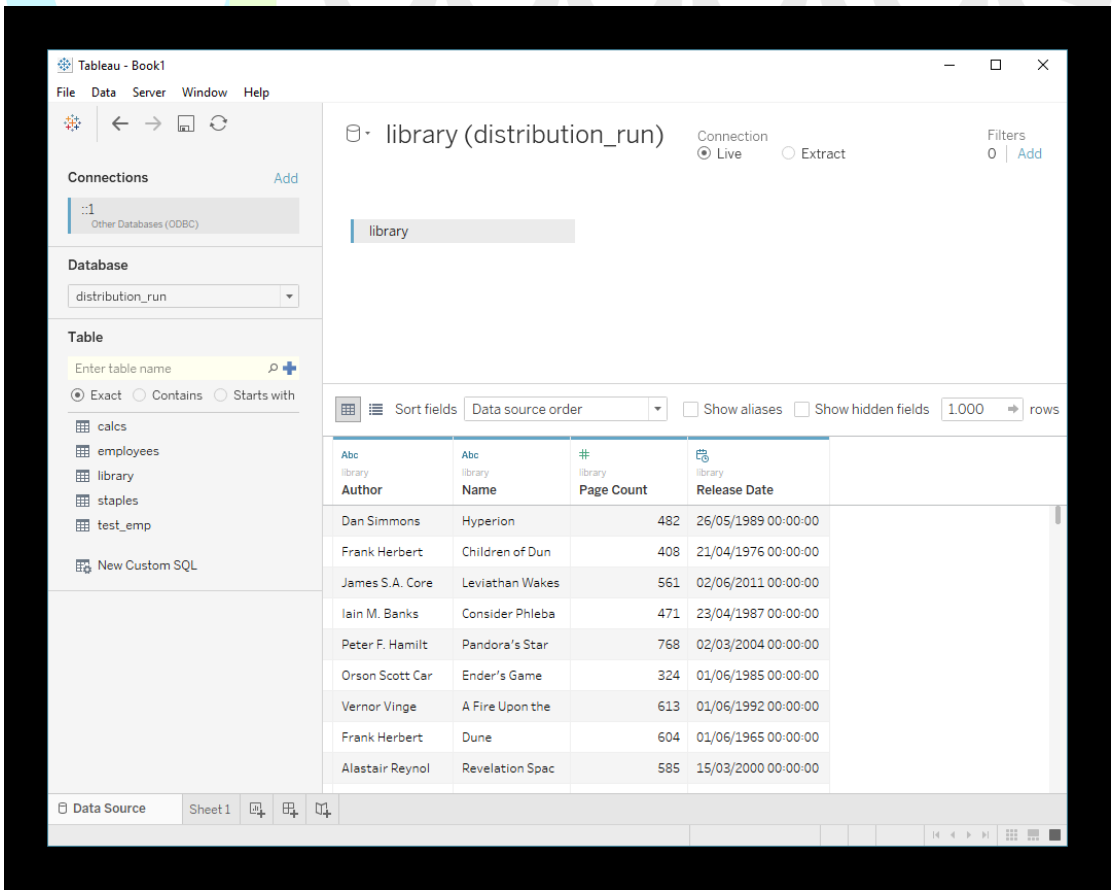


Tableau - Book1
File Data Server Window Help
Connections
Add
::1
Other Databases (ODBC)
Database
distribution_run
Table
Enter table name
Exact Contains Starts with
calcs
employees
library
staples
test_emp
New Custom SQL
library (distribution_run)
Connection
Live Extract
Filters
0 Add
library
Sort fields
Data source order
Show aliases
Show hidden fields
1.000
rows
Author
Name
Page Count
Release Date
Dan Simmons Hyperion 482 26/05/1989 00:00:00
Frank Herbert Children of Dun 408 21/04/1976 00:00:00
James S.A. Core Leviathan Wakes 561 02/06/2011 00:00:00
Iain M. Banks Consider Phleba 471 23/04/1987 00:00:00
Peter F. Hamilt Pandora's Star 768 02/03/2004 00:00:00
Orson Scott Car Ender's Game 324 01/06/1985 00:00:00
Vernor Vinge A Fire Upon the 613 01/06/1992 00:00:00
Frank Herbert Dune 604 01/06/1965 00:00:00
Alastair Reynol Revelation Spac 585 15/03/2000 00:00:00
Data Source
Sheet 1

五、Elastic 7.10.x

1、新特性

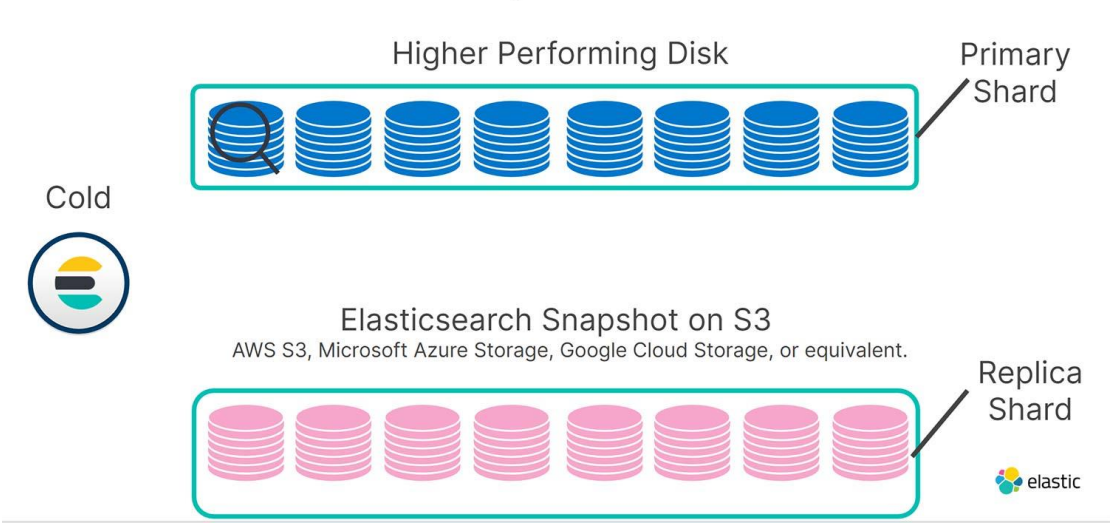
1）data tier 数据层概念

- 在以往版本里面，若要索引数据实现冷热分离，需要借助自定义标签的方式，在 7.10.0 版本之后，官方配置定义了此种标签，采用的是节点角色机制，推出了多种节点角色，代表数据属于不同的生命周期，属于不同的数据层；
- 资源隔离上也推出了官方的设置参数，可以参考历史版本对比下，大同小异。

```
#数据层次，4种角色
node.roles: [ data_content ]
node.roles: [ data_hot ]
node.roles: [ data_warm ]
node.roles: [ data_cold ]

# 索引设置 ，限制索引分布的数据层
PUT my-index-001/settings
{
 "index.routing.allocation.include._tier": "data_hot",
  "index.routing.allocation.require._tier": "data_hot",
  "index.routing.allocation.exclude._tier": "data_hot",
  "index.routing.allocation.include._tier_preference": "data_hot"
}
```

2）snapshot search 快照搜索

- 快照支持搜索绝对是最应该宣传的新功能，带来的好处不言而喻，避免原有数据还原之后搜索查询操作的麻烦，可直接搜索备份的数据，节约数据存储，数据冷热分离之后，可直接在远程更加便宜的存储介质上查询数据；
- 目前 7.10.0 推出的依然是实验性版本，预计到下个版本就可以正式使用了。



3）version datatype 版本字段类型

新推出的 version 字段类型真是方便，早期为了适应程序版本领域的检索需求，要么设定多个字段，要么采用更大的数值替代。Elastic.co 真是顺应了那句话，如何让程序员用的舒服，就怎么开发应用功能。

```
#设定版本数据字段
PUT my-index-000001

  "mappings": {
    "properties": {
      "my_version": {
        "type": "version"
      }
    }
  }
}
```

六、总结

Elastic.co 版本发布确实快，是笔者已知发布最频繁的数据产品，同比其它数据产品简直是火箭比马车，至于为什么这么快，官方也没有给出准确的解释，个人认为有以下好处：

- 已知旧版本的一些运行问题，在新版本都已经解决或者变通，解决问题的思路不是在旧版本里面寻找出路，二是看看新版本是否已经解决，若解决，进行升级即可；
- ES 越来越像一个综合型全面型的的数据产品，有自己特有的功能亮点，也融合其它数据产品的强项，在大数据产品百花齐放的当下，技术栈少融合一个技术产品，系统的成本与稳定性都可以大幅度提高，大大节约综合的成本，越快推出新功能，越快占领用户的优先选择权。

Greenplum 每月迭代一个小版本，最新发布 6.13.0

Greenplum 是基于 MPP 架构的数据库产品，它可以满足下一代大数据仓库和大规模的分析任务的需求。通过自动对数据进行分区以及多节点并行执行查询等方式，它使一个包含上百节点的数据库集群运行起来就像单机版本的传统数据库一样简单可靠，同时提供了几十倍甚至上百倍的性能提升。除了传统的 SQL，Greenplum 还支持 MapReduce，文本索引，存储过程等很多分析工具，可支持从 GB 到 PB 级的数据规模。

Greenplum 6.0 升级了对应 Postgres 版本的内核（9.4），从而获得了更多 Postgres 的兼容特性；大幅增强了 OLTP 型负载的处理能力，从而胜任流计算和 HTAP 的场景。Greenplum 6.0 的其它重要更新还包括：支持复制表，在线扩容，

磁盘配额，支持 Zstandard 压缩算法，基于流复制的全新高可用机制等。

Greenplum 6.0 自正式版发布以来，Greenplum 保持每月一个小版本的迭代速率，持续为用户提供新功能和修复补丁，目前的最新版 6.13.0。

Greneplum 6.13.0 于 2020 年 12 月 18 日发布，累积更新的新功能如下：

一、Greenplum 6.13.0

1、新增功能

内置了全新的 VMware Tanzu Greenplum Connector for Apache NiFi 1.0.0. 该连接器提供了一种快速，简单，基于 UI 的方式来为 Greenplum 数据库搭建数据摄取管道，无需构建代码。连接器可以在 VMware Tanzu Network(https://network.pivotal.io/products/pivotal-gpdb)上单独下载。

二、GPSS 升级至 1.5.0 版本

- 以前强制设置的负载配置文件 ERROR_LIMIT 属性改成了可选；
- 集成了开箱即用的 Prometheus 工具。

1、增加了 advanced_password_check contrib 模块

用户可以使用此模块为 Greenplum 数据库定制密码策略。

2、Greenplum 数据库查询优化器增加新的服务器配置参数

该参数被用于 index-only scan，此类查询仅从索引回应查询，而无需访问堆表（heap table）。此配置参数名为 optimizer_enable_indexonlyscan，并且默认情况下启用。

3、修复 bug 列表

1）集群管理

- 修复了 gpstart 的 bug，在某些情况下 cluster host 无法访问时，gpstart 可以继续启动进程；
- 修复了使用 gpconfig -s client_min_messages 设置客户端消息传递级别(如“notice”或“warning”）失效的问题；
- 修复了当用户使用 gpaddmirrors -p 选项指定了有效的端口范围时，gpaddmirrors 会生成类似这样的错误：Value of port offset supplied via -p option produces ports outside of the valid range. Mirror port base

range must be between 6432 and 61000。

2）计划器（planner）

修复了查询使用分组表达式（而非列名），且其他未分组目标引用了 group key，会导致 Postgres Planner 崩溃或返回错误结果的问题。

3）锁

改进了锁行为，以避免在 append-only(AO)表上创建多个索引时可能发生的死锁。

4）查询优化器

修复了对 Greenplum 复制表的 serial-type column 的查询（创建的 DISTRIBUTED REPLICATED）不能在所有 segment 上返回一致的值的问题。

5）执行器

- 解决了 Greenpum 数据库在 EntryDb（在主实例上运行的特殊 QE）中运行某些系统功能时有时会返回错误结果的问题；
- 解决了哈希表溢出到磁盘时可能导致 segment fault 的问题；
- 修复了当某些查询包含 merge join 和动态分布消除时，计划器会生成不正确的查询计划的问题。

6）Segment 镜像

修复了统计信息收集器收集镜像将统计信息时，内核 Recv-Q 缓冲区在 Greenplum mirror segment 实例上被占满，但由于镜像未处于热备用模式收集器未运行的问题。

7）存储：Segment 镜像

更新了 pg_rewind，以减少其执行的 lstat 操作的数量，从而提高大量数据文件的增量恢复的性能。

三、Greenplum 6.12.1

1、修复 bug 列表

1）服务器

修复了可能导致 Stand Master 节点出错关闭的内存溢出问题。

2）Postgres 计划器

修复了选择在指定一个或多个空 GROUP SETS 的查询返回错误结果的问题。

3）查询优化器

修复了当查询包含带有本地谓词和链接谓词的 index join 时，查询优化器未能执行分区消除，具有 btree 索引的表的查询时间比预期长的问题。

4）优化器

- 修复了由于使用了全表扫描而非索引导致某些 IN 查询执行缓慢的问题；
- 解决了在简化预处理过车中的约束时，GPORCA 不会考虑将列与空数组进行比较的约束的情况，此类查询会导致 GPORCA 的崩溃。

5）集群管理

- 修复了将 mirror segment 移动到备用 host 时，使用 gprecoverseg -F 失败的问题；
- 修复了使用 gprecoverseg 进行 segment 的增量恢复失败，而未记录或报告任何错误的问题。

6）analyzedb

修复了在 analyzedb 过程中，当表被删除或重建时会导致 analyzedb 失败的问题。

7）gpexpand

修复了集群扩展的重分发阶段，当尝试扩展实例化试图时，返回错误：failed to expand: error ERROR: ... is not a table 的问题。

8）Metrics Collector

- 修复了在某些情况下，GPCC 6.3.0 和 6.3.1 在查询监控器中无法显示可视查询计划的问题；
- 修复了 GPCC 无法显示 CREATE TABLE AS SELECT...FROM 和 COPY（SELECT…FROM…）TO...等语句的指标。

9）autovacuum

解决了当 autovacuum daemon 在 template0 数据库上执行 VACUUM 操作时可能发生的致命错误。

10）资源组

修复了如果一个较早的 DROP RESOURCE GROUP 命令无法删除资源组，可能会导致查询失败的问题。

11）Query Dispatcher

解决了可能导致事务错误地使用单阶段提交而非两阶段提交的问题；
解决了当 query dispatcher 需要刷新实例化视图时，可能会导致错误：ERROR: unrecognized node type: 2139062143 (copyfuncs.c:6059)的问题。

四、Greenplum 6.12.0

1、新增功能

- 现支持使用带有 gp_interconnect_proxy_address 参数定义代理端口时 segment 主机名；
- 增加了在执行 TRUNCATE 命令对参照完整性的检查；
- 支持诊断和数据收集工具 Greenplum Magic Tool（GPMT）；
- 支持 postgres_fdw PostgreSQL contrib 模块。

2、修复 bug 列表

1）Query Execution

- 当执行包含多字节字符的长查询时，Greenplum 可能会错误的截断查询字符串（删除多字节字符），并且如果 log_min_duration_statement 设置为 0，则可能随后写入无效的符号来分割日志。会导致 gp_toolkit 和 Command Center 的错误。该问题已被修复；
- 修复了在表分区上执行 CREATE UNIQUE INDEX 会隐式更改分区的分发键的问题。

2）事务

- 修复了在某些情况下，Greenplum 数据库 master 重置会生成一个或多个孤立的、准备好的事务；
- 修复了在某些情况下，由于检查点和 xlog COMMIT PREPARE 记录之间的竞争状况，Greenplum 数据库在重启后生成 PANIC 的问题。

3）查询优化器

- 修复了在某些情况下，当查询优化器尝试从包含子查询的谓词生成索引扫描时，Greenplum 数据库会出现崩溃的问题；
- 修复了由于查询优化器未生成唯一的扫描号来区分两个查询到外部表，事务中外部表上的第二个 SELECT 返回零条记录的问题；
- 修复了由于对存储区边界值与较小的 Epsilon 进行不正确的比较，查询优化器在合并 UNION 和 UNION ALL 查询中的双精度值的统计信息存储区时，进入无限循环的情况；
- 修复了当执行 TRUNCATE 和 CREATE TABLE 语句时，共享内存不足的问题。

4）Postgres Planner

- 修复了当查询的 HAVING 子句包含子查询，且 GROUP BY 列集中也未指定盖子查询中引用的一个或者多个列时，Postgres Planner 崩溃或产生不正确的结果的问题；
- 修复了为具有 GROUPING SETS 的子查询创建查询计划时可能导致 Query Dispatcher 崩溃的问题。

5）资源组

修复了无法将 pg_resgroup_get_status（NULL：oid）上的查询结果保存到表的问题。

6）gpbackup

修复了当插入由 bpchar 分发的表中，并使用旧版 bpchar 哈希运算符时，行始终使用跳转一致哈希而不是旧式（模）哈希的问题。

五、Greenplum 6.11.2

1、新增功能

- GPTEXT 版本升级至 3.4.5 版本；
- Greenplum-Spark connector 升级至 2.0.0 版本。

2、修复 bug 列表

1）管理与监控

修复了 Greenplum 从复制条目中排除外部可路由的 loopback 地址，会导致 gpinitstandby 和 gpaddmirrors 等实用程序失败的问题。

2）GPORCA

- 修复了 GPORCA 不对某些子查询使用索引扫描，可能会导致受影响的查询的性能下降的问题；
- 如果使用 CREATE TABLE .. AS 语句从使用旧式（模）哈希算法的源表中创建具有非旧式（跳转一致）哈希算法分布的表，GPORCA 将根据 gp_use_legacy_hashops 的值来分配数据；但是，它会将表的分发策略哈希算法设置为原始表的值。如果分发策略与数据分发不匹配，这可能导致查询给出不正确的结果。该问题已被解决；
- 修复了启用 gp_use_legacy_hashops 后，在为包含汇总的某些查询生成查询计划时，GPORCA 可能会崩溃的问题。

3）Metrics Collector

修复了有时过早地释放工作文件条目可能会导致 segment 上的 postmaster 进程重置，查询执行失败或 segment PANIC 的问题。

4）Query Execution

修复了 gp_toolkit 和 GPCC 的错误问题。当执行包含多字节字符的长查询时，Greenplum 可能会错误地截断查询字符串（删除多字节字符），并且如果 log_min_duration_statement 设置为 0，则可能随后写入无效的符号来分割日志，可能会导致 gp_toolkit 和 GPCC 的错误。

六、Greenplum 6.11.1

1、新增功能

PXF 升级至 5.15.1；

2、修复 bug 列表

1）查询优化器

修复了当服务器将参数 optimizer_join_order 设置为 experiative2 时，包含至少一个左或右外部联接的相关子查询导致 Greenplum 数据库主数据库崩溃的问题。

2）gpload

修复了如果表列名称包含大写字母或特殊字符，gpload 操作将失败的问题。

3）GPORCA

对于在子查询中包含外部引用的查询，例如 select * from foo where foo.a = (select foo.b from bar)，GPORCA 在取消嵌套外部引用后始终使用子查询的结果。如果子查询不返回任何行，或者子查询包含一个在外部引用之下具有多个值的投影，则可能导致崩溃或错误的结果。该问题已被解决。

4）gpfdists

修复了如果在外部表定义的 LOCATION 子句中指定主机系统时，如果外部表未使用 IP 地址，使用 gpfdists 协议访问外部表的命令将失败的问题。

5）gpstart

修复了在无法访问该备用主段时，它不会尝试启动该备用主段，从而防止了启动期间的关联堆栈跟踪。

七、Greenplum 6.11.0

1、新增功能

- GPORCA 分区消除功能已得到增强，以支持有损分配强制转换的子集，这些子集具有顺序保留（增加）功能，包括 timestamp::date and float::int；
- PXF 升级至 5.15.0；
- GPCC 升级至 6.3.0（4.11.0）版本，新增了工作量管理等其他新功能；
- DataDirect ODBC Drivers for Pivotal Greenplum 升级至 07.16.0389 (B0562, U0408)。新版本支持以下数据类型：

Greenplum Datatype	ODBC Datatype
citext	SQL_LONGVARCHAR
float	SQL_REAL
tinyint	SQL_SMALLINT
wchar	SQL_CHAR
wvarchar	SQL_VARCHAR

2、修复 bug 列表

1）资源组

修复了当运行查询由资源组管理时，由于锁问题，Greenplum 数据库在管理失控查询（使用过多内存的查询）时会生成 PANIC 的问题。

对于资源组管理的查询，由于 VMEM 使用率较高，因为计算出查询所使用的内存不正确，资源组用消息 Canceling query because of high VMEM usage 取消查询。此问题已解决。

2）VACUUM

修复了在某些情况下，运行 VACUUM 会返回错误：found xmin <xid> from before relfrozenxid <frozen_xid>的问题。

3）Segment Mirroring

修复了在某些情况下，对 Greenplum 数据库 segment 实例执行增量恢复失败，由于恢复检查点创建不正确请求的 WAL 段消息已被删除的情况。

4）analyzedb

修复了 analyticsb 尝试更新一组表的统计信息时，当分析表运行时，其中一个表被删除然后重新创建，analyticsb 会失败的问题。

5）Query Execution

修复了在运行多个查询负载很重时，某些查询随机失败并显示错误 Error on receive from seg<ID>的问题。

6）Postgres Planner

修复了在某些情况下,GPCC 取消 DROP VIEW 命令后,Greenplum 数据库生成 PANIC 的问题。

7）gpcheckcat

修复了如果安装了 gp_sparse_vector 扩展,则 gpcheckcat 会因缺少或多余的条目检查错误而失败的问题。

8）查询优化器

对于某些针对分区表的查询，当包含 partition 列的谓词执行显式转换时，GPORCA 不会执行分区消除。GPORCA 分区消除的功能已增强，以支持指定的查询类型。

9）Postgres Planner

修复了对于某些包含未指定关系的嵌套子查询且还包含嵌套 GROUP BY 子句的查询，Greenplum 数据库生成了 PANIC 的问题。

10）服务器

Greenplum 数据库不支持更改定义为分配键或具有约束的列的数据类型。尝试更改数据类型时，错误消息未明确指出原因。错误消息已更改为提供更多信息。

11）Interconnect

修复了在某些情况下 Greenplum 数据库使用代理进行互连通信(服务器配置参数 gp_interconnect_type 设置为 proxy)时,Greenplum 后台工作进程在 postmaster 进程终止后会成为孤立的进程的问题。

当 Greenplum 数据库使用代理进行互连通信(服务器配置参数 gp_interconnect_type 设置为 proxy)时,如果查询包含在 segment 实例上运行的多个并发子计划,则该查询可能已挂起。当 Greenplum 互连未正确处理并发子计划之间的通信时,查询将被挂起。此问题已解决。

12)集群管理-gpinitsystem

gpinitsystem 会在调用 gpconfig 之前导出 MASTER_DATA_DIRECTORY 环境变量,以避免在 Greenplum Database Appliance(DCA)上配置系统参数时引发警告消息。

国产数据库

本期新秀：openGauss 正式发布 1.0.1 及 1.1.0 版本

openGauss 在 2020 年 6 月建立社区，9 月和 12 月分别发布 1.0.1 及 1.1.0 版本，热度持续上升，并荣获 2020 年度国产数据库。openGauss 体现了华为特色，在过去的一年进行了大量的能力增强，快速进行产品商业化能力完善。

其中主要的亮眼特性有：

1、通用能力增强：

- OS 平台支持：支持 openEuler 20.3 LTS on X86-64 和麒麟 V10 on ARM；支持在 openEuler 和 CentOS 的容器上运行；
- 支持 ipv6 协议：数据库支持使用 ipv6 协议进行连接。

2、Oracle 及 PG 兼容性：

- LIST 分区和 HASH 分区：截止 1.1.0 版本，openGauss 支持 RANGE 分区、LIST 分区和 HASH 分区，除多级分区能力外，openGauss 已经能支持 Oracle 全部的表分区类型；
- 支持 Alter System set 语法修改数据库实例级参数：在会话内通过 alter system set 语句可以修改系统参数的值。根据不同参数的要求， 在重新连接或者重启数据库后，可以修改成功并将参数写入 postgresql.conf 和 postgresql.conf.bak 配置文件中；
- 数据类型兼容扩展：char 和 varchar 支持对 PG 模式的兼容，在计算长度时，返回字符的长度，而不是字节的长度；
- 数据类型支持 sysdate：Sysdate 返回当前日期时间，该时间为数据库所在宿主机的 Linux 操作系统时区时间；
- XML 类型：XML 数据类型可以用于存储 XML 数据。将 XML 数据存到 text 类型中的优势在于它能够为结构良好性来检查输入值， 并且还支持函数对其进行类型安全性检查。要使用这个数据类型，编译时必须使用 configure --with-libxml。
- 伪列 ROWNUM：ROWNUM 为查询出来的每一行记录生成一个序号，从 1 开始依次递增且不会重复；
- 聚合函数 median：median 返回给定数值的中值，中值是在一组数值中居于中间的数值， 如果参数集合中包含偶数个数字，函数 median 将返回位于中间的两个数的平均值；
- 全局临时表：每个数据库只需要创建一次临时表，全局临时表对象放在数据字典中。临时表(Temporary table)用于保存事务或会话期间的中间结果集，临时表中保存的数据只对当前会话可见，所有会话都看不到其他会话的数据； 即使当前会话已经提交了(commit)数据，别的会话也看不到它的数据。临时表可以是会话的，临时表中的数据可以跨提交存在，即提交之后仍然存在， 但是断开连接后再连接时数据就没有了。也可以是基于事务的，提交

之后数据就消失；

- 外部表：数据不存在于数据库中的表。通过向 DB 提供描述外部表的元数据，可以把一个操作系统文件或者外部数据源当成数据库表， 就像这些数据存储在一个普通数据库表中一样来进行访问。 外部表是对数据库表的延伸。FDW(Foreign Data Wrappers)插件允许在 openGauss 里访问其他异构数据库的表， openGauss 支持 Foreign Data Wrappers for oracle（oracle_fdw），Foreign Data Wrappers for MySQL（mysql_fdw）和 Foreign Data Wrappers for PostgreSQL（Postgres_fdw）， 从而支持在 openGauss 中访问异构其他数据库。使用这些插件需要安装相应数据库的客户端包， 需要重新编译 openGauss，在 configure 时配置 enable_mysql_fdw 和 enable_oracle_fdw。数据库启动后使用 create extension 创建扩展组件，create server 配置异构数据库连接参数， create user mapping 创建异构用户映射关系，CREATE FOREIGN TABLE 创建指定数据库的外表；
- 物化视图：物化视图是数据库查询结果数据的本地副本，存储基于数据表的数据，也可以称为快照。 在关系型数据库中普通的视图是一种虚拟（virtual）表，只是存储 SQL 语句。 而物化视图是将视图的查询结果缓存（cached）到了具体（concrete/materialized）表中，存储实际的数据。 物化视图主要用于预先计算并保存表连接或聚合等耗时较多的操作的结果，这样，在执行查询时， 就可以避免进行这些耗时的操作，从而快速的得到结果。 物化视图使用查询重写（query rewrite）机制，不需要修改原有的查询语句， 引擎自动选择合适的物化视图进行查询重写，完全对应用透明；
- 外键：外键表示了两个关系之间的相关联系。以另一个关系的外键作主关键字的表被称为主表， 具有此外键的表被称为主表的从表，外键建立了主表和从表之间的参照完整性约束；
- 自治事务：自治事务(autonomous transaction)允许你创建一个”事务中的事务”，它能独立于其父事务提交或回滚。 利用自治事务，可以开始一个新事务，完成一些工作，然后提交或回滚， 所有这些都不影响当前所执行事务的状态。自治事务约束参照规格约束的自治事务部分说明。该功能在 1.1.0 版本重构为线程间的通信方式；
- 关键字别名：关键字如 name、value、type 作为查询结果列别名；
- 全局分区索引（Global Partitioned Indexes）：全局索引就是在全表上创建索引，它是独立的索引。如果查询引用非分区字段时可以提升性能。

3、高可用能力增强

- 双机 HA 增强：支持级联备机，级联备机从备机上复制日志，减轻主机的业务处理压力；
- 备机个数扩展到 8 个：支持备机变为同步模式时间 catchup2normal_wait_time 参数可配置，备机启动与主机建立链接后，先处于日志追赶状态，等追赶的日志差距小于 catchup2normal_wait_time，把备机变为同步模式。 支持不同步配置文件，主备双机可能部署在不同规格的硬件上，主备的配置参数可能也不相同。修改原来的主备参数配置文件同步功能，支持不进行参数同步；
- 逻辑复制：在逻辑复制中把主库称为源端库，备库称为目标端数据库， 源端数据库根据预先指定好的逻辑解析规则对 WAL 文件进行解析， 把 DML 操

作解析成一定的逻辑变化信息（标准 SQL 语句），源端数据库把标准 SQL 语句发给目标端数据库， 目标端数据库收到后进行应用，从而实现数据同步。逻辑复制只有 DML 操作。 逻辑复制可以实现跨版本复制，异构数据库复制，双写数据库复制，表级别复制；

- 备机 replay 模式：主备双机同步支持 remote_apply 模式，在 remote_apply 模式下，主机需要等待备机日志 redo 恢复完才返回给应用；
- gs_basebackup 支持备机备份：gs_basebackup 支持从备机上上备份数据，减轻主机的业务处理压力；
- 支持增删备机节点：提供 om 工具，支持对备机进行在线扩容和缩容，在不影响业务状态下动态的增删备机；
- 增量备份/恢复（beta）：支持对数据库进行全量备份和增量备份，支持对备份数据进行管理，查看备份状态。 支持增量备份的合并，过期备份的删除。数据库服务器动态跟踪页面更改， 每当一个关系页被更新时，这个页就会被标记为需要备份。 增量备份功能需要打开 GUC 参数 enable_cbm_tracking，以便允许服务器跟踪修改页；
- 恢复到指定时间点（PITR）：时间点恢复(Point In Time Recovery)基本原理是通过基础热备 +WAL 预写日志 +WAL 归档日志进行备份恢复。重放 WAL 记录的时候可以在任意点停止重放，这样就有一个在任意时间的数据库一致的快照。 即可以把数据库恢复到自开始备份以来的任意时刻的状态。在恢复时可以指定恢复的停止点位置为 TID，时间和 LSN；
- 支持异构数据库事务级同步能力（限 DML）。

4、运维能力提升：

- 丰富监控维度：get_instr_unique_sql()返回视图 增加 sort&hash 关于 work_mem 方面的信息监控；
- 日志缓冲区 wal_buffer 监控，在 get_instr_wait_event 视图中添加 WAIT_EVENT_WAL_BUFFER_ACCESS 和 WAIT_EVENT_WAL_BUFFER_FULL 等待事件。其中 WAIT_EVENT_WAL_BUFFER_ACCESS 统计的是对 wal buffer 的访问次数（出于性能考虑，未统计访问耗时）；WAIT_EVENT_WAL_BUFFER_FULL 统计的是对 wal buffer 满的访问次数和访问耗时统计；
- AI 增强：对于简单查询，根据 SQL 语句的访问条件，自动推荐合适的索引；
- 权限管理模型细化：支持 DDL 权限 Grant 和 Revoke；
- 在线添加索引：通过 create index concurrently 语法，以不阻塞 DML 的方式在线创建索引；
- 安装与 OM 工具解耦：1.1.0 版本将 OM 工具与数据库内核进行了解耦，工具单独划分了仓库 openGauss-OM，后续 OM 工具代码使用该仓库进行管理。 具和内核分开打包，可以将两者镜像放到同一目录使用 OM 安装，安装方式保持不变。或者只关注内核则可以把内核镜像解压单独运行；
- 支持 WDR 自动诊断分析报告：WDR(Workload Diagnosis Report)基于两次不同时间点系统的性能快照数据， 生成这两个时间点之间的性能表现报表，用于诊断数据库内核的性能故障；
- 支持容器化部署（alpha）：提供单机数据库的容器化部署能力。先通过脚本构建数据库的 docker 镜像， 启动镜像后，可以将单机数据库以容器化方式部署运行。

5、面向开发人员的能力增强

- 支持 support plpython：支持 python 语言为 SQL 编程语言；
- 并行查询（beta）：支持并行扫描算子，优化 SQL 语句的执行方式，从单一线程，最多使用单个 CPU 运算的模式， 提升到多线程，协同完成工作的模式。并行查询消耗更多的硬件资源， 但大大提高了任务的执行效率。当前支持顺序扫描，nestloop 算子的并行查询。并在 1.1.0 版本重构为统一的并行框架；
- Interval 分区：在数据库插入表中的数据超过现有范围分区时，自动创建指定间隔的分区；
- 存储过程内 commit，单独调试和调用无参数存储过程可以省去括号：在存储过程中可以调用 commit，分批提交数据，保证数据的可靠性。 支持设置断点和单步调试，存储过程调试是一种调试手段， 可以在存储过程开发中，一步一步跟踪存储过程执行的流程， 根据变量的值，找到错误的原因或者程序的 bug，提高问题定位效率；
- UPSERT（insert on conflict do）：当插入遇到约束错误时，直接返回或者改为执行 UPDATE；
- postgis 插件：postgis 一个空间数据库，它增加了地理对象的支持， 它使用的数据类型为点、线、面，允许在 SQL 中进行空间关系位置查询。 使用该功能前需要安装 postgis 插件；
- 支持 Gin 索引：gin 索引，即通用倒排索引，是一个存储集合的数据结构。gin 索引适合用于多值类型的元素搜索，比如数组和全文检索；
- 支持存储过程调试：通过安装 pldebuger 插件，可以对函数和存储过程的 SQL 语句进行调试。

ArkDB 2020 年度重大更新及技术要点分析

一、重大更新

1. 升级兼容 MySQL 8.0.20。
2. 重构部分物理复制逻辑，提升从库复制性能和稳定性。
3. 解决 mvcc index search btree 一致性读的问题，index lock 无锁化优化。
4. 增加大量测试用例，大幅提升 ArkDB 稳定性。
5. 重构主从切换方式，修改为异步切换。
6. 修改多个内部参数变量为原子变量，防止出现并发问题。
7. ArkDB 连接层 Arkproxy 开源：https://github.com/arkdb/arkproxy
8. 优化 arkdb redo log 的对齐方式，可以提升读写性能。
9. ArkDB 热备份工具开发，用于 ArkDB 数据物理备份，同时也支持从原生 MySQL 8.0 热备份升级到 ArkDB。

二、新功能

1. 新增 Arkolap 引擎，开启数据库多模引擎，提升 ArkDB 的 olap 分析能力。
2. 支持 ArkDB、Arkolap 数据双引擎混合存储。

3. 实现 Arkolap 数据智能同步，断点续传功能。
4. 实现了一条复杂 SQL 语句在执行时，可选择在 ArkDB 和 Arkolap 任一存储引擎中查询的功能，提升并发查询效率。

三、技术要点分析

ArkDB 在 2020 开发新增了 Arkolap 列式存储引擎，支持数据行列混合双引擎存储，用户只需在建表时，标注数据需要同步到 Arkolap 引擎，ArkDB 内部同步数据到 Arkolap 引擎。在复杂的 OLAP 数据分析场景下，ArkDB 支持 TP 数据和 AP 数据的混合连接查询，且 100%兼容 MySQL 语法，保证最高兼容性的同时，ArkDB 也拥有了强大的 OLAP 分析能力。

QianBase™2020 年度重大更新及技术要点分析

易鲸捷 QianBase™ 在 2020 年进行了多个版本的产品迭代并发布了 4 个版本，1.6.0~1.6.3，其主要新增特性如下：

1、新增 binlog reader 功能（技术预览）

binlog_reader 读取一条日志，是通过 protobuf 来实现。反序列化之后，可以获取该日志对应的 RegionName，进而提取出表的名字。表的主要元数据信息：列的类型、主键的列表等会写入 binlog 表的 meta CF，表名存储在 Key。Binlog_reader 接口通过读取表的元数据信息，用于针对 ddl/dml 操作记录的查询，并为应用提供主库到备库的数据实时同步能力。

2、支持悲观锁

QianBase™悲观锁采用加锁保护机制控制并发，为了保证数据的一致性，在更新记录时，防止其他事务更新该记录，需要对行进行加锁，事务提交前会一直持有相应的锁，其他事务更新相同记录时会等待。另外 QianBase™还支持死锁检查，异常锁信息清理，锁信息诊断等高级功能。死锁检测可以防止事务之间相互等待出现死锁；锁清理功能在异常情况下，例如节点宕机，客户端异常断网等情况下，事务信息将会被自动清理，以防止因锁泄漏而导致的系统僵死；锁信息诊断可以帮助定位出现锁相关问题时快速的定位和解决相关问题。

3、增强在线 DDL 功能

QianBase™改进的版本支持了大部分常见的在线 DDL 操作。
例如，执行“CREATE INDEX”命令有几个阶段：

- 创建索引结构；
- 执行内部 INSERT/SELECT 以将表数据填充到索引中；
- 更新表的元数据，指示该表现在有一个新索引；
- 传播 QI 键以使引用表的计划重新编译。

之前版本在执行步骤 2 时，基表上没有锁。此时，写 DML（INSERT、UPDATE、DELETE）可以更改表中的数据。这种变化不会反映在索引上。

在过去，我们建议客户在 DML 处理这些对象时避免执行 DDL 对象。 但是在金融行业中，需要保证数据一致性的情况下在线执行 DDL，同时最大限度地提高数据的可用性。因此在 QianBase 新版本中提供 CQD 开关来进行控制。当 CQD TRAF_LOCK_DDL ‘ON’ 时，支持 DDL 和 DML 在线并发功能。

出于数据安全性和一致性考虑，在 QianBase™ 中还可以通过 CQD MAINTENANCE_WINDOW 来限制一些 DDL 操作，以免在不知情或不合适的时机做危险的 DDL 操作。

4、扩展 Hint 功能

QianBase™ 之前的 hint 功能支持在表名后面指定，格式如：<<+ cardinality 1e10>>，能够支持的 hint 选项较少，只有 cardinality，selectivity 和 index。但是 QianBase™ 里能影响执行计划的参数很多，CQD（Control Query Default）是影响执行计划的主要参数。

通过 CQD 我们可以来关闭某些关联类型，如：HASH_JOIN、MERGE_JOIN 等，关闭或开启并行执行计划的功能，影响优化器的某些优化指标。之前 CQD 的变动必须在执行 SQL 之前触发，这样会导致用户想得到预定的执行计划的时候要先执行额外的处理 CQD 的 SQL。

通过新的 Hint 扩展功能：

- 首先产品可以兼容 Oracle 的 hint 语法，这样可以保证同一条带有 Hint 的 sql 既可以在 QianBase™ 上执行，也可以在 Oracle 上执行；
- 其次 CQD 可以在 Hint 里设置，这样也方便了用户对需要手工优化的 SQL 的处理。

5、优化存储引擎和计算引擎，提升系统高可用性

QianBase™ 作为专门为金融行业打造的企业级数据库，高可用性是其体系结构的基础。尽管没有集成针对高可用性专门支持的商品硬件，但故障和恢复对于 QianBase™ 用户而言往往是透明的。

当无法将故障与用户隔离开时，该系统通过将故障对一个或多个特定查询的负面影响隔离并控制，从而最大限度地降低了总体影响，同时为其他查询提供了连续的可用性。同时 QianBase™ 提供了快速硬件故障检测插件软件包。 该软件包可在不到一秒的时间内检测到硬件或软件（存储引擎）故障，并加快恢复速度。 它包括扩展 Red Hat Linux 操作系统和 QianBase™ 使用的存储引擎的相关模块。

为了实现高可用性，QianBase™ 平台遵循以下设计原则：

- 避免故障：识别并消除软件中的单点故障；
- 快速恢复：检测到特定故障后，提供简单快速的机制来从故障中恢复；
- 自动快速接管：检测到故障后，如果在平台中配置了辅助或备用组件，平台

会快速自动执行切换。

6、优化 RMS 内存管理机制，提升系统稳定性

RMS（Runtime Management System）是 QianBase™用来做消息同步的系统。如：

- 某个节点上做了 DDL 操作，对应的需要更新表的消息会通过 RMS 分发到其他的节点；
- 运行时的 SQL 相关的信息也会记录在 RMS 里面，方便在各个节点上查询；
- 另外 DDL/DML 冲突检测的锁的相关信息也保存在 RMS 内。

RMS 本身是放在一块固定大小的共享内存内，如果并发执行的 SQL 量特别大，会导致 RMS 成为瓶颈，产生系统不稳定的情况。

因此新版本对 RMS 的内存做了基层保护机制，设定了两个初始阈值，并对 RMS 能够记录的内容按照重要程度做了分级。当 RMS 内存占用达到第一级阈值时，和调优相关的运行时统计信息就不再记录，当达到第二级阈值时，SQL 相关的运行时信息都不再记录。当 RMS 的内存下降到相应的阈值以下的时候，对应的功能又重新工作，以此来保证系统在高负载内存紧张的情况下仍然可以稳定运行。

7、ODBC 驱动支持 ClipVarchar

Clipvarchar 功能的设计初衷是用来减少 Server 与各类驱动之间的网络带宽。QianBase™之前的 varchar 并不是变长的，而是与 char 类型一样是定长的，当前的传输协议也不支持变长的方式。在实际的应用中存在这种情况：

表结构中有非常大的 varchar 列，但是实际插入或者存储的数据只占用了网络带宽的一小部分。这导致了明明数据量非常小但网络带宽占用十分大，从而降低了数据写入与获取得速率。ClipVarchar 功能将在客户端或者 server 发送数据之前对 varchar 字段进行裁剪，只保留实际的数据量的大小，这样在 varchar 定义非常大但是实际数据量很小的情况下可以节省大量网络带宽。

云数据库

阿里云 2020 年度多款数据库产品更新汇总

一、云原生关系型数据库 PolarDB 与分布式版 PolarDB-X 2020 年度更新

2 月：

- PolarDB MySQL 支持全局一致性：PolarDB 的读写分离模块从低到高分别提供了最终一致性、会话一致性和全局一致性共三种。

3-4 月：

- PolarDB-X 底层数据节点支持 RDS 三节点（X-DB）：通过支持 RDS 三节点（X-DB）和访问链路优化，PolarDB-X 提供了 RPO = 0 以及更加底的访问延迟。

- PolarDB MySQL 支持绑定私有地址（域名），用户可以不改数据库地址（域名）上云或迁移：PolarDB 通过绑定用户自定义的私有域名，可以在指定的 VPC 内优先解析该域名到 PolarDB 地址，从而达到用户在迁移/更换数据库的时候不需要修改原来使用的数据库地址（域名）的目的。

- PolarDB MySQL 8.0 支持透明数据加密（TDE）功能，提升数据安全防护能力：对于 PolarDB MySQL 8.0 版本，用户可以使用自己的密钥或者由阿里云自动生成密钥，通过 SQL 语句『alter table t encryption='Y';』来加密数据库的部分表，来避免物理文件或备份泄露造成的数据丢失。

6 月：

- PolarDB-O 和 PolarDB-P 发布自定义集群地址功能发布：用户可以在 PolarDB-O 和 PolarDB-P 集群上新增自定义集群地址，通过设置集群地址的读写模式、一致性级别及关联的只读节点等，来满足不同的业务场景，增强业务的灵活性。

- 云原生分布式数据库 PolarDB-X 正式发布，支持超高并发和海量数据存储能力。

7-8 月：

- PolarDB-O 发布索引增强功能：PolarDB-O 支持在分区表上并发创建索引，解决索引创建在大表中耗时过长的问题，同时支持索引添加 Invisible 状态，解决用户希望索引先失效然后确认无影响后再进行删除的需求。

9 月：

- PolarDB-X 发布两大全新企业级功能：混合负载 HTAP 和 全局二级索引透明分布式：利用 PolarDB-X 的 HTAP 智能混合负载技术、数据查询强一致

技术、资源链路强隔离技术和在线分析加速技术，PolarDB-X 可以使在线交易和在线复杂查询的性能大大提升，效率提升 5 到 10 倍以上。全局二级索引功能，可以支持多维字段拆分，提供透明分布式拆分能力，满足业务对不同维度查询拆分的诉求。

10 月：

- PolarDB MySQL 发布公测新版本 8.0.2：PolarDB MySQL 发布公测新版本 8.0.2 与社区版 MySQL 8.0.18 版本完全兼容，支持热重启（Warm BufferPool）、hash join 等功能，进一步提升 PolarDB 的可用性稳定性和面向大表查询的性能。

- PolarDB-O 提供分区表 Hash 分区增强和 oracle 兼容性开发包 dbms_debug 功能：PolarDB-O 提供分区表 Hash 分区增强，同时支持根据名称和根据指定数量自动创建分区，同时支持一级分区和二级分区，并且提供 oracle 兼容性开发包 dbms_debug 功能，通过该开发包，可以对 PL/SQL 代码层面进行调试。

11 月：

- PolarDB-O 到 PolarDB-O DBLINK 发布并上线。

12 月：

- PolarDB-O 到 Oracle 的 DBLINK 发布上线：PolarDB-O 支持 PolarDB-O 到 Oracle 的 DBLink 功能，支持公有云 PolarDB-O 用户通过 DBLINK 功能访问线下或者 ECS 自建 Oracle 的数据，满足客户 Oracle 迁移至 PolarDB-O 后跨 IDC 网络跨实例访问远端数据库数据的需求。

二、RDS & NoSQL & 管控工具 2020 年度更新

1 月：

- RDS MySQL 8.0 发布三节点企业版：RDS MySQL 8.0 发布三节点企业版形态，三节点企业版三可用区部署实例支持可用性 SLO 是 99.99%，同时实例主节点和备节点之间数据复制是强同步，提供 RPO=0 的数据库解决方案。

  本次发布的核心功能包括：
  1）8.0 三节点企业版形态，存储类型是本地 SSD 盘
  2）8.0 三节点企业版只读实例
  3）8.0 三节点企业版独享代理，支持自动读写分离

- RDS SQL Server Sa 权限账号灰度开放，解决权限诉求：线下客户使用 Sysadmin 角色，上云后可无缝适配 RDS SQL Server,适用于各大 ISV 的软件系统无缝适配上云。Sysadmin 是最 SQL Server 高级别的权限开放，用户更加自主可控。

- 云 Cassandra 支持同城多 DC 部署，支持已有单 DC 实例在线扩展成为多 DC 实例：基于多 DC 部署，用户可构建同城多 DC 容灾方案，从而具备更高的容灾等级。

2 月：

- 云数据库专属集群 MyBase 产品发布：云数据库专属集群，提供企业级云数据库解决方案。用户专享数据库集群资源池，在集群内可灵活创建 MySQL、Redis 等云数据库实例，与其他云数据库一样具备备份、监控等完全一致的能力之余，在专属集群内支持更多可扩展的管理策略，帮助企业客户解决数据库运维难题，支撑业务发展诉求。

- RDS SQL Server 2019 标准版 Beta 发布上线：SQL Server 2019 标准版 Beta 发布上线，SQL Server 2019 版提供大数据集群扩展、混合缓存池、扩展标准语言 SDK 支持等新的数据库特性，同时阿里云数据库平台加持了性能洞察、按时间点恢复、数据审计、加密等全套运维解决方案,协助客户在既有业务场景或新型机器学习、人工智能业务场景下,更好地使用及管理 SQL Server 数据库。

- RDS PostgreSQL V12 发布 plv8 2.3.13 版本：支持使用 plv8 语言进行数据库的存储过程、函数编程。

- HBase 全文索引(Solr)服务优化：HBase 增强版全文检索服务上线，通过智能集成搜索引擎 Solr，实现存储与检索的一体化使用和资源独立伸缩，支持数据统一写入、自动同步、自动 Schema 映射、历史索引并行构建等核心能力，满足海量数据下的存储、多维查询、全文索引等统一混合访问的场景需求。

3-4 月：

- MyBase 专属集群支持 MySQL 5.6 版本、SQL Server 引擎、PostgreSQL 引擎：专属集群是由多台主机（底层服务器，如 ECS I2 服务器、神龙服务器）组成的集群，用户通过专属独享一片资源池，可自由灵活分配 MySQL（现以支持 5.6、5.7、8.0 版本）、SQL Server 和 PostgreSQL 实例，进一步满足企业对安全合规、高性价比、自主可控的数据库需求。

- MySQL 支持客户自定义调整 Buffer Pool 大小：目前阿里云是首家支持该特性的中国云厂商。RDS MySQL 支持修改 InnoDB buffer pool 空间大小，innodb_buffer_pool_size 调整范围最小值 128MB，最大值是 RDS MySQL 实例规格内存*80%。目前仅支持通过公式修改，在 RDS 参数管理和 RDS 参数模板中，通过系统变量 DBInstanceClassMemory 来设置 innodb_buffer_pool_size，如 {DBInstanceClassMemory*7/10} 表示将 RDS MySQL 实例的 innodb_buffer_pool_size 设置为规格定义的内存空间的 70%大小。

- Redis 支持手动主从切换：Redis 提供服务可用性管理功能，提供分片及对应可用区信息展示，让客户能够直观感知实例分片的分布状况。提供手动主从切换功能。方便客户进行主可用区切换，同时提供容灾演练能力，降低企业级客户上云测试门槛。

- HBase 全品功能全面优化：
  1）备份计划状态告警
  2）监控指标中新增"BDS 冷存储使用空间大小"
  3）全量迁移、实时同步链路支持列、列族的过滤
  4）HBase 归档 ODPS，中间表改造分区表，节约计算资源
  5）HBase 实时归档链路支持指定起始时间
  6）标准版新增大请求限制
  7）标准版 OFFHEAP 配置简化
  8）标准版 JDK 升级，优化特定场景下 FullGC 性能

5 月：

- MySQL 库表级恢复效率提升四倍：RDS MySQL 高可用版本地盘实例，包括 5.6、5.7、8.0 三个版本，单库单表级恢复效率提升四倍，一般情况下可达 100MB/S 的恢复速度，即相当于 1 分钟可以恢复 6GB 的单表。 本次优化对于如游戏类业务场景非常友好，可以实现快速的开服回滚等业务操作。

- RDS PostgreSQL V11 支持 DDL 回收站、防火墙、增量订阅：RDS PostgreSQL V11 开放事件触发器，支持 DDL 回收站、防火墙、增量订阅等功能. 增强安全、增量复制功能。

- Redis 6.0 发布：Redis 6.0 版本在一系列关键领域进行了改进，支持新的 Redis 协议：RESP3，采用网络请求多线程处理提高了性能，并修复了之前版本的多个缺陷。 此次在阿里云上线的规格是社区云盘版，后续会逐步支持本地盘版。

6 月：

- RDS PostgreSQL 11 发布内核新版本，时序、逻辑订阅、管理等功能增强：RDS for PG 11 20200610 版本 release note: 1. 支持插件 timescaledb 1.7.1。 2. 支持 rds_superuser 使用插件 pageinspect。 3. 支持 rds_superuser 赋予其他用户 replication 权限。 4. 支持插件 decoderbufs 0.1.0。

- Redis 数据闪回功能上线：企业版 Tair 性能增强版主从规格支持数据闪回，用户可以按照指定时间点进行数据恢复。云数据库 Redis 企业版 Tair 是目前线上主推产品，为了打造更安全的，更高用户体验度，企业版需要支持按照指定时间点进行数据恢复，让用户不再受”从删库到跑路“的噩梦困扰，游戏客户更是能够从这个功能中得到巨大业务助力。

- 阿里云图数据库 GDB 兼容 OpenCypher 图查询语言：至此，GDB 已兼容 Gremlin 和 OpenCypher 两大主流查询语言。使用这两大图查询语言开发的应用，可平滑迁移到阿里云图数据库 GDB，而无需大改应用代码。

7-8 月：

- MySQL 透明数据加密支持 8.0 版本：透明数据加密 TDE（Transparent Data Encryption）可对数据文件执行实时 I/O 加密和解密，数据在写入磁盘之前进行加密，从磁盘读入内存时进行解密。TDE 不会增加数据文件的大小，开发人员无需更改任何应用程序，仅需要在 RDS 安全性中打开 TDE 即可使用。本次上线支持 8.0 高可用版本地盘实例。

- RDS PostgreSQL V12 发布模糊查询加速插件 pg_bigm，支持任意字符集模糊查询，包括中文模糊查询的索引加速：RDS PostgreSQL V12 提供毫秒级模糊查询，支持任意字符集、任意字符个数的模糊查询。

- 阿里云 Redis 企业版集群架构慢日志透传客户端 IP：集群架构由于采用代理模式，在查询慢日志的时，无法看到真实的客户端 IP。本次发布后，Redis 企业版集群架构慢日志可以透传客户端 IP，方便用户对问题的直接定位。注：本次发布不涉及社区版集群代理架构。

- 阿里云数据库 Cassandra 版新增支持 30 余个 nodetool 查询命令：nodetool 是 Cassandra 数据库的重要系统管理工具，通过 nodetool 查询命令可以查看到数据库集群的详细信息，如集群节点状态、哈希环状态、数据迁移进度、表状态、核心线程池状态、compaction 任务详情等。方便用户管理 Cassandra 数据库，快速定位性能问题等。

9 月：

- 专属集群 MyBase 支持 MongoDB，MySQL 企业级三节点，MyBase 实现 5 大主流开源数据库引擎的全部支持：MongoDB 支持云数据库专属集群 MyBase 部署形态，提供用户级别资源隔离，灵活的资源部署能力，通过合理的超配设置可以有效降低企业资源成本，满足中大型企业级客户大规模数据库部署需求。

10 月：

- 云数据库 SQL Server 版发布跨地域备份功能，国内首家支持 SQL Server 跨地域域物理备份容灾能力，解决 ISV 等行业核心业务数据容灾诉求：云数据库 SQL Server 版发布支持跨地域备份功能，用户可将实例备份数据跨地域进行多分转储。用于业务数据容灾，防范地域级别灾害导致的数据丢失。

- 云数据库 Redis 版发布数据闪回功能，业界独家支持一键秒级恢复数据，可应用于金融、游戏等行业，提升数据安全：传统的 Redis 社区版备份恢复均采用定时（一般为每天或者每小时）快照机制，系统开销高的同时也无法做到高精度的数据恢复，数据安全存在一定风险。阿里云 Redis 企业

版实例（性能增强系列）业界独家支持数据闪回功能，最长可恢复 7 天内任意时间点（秒级）的 Redis 数据，避免误操作带来的数据损失，极大降低了运维复杂度，实时保护用户数据，此功能亦可以应用于游戏行业数据回档业务。

11 月：

- 云数据库专属集群 MyBase 发布 MySQL 5.7 单机基础版并且开放了 change master 权限：可以利用本版本构建混合云多云的实时灾备方案，利用源生 MySQL 复制做数据同步。同时结合 MyBase 本身的 CPU 超配能力，可最大限度优化成本投入，帮助客户实现整体构建多云混合云最优部署成本。

- 专属集群 MyBase 支持 Windows 主机创建管理员账号，业界独家实现 Windows 体系自主可控能力数据库服务：云数据库专属集群 Windows 主机支持创建管理员账号（administrator），用户通过管理员账号可在云数据库主机进行更多自定义操作，保留企业自身的运维及管理习惯，最大程度实现自主可控。

- RDS MySQL 独享代理地址支持 SSL 加密，对开启只读实例的客户，SSL 加密可提升自动透明读写分离的连接安全性，保障数据库安全：RDS MySQL 提供数据库独享代理服务，基于独享代理提供如读写分离、连接池、事务拆分等高级功能，独享代理提供独享代理地址，本次功能发布支持对该地址 SSL 加密，保障数据传输安全。数据库独享代理，目前支持 RDS MySQL 5.6、5.7、8.0 高可用版和三节点企业版，适用于有只读实例分担流量请求的业务场景。

12 月：

- RDS PostgreSQL 国内首发 V13 大版本，具体功能如下：
  1）通过 de-duplicate 提升 btree 索引性能，压缩索引空间；
  2）聚合运算性能提升；
  3）分区表性能提升；
  4）通过多字段组合统计信息提升 PLAN 优化能力；
  5）单个 table 的多个 index 支持并行 vacuum；
  6）支持 incremental sort。

- 阿里云 Redis 全球分布式缓存功能发布，实现数据异地灾备和多活：发布 Redis 全球分布式缓存产品，该产品是基于云数据库 Redis 自研的多活数据库系统，为您提供跨域复制（Geo-replication）能力，可实现数据异地灾备和多活，帮助用户解决业务因跨地域访问导致延迟大的问题。

- MongoDB 发布 4.4 版本：MongoDB 4.4 是 MongoDB 公司 2020 年推出的最新稳定大版本，被称为是用户驱动的全面提升版本，目前阿里云在云厂商首先发布了最新 4.4 版本，带来隐藏索引，Refinable Shard Keys，复合哈希分片键， 对冲读，镜像读，Union 聚合等多项新功能，产品的可用性，易用性，性能，容错性，扩展性都得到了全面提升。

腾讯云 2020 年度 12 款数据库产品更新汇总

一、腾讯云 MySQL 8.0 2020 年度重大更新及技术要点分析

腾讯云 MySQL 8.0 中针对性能提升的特性有：

1、快速加列

加字段需要对表进行重建，尤其是对亿级别的大表，DBA 会反馈开发，表太大，加不了，那么开发就得重新调整业务逻辑，势必增加了工作量。

虽然 Online DDL 可以避免锁表，但如果在主库上执行耗时 30 分钟，那么再复制到从库上执 行，主从复制就出现延迟。使用 instant ADD COLUMN 特性，弹指间，字段就加好了，享受 MongoDB 那样的非结构化存储的灵活方便,无形中减少了开发的工作量。

限制：

- 如果指定了 AFTER，字段必须是在最后一列，否则需要重建表；
- 不适用于 ROW_FORMAT = COMPRESSED；
- DROP COLUMN 需要重建表；
- modify 修改字段属性需要重建表。

2、异步删除大表

对大表进行删除时，由于会影响 IO 性能，往往会选择夜间或者业务低峰期，尽量降低大表删除带来的性能影响。不过，依然无法从根本解决删除大表所带来的 IO 性能骤降的问题，对业务产生较大的影响。

使用异步删除大表特性，能够逐步的删除大表，将大表的删除速度控制在不影响业务的程度，而对于前端是无感知的，删除的同时显著的降低了对业务的影响。

3、SQL 限流

数据库是大部分应用的核心依赖。为防止数据库压力过大，一般都会在应用端做优化和控制。但在以下场景，也需要在数据库端做优化控制：

场景	说明
SQL 并发急剧上升	例如缓存穿透或异常调用，可能会导致 SQL 并发量突然上升
访问表未创建索引	例如特定表访问量特别大，并且没有创建索引，导致整体系统繁忙
表内数据倾斜严重	例如各省市人口详细信息差异较大，

	相同的 SQL 查询不同的省份导致整体系统繁忙程度不同

可以通过创建 SQL 限流任务，自主设置 SQL 类型、最大并发数、限流时间、SQL 关键词，来控制数据库的请求访问量和 SQL 并发量，进而达到服务的可用性，不同的任务之间不会发生冲突。

- SQL 类型：包含 select、update、delete、insert、replace；
- 最大并发数：为 SQL 最大并发数，当包含关键词的 SQL 达到最大并发数时会触发限流策略。如果该值设为 0，则表示限制所有匹配的 SQL 执行；
- 执行方式：支持“定时关闭”和“手动关闭”；
- 限流时间：选择“定时关闭”时，需选择 SQL 限流的生效时间；
- SQL 关键词：为需要限流的 SQL 关键词，当包含多个关键词时，需要以英文逗号分隔，逗号分隔的条件是逻辑与的关系，且逗号不能作为关键词。

通过以上设置即可实现 SQL 的限流，对数据库实现优化控制，实现运行负载保障。

4、热点更新保护

大量请求同时更新数据库中的同一行数据，update 操作给表加上行锁，导致后面的请求全部排队等待前面一个 update 完成，释放行锁后才能处理下一个请求。大量后来请求等待，占用了数据库的连接。一旦数据库连接数被占满，就会导致后来的全部请求因拿不到连接而超时，业务请求出现无法及时处理的情况，数据库系统的 RT 会异常飙高，业务层由于等待出现超时，应用层的连接耗尽，一系列的雪崩效应。

解决热点数据并发更新需要注意核心问题：减少直接对 DB 层数据热点的并发更新。而多数直接减少对 DB 层数据热点的并发更新的修改，很难快速在应用层面进行处理和解决。使用热点更新保护功能，针对语句的排队机制，尽可能把具有相同冲突的语句放在内存队列排队，通过开启热点更新保护减少锁冲突的开销，提高高并发场景的数据库性能。

创建热点更新保护任务，可以自主设置等待超时阈值、执行方式，其中执行方式包括定时关闭和手动关闭，在定时关闭的执行方式下，可以自主设置执行时间：

腾讯云 MySQL 中针对安全提升的特性有：

1、支持透明加密

云数据库 MySQL 提供透明数据加密（Transparent Data Encryption，TDE）功能，透明加密指数据的加解密操作对用户透明，支持对数据文件进行实时 I/O 加密和解密，在数据写入磁盘前进行加密，从磁盘读入内存时进行解密，可满足静态数据加密的合规性要求。

未开启透明加密：

- 下载备份文件；

- 解压至指定目录；

```
[root@VM_17_57_centos ~]# ll /data
total 30680
-rw-r----- 1 root root      396 Nov 27 16:50 backup-my.cnf.qp
drwxr-x--- 2 root root     4096 Nov 27 16:50 hello
-rw-r----- 1 root root  1005586 Nov 27 16:50 ibdata1.qp
-rw-r----- 1 root root   165279 Nov 27 16:50 ibtmp1.qp
drwxr-x--- 2 root root     4096 Nov 27 16:50 mysql
drwxr-x--- 2 root root     4096 Nov 27 16:50 performance_schema
drwxr-x--- 2 root root     4096 Nov 27 16:50 test
-rw-r----- 1 root root 30037154 Nov 27 16:50 undo001.qp
-rw-r----- 1 root root      165 Nov 27 16:50 xtrabackup_binlog_info.qp
-rw-r----- 1 root root      390 Nov 27 16:50 xtrabackup_cdb_result.qp
-rw-r----- 1 root root      188 Nov 27 16:50 xtrabackup_checkpoints.qp
-rw-r----- 1 root root      732 Nov 27 16:50 xtrabackup_info.qp
-rw-r----- 1 root root   108488 Nov 27 16:50 xtrabackup_logfile.qp
-rw-r----- 1 root root      197 Nov 27 16:50 xtrabackup_slave_info.qp
```

- 修改启动参数文件；

```
vi /data/backup-my.cnf
```

```
# This MySQL options file was generated by innobackupex.

# The MySQL server
[mysqld]
innodb_data_file_path=ibdata1:12M:autoextend
innodb_log_files_in_group=2
innodb_log_file_size=536870912
innodb_undo_directory=.
innodb_undo_tablespaces=0
server_id=0
#innodb_checksum_algorithm=innodb
#innodb_log_checksum_algorithm=innodb
#innodb_fast_checksum=false
#innodb_page_size=16384
#innodb_log_block_size=512
#redo_log_version=0
```

- 启动数据库；

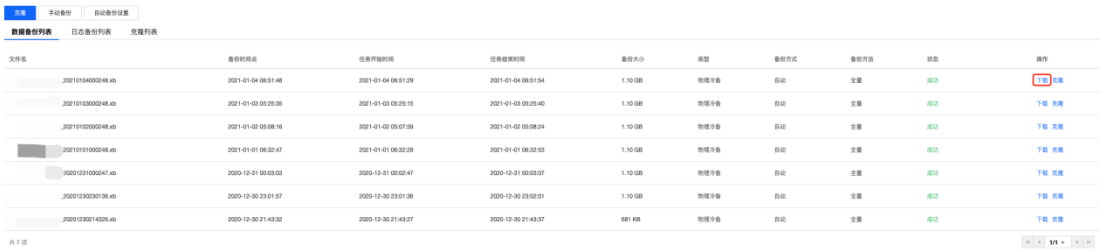
```
mysqld_safe --defaults-file=/data/backup-my.cnf --user=mysql
--datadir=/data &
```

- 数据库库被正常访问；

```
[...154822]$ mysql -S /tmp/mysql_3307.sock
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.6.23 Source distribution

Copyright (c) 2000, 2011, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| test               |
+--------------------+
2 rows in set (0.00 sec)

mysql>
```

开启透明加密：

- 下载备份文件；



- 解压至指定目录；

```
[root@VM-32-2-centos data]# ls -l
总用量 24928
-rw-r--r-- 1 root root      427 1月   4 17:01 backup-my.cnf
-rw-r----- 1 root root      415 1月   4 17:00 backup-my.cnf.qp
drwxr-x--- 2 root root     4096 1月   4 17:01 ccm
-rw-r--r-- 1 root root 12582912 1月   4 17:00 ibdata1
-rw-r----- 1 root root   966381 1月   4 16:59 ibdata1.qp
drwxr-x--- 2 root root     4096 1月   4 17:01 mysql
drwxr-x--- 2 root root    12288 1月   4 17:01 performance_schema
drwxr-x--- 2 root root    16384 1月   4 17:00 sys
drwxr-x--- 2 root root     4096 1月   4 17:00 test
-rw-r--r-- 1 root root 11534336 1月   4 17:00 undo001
-rw-r----- 1 root root   355449 1月   4 16:59 undo001.qp
-rw-r--r-- 1 root root       75 1月   4 17:01 xtrabackup_binlog_info
-rw-r----- 1 root root      158 1月   4 17:00 xtrabackup_binlog_info.qp
-rw-r----- 1 root root      121 1月   4 17:00 xtrabackup_checkpoints
-rw-r--r-- 1 root root     1026 1月   4 17:00 xtrabackup_info
-rw-r----- 1 root root      887 1月   4 17:00 xtrabackup_info.qp
-rw-r--r-- 1 root root     2560 1月   4 17:01 xtrabackup_logfile
-rw-r----- 1 root root      622 1月   4 17:00 xtrabackup_logfile.qp
-rw-r--r-- 1 root root      114 1月   4 17:01 xtrabackup_slave_info
-rw-r----- 1 root root      196 1月   4 17:00 xtrabackup_slave_info.qp
```

- 修改启动参数文件；

```
vi /data/backup-my.cnf
```

```
# This MySQL options file was generated by innobackupex.

# The MySQL server
[mysqld]
innodb_data_file_path=ibdata1:12M:autoextend
innodb_log_files_in_group=2
innodb_log_file_size=536870912
innodb_undo_directory=.
innodb_undo_tablespaces=0
server_id=0
#innodb_checksum_algorithm=innodb
#innodb_log_checksum_algorithm=innodb
#innodb_fast_checksum=false
#innodb_page_size=16384
#innodb_log_block_size=512
#redo_log_version=0
```

- 启动数据库；

```
mysqld_safe --defaults-file=/data/backup-my.cnf --user=mysql --datadir=/data &
```

- 访问加密数据报错。

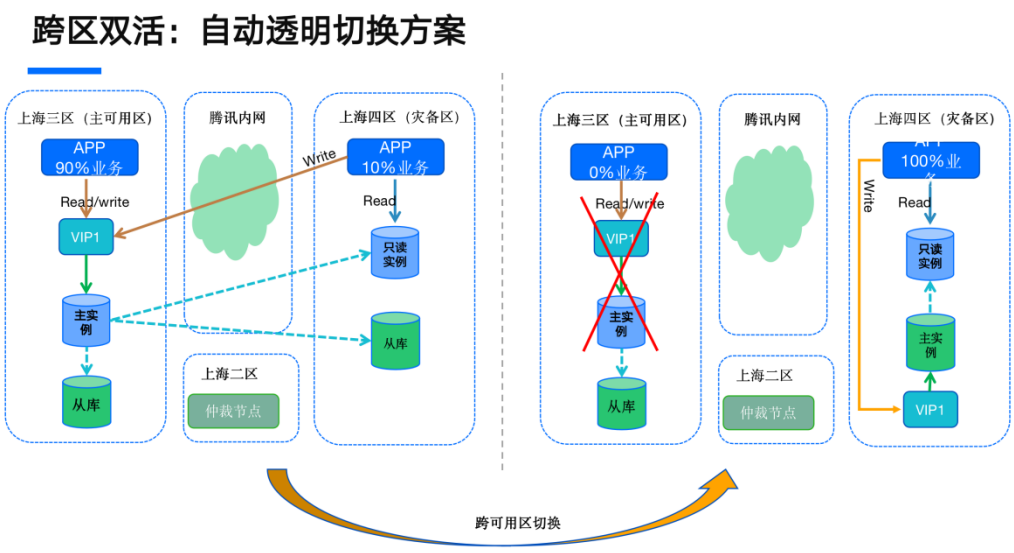
```
[ERROR] InnoDB: Encryption information in datafile: ./tencent/emp.ibd can't be decrypted , please confirm the keyfile is match and keyring plugin is loaded.
[Warning] InnoDB: Ignoring tablespace `tencent/emp` because it could not be opened.
```

从以上测试可以看出，经过加密的数据文件，就算能够完整的恢复后也无法访问加密的数据，从而保障了数据安全性。

腾讯云 MySQL 中针对可用性提升的特性有：

1、多可用区容灾

针对业务连续服务和数据可靠性有强需求或是监管需要的场景，云数据库 MySQL 提供跨地域灾备实例，帮助用户以较低的成本提升业务连续服务的能力，同时提升数据的可靠性。



二、腾讯云数据库 PostgreSQL 2020 年度功能更新

2020 年，腾讯云数据库 PostgreSQL 每月均发布有版本，其中在 3 月份发布了 Serverless 化的 PostgreSQL 功能，主从延迟同步优化功能和只读实例功能。

功能速递：

- Serverless 实例支持灵活的资源弹性分配；
- 主从同步优化解决了社区仍未解决的大量 drop 场景下的同步日志堆积问题；
- 只读实例能有效的降低用户主实例的数据访问压力，实现读写分离的能力。

三、腾讯云 Redis 支持多可用区部署

2020 年 12 月，腾讯云数据库 Redis 发布了跨可用区部署功能，支持同 Region 下跨任意可用区部署，同时针对缓存场景对延迟的苛刻要求，提供故障本地就近切换，就近只读功能。在提升服务可用性的同时，保障缓存的读取延迟。

四、云数据库 Tendis 正式发布

12 月，腾讯云数据库 Tendis 正式发布，Tendis 是腾讯云自研、100%兼容 Redis 协议的数据库产品，作为一个高可用、高性能的分布式 KV 存储数据库，从访问时延、持久化需求、整体成本等不同维度的考量，推出了存储版和混合存储版的不同产品形态，为客户提供更丰富的数据结构和更灵活的存储方式，满足不同场景下的业务需求。

- 存储版：兼容 Redis 核心数据结构与接口，可提供大容量、低成本、强持久化的数据库服务，适用于兼容 Redis 协议、需要大容量且较高访问性能的温数据存储场景；

- 混合存储版：由缓存 Redis 和引擎 Tendis 两大组件构成，适用于 KV（key-value）存储场景，通过将冷数据从内存驱逐，并在磁盘存储全量数据的方式，平衡了存储场景中性能和成本之间的难题，在冷数据占比较大的场景中可帮业务降低多达 80%的运营成本。

Github 开源地址：https://github.com/Tencent/Tendis

五、腾讯发布全 Serverless Database 架构的云原生数据库 TDSQL-C（原 CynosDB ）

- 全 Serverless Database：2020 年 12 月腾讯 Techo 大会发布了国内首个计算和存储全 Serverless 架构的云原生数据库 CynosDB Serverless；
- 支持数据库根据业务负载自动启停，无感扩缩容，并按实际使用的计算和存储资源计费，不用不付费；
- 满足小程序云开发等 SaaS 应用，IoT 和边缘计算等不确定负载，以及开发测试等低频使用场景。

六、时序数据库 CTSDB 超大集群架构升级

2020 年，CTSDB 持续更新，其中在 9 月份版本更新的超大集群架构升级新特性最值得关注。

- 超大集群架构升级

CTSDB 在 9 月份进行架构升级，在集群中节点数量超过 30 个的超大规模场景，可以将当前的通用集群架构优化升级为混合节点集群架构，充分保证多节点超大集群的性能稳定；

- 多伦多一区上线

随着 CTSDB 上线多伦多一区，时序数据库 CTSDB 已经上线华北、华东、华南和北美等地区。

发布动态详见：
https://cloud.tencent.com/document/product/652/48652

七、云数据库 MongoDB 版本更新总结

1、腾讯云 MongoDB 新版本迁移产品 newDTS for MongoDB 1.0 版本发布

7 月份，由腾讯云 MongoDB 自研的新版本迁移产品 newDTS for MongoDB 1.0 版本发布，具备跨版本迁移、自动重试、断点续传等企业级特性。

具体特性：

- 数据迁移：通过 DTS 迁移服务实现全量迁移 + 增量实时同步；
- 库表迁移：支持库表级别迁移；
- 迁移可用性：支持自动重试机制 + 断点续传；

- 时延显示：支持增量同步时延显示；
- 支持场景：上云迁移、云上自建迁移、其他云厂商 MongoDB 服务迁移、云实例间迁移。

2021 年 newDTS for MongoDB 将支持更多特性，如跨架构迁移、异构迁移如迁移到 es 等场景，满足用户更多需求。

2、11 月份腾讯云 MongoDB 正式发布了单节点版本

11 月份，腾讯云 MongoDB 正式发布了单节点版本，在保障性能的同时，运营成本仅为集群版本 1/3，适用于个人或企业学习、业务内部测试环境等场景。

3、12 月份腾讯云 MongoDB 发布了跨可用区版本

12 月份，腾讯云 MongoDB 发布了跨可用区版本，将支持同 region 下将实例的节点分布到任意不同的可用区，实现单实例跨区容灾的能力，满足企业的容灾需求。

八、云数据库 Tcaplus 版本更新总结

1、TcaplusDB 发布 3.46 版本

7 月，TcaplusDB 发布 3.46 版本，支持全局索引能力，客户可以在使用 PB 级存储、毫秒时延的 KV 数据库服务的同时，享用数据的聚合查询能力，支持在海量并发、不锁表、异步构建索引。

- 全局索引：非实时索引，正常情况下秒级延时，即数据更新后，一秒后就能够通过全局索引查询出来；全局索引提供了条件查询，模糊查询，聚合查询，分页查询等功能，如 a > 100 and b < 1000; name like "test%"; limit 10 offset 10; sum(a), avg(a)等。

2、TcaplusDB 发布 3.50 版本

10 月，正式发布了 TcaplusDB 的 3.50 版本，支持本地开发环境，具体内容如下：

- 利用 TcaplusDB 的 Local 版本，支持本地开发环境，客户可以在不访问 TcaplusDB 云服务环境的情况下开发和测试应用程序；
- Local 版本帮助客户节省吞吐量、数据存储和数据传输费用。客户在开发应用程序时无需 Internet 连接；
- Local 版可提供免费的下载，并通过 Docker 镜像提供服务。具有单示例即可服务、一键安装、一键启服的便捷使用能力。

九、腾讯云 TDSQL-A ClickHouse 版重磅上线

12 月，发布了腾讯云 TDSQL-A ClickHouse 版。TDSQL-A ClickHouse 版基于 ClickHouse 引擎，提供高性能，高可用和低成本的易用实时分析数据库解决方案。

- TDSQL-A ClickHouse 版与社区版高度兼容，且做了大量优化和云上适配，尤其对数据导入做了大量优化；
- 基于数据传输服务 DTS 的强大数据同步能力，能稳定高效将 OLTP 的数据同步到 ClickHouse，提供 OLTP 到 OLAP 的无缝联动，并能和已有腾讯云 IT 架构融为一体。

十、腾讯云 TDSQL PostgreSQL 版（原 TBase OLTP 版）

2020 年 9 月底发布 Oracle 兼容版本，整体 Oracle 兼容性 95%以上。可以帮助用户尽可能平滑地从 Oracle 迁移到腾讯云数据库上。

- 新发布的 Oracle 兼容版高度兼容 Oracle 语法，全面支持 Oracle 数据类型和操作符，包括大对象 BLOB、CLOB 等；
- 支持 Oracle 的各种分区表类型和语法；
- 支持 connect by，merge into，pivot，同义词，hint、rowid、rownum 等各种 Oracle 语法；
- 支持各种 Oracle 系统内置包，包括 DBMS_SQL、DBMS_OUTPU 等等；
- 支持自治事务；
- 支持 dblink 等。

十一、DBbridge 具备数据库国产化迁移完整方案

6 月份，DBbridge 1.0 版本发布，支持 Oracle 到 tencentDB 的评估、结构迁移、全量迁移等功能，提供一站式数据迁移平台以及专家服务，满足企业数据库国产迁移需求。

12 月份，DBbridge 2.5.11 版本发布，支持 Oracle 到 tencentDB 的增量同步，反向同步和数据对比等功能，实现了完整的数据库国产迁移解决方案，满足企业数据的迁移、同步及灾备需求。

十二、腾讯云数据库智能管家 DBbrain 功能更新总结

5 月，腾讯云数据库智能管家 DBbrain 升级数据库健康得分体系，结合 AI，更贴合用户数据库真实运行状况。同时推出了业内首款基于规则和代价估算的 SQL 优化效果预测及对比引擎，不仅使优化建议有量化的标准，能够更为精准，还能够在未执行变更前对变更效果进行预估，让用户能预知变更的优化效果，根据优化建议进行变更。

7 月，腾讯云数据库智能管家 DBbrain 发布了监控大盘、性能趋势、数据库帐号鉴权及常用运维命令快捷执行、数据库帐号安全扫描、异常告警消息通知推送、健康报告自定义邮件推送等功能，重点发布“SQL 限流”和“热点数据防护”两大功能，提供了帮助用户可以在数据库端实现切实有效的降级和防护，保障用户核心业务能正常稳定运行。

10 月，腾讯云数据库智能管家 DBbrain 助力国产数据库，完成对 TDSQL、TDSQL-C 的完美适配，将大量传统人工的数据库运维工作智能化，打造国产数据库智能化核心竞争力。

11 月，腾讯云数据库智能管家 DBbrain 正式对外发布数据库安全特性，包括合规审计、数据脱敏、敏感数据发现、安全治理等功能，可挖掘数据库运行过程中的各类潜在风险和隐患，提升用户数据安全治理的能力，能够为生产数据在分发、复制时提供多元化的保护手段，为企业数据库安全保驾护航，助力用户通过等保合规测，获公安部专用产品认证。

京东智联云数据库 2020 年度发布汇总

- 发布了分析型云数据库 JCHDB，查询性能比传统开源数据库快上千倍，数据分析可实时得到结果，可满足业务对实时数据分析的需求。
- 分布式数据库 TiDB 可支持 TiFlash，可在同一数据库中同时支持交易和数据分析，简化了业务架构，提升了业务性能。
- 云数据库 MySQL 提供了读写代理和只读代理功能，不但可以自动进行读写分离，并且只读实例也可具有高可用性；支持小版本升级，无感知迁移到最新版本。
- 云数据库 PostgreSQL 发布 12.2 版本，支持 SQL/JSON 路径，支持在线重建索引，分区表性能大幅度提升；发布 s3_fdw 数据流转插件，支持 PostgreSQL 到对象存储 OSS 数据流转，扩展 PG 数据库存储能力。
- 云数据库 MongoDB 提供多种节点副本集，可提供更高的数据读取性能；分片集群支持横向扩容，可按需升级集群的性能与容量。
- 数据仓库 JDW 发布 6 版本，OLTP 性能比 5 版本提升 70 倍以上；支持小版本升级，无感知迁移到最新版本。
- DTS 发布数据订阅服务，可实时获取数据库的增量数据；发布数据同步服务，可用于数据异地灾备、业务系统数据流转等场景。

RadonDB 2020 年度重大更新及技术要点分析

MyNewSQL 领域的 RadonDB 云数据库，是一款基于 MySQL 的新一代分布式关系型数据库（MyNewSQL），基于 Go 语言研发。向用户提供具备金融级高可用、强一致、超大容量的数据库服务，高度兼容 MySQL 语法，自动水平分表，智能化扩容。

值得关注的新功能有：

1、一键数据均衡功能： radon rebalance

在用户业务在分布式数据库上运行一段时间后，在大量数据的写入下，通过监控会发现不同后端数据有多有少，随数据量的增加，有的差别会比较大，从而导致数据量很大的节点负载会比较重。基于这种场景，我们新增加了数据均衡功能：radon rebalance。这个命令旨在重新平衡后端之间的数据（分区表）。

详细信息可以参考：
https://github.com/radondb/radon/blob/master/docs/sql_statements/radon_administration_statements.md#radon-rebalance

2、支持白名单地址段功能

针对有些用户需要支持一些连续地址片段的白名单，比如 10.0.0.0/24，配置可以根据正则表达式进行配置。详细信息可以参考：

https://github.com/radondb/radon/blob/master/docs/api.md#config

3、增加 xa 事务的管理功能

针对一些异常场景下两阶段事务中有些 prepared 事务没有最终提交，且后端节点可能比较多，需要提供统一管理接口可以查看整体集群中事务相关状态，并能及时处理异常事务。详细信息可以参考：

https://github.com/radondb/radon/issues/602

4、进一步完善读写分离功能，完善审计相关功能，支持后端指定分片数，丰富和完善集成测试

5、各种生产环境和场景下发现的问题的 fix 和完善，如并发极端情况的死锁问题

6、把迁移工具移到 RadonDB 中，成为功能子集，完善接口：

整合 shift 迁移功能，作为基础库，抽象出 interface 接口，供 radon 层 shift manager 模块调用。

radon 层设计 Shift manager 插件模块，在 plugin 目录下新增 shiftmanager 目录，封装一组对 shift 启动/等待迁移结束/终止迁移等基本操作接口，并提供获取相应对状态信息的接口。并对每一个迁移对实例进行管理。维护每个实例的迁移状态、迁移进度、成功失败等信息，简化设计，降低了代码耦合及后续新增迁移类型功能的开发和维护的难度。

7、语法兼容 MySQL 8.0 工作

举例:完善 insert/replace 语句语法解析(insert/replace 语法基本一致)：

围绕 insert/replace 语句，做了以下重大改进：

- 完善 insert/replace 语句语法解析，当前语法解析同 MySQL 保持一致；
- 去掉 replace 解析 “ON DUPLICATE KEY UPDATE” 选项的功能，这是一个 bug；
- 支持插入表数据时不带列名（在做完语法解析后，获取该表的列信息，写入并改造生成的语法树，给后续执行计划使用）；
- 重构 insert/replace 执行计划，子 sql 生成方式由打印字符串形式统一调整为语法树格式化生成，更为灵活和方便。

当前注意地方：

- 支持解析 INSERT ... SELECT Statement（但子查询功能暂不支持）；
- INSERT ... ON DUPLICATE KEY UPDATE Statement（注意的地方在于分不能更新分区键）；
- 支持解析 INSERT DELAYED Statement(注意：5.7 DELAYED 已经废弃，只解析关键字，不透传)。

以上功能已全部包含在我们的 master 分支：
https://github.com/radondb/radon

推出 dbaplus Newsletter 的想法

dbaplus Newsletter 旨在向广大技术爱好者提供数据库行业的最新技术发展趋势，为社区的技术发展提供一个统一的发声平台。为此，我们策划了 RDBMS、NoSQL、NewSQL、时序数据库、大数据生态圈、国产数据库、云数据库等几个版块。

我们不以商业宣传为目的，不接受任何商业广告宣传，严格审查信息源的可信度和准确性，力争为大家提供一个纯净的技术学习环境，欢迎大家监督指正。

至于 Newsletter 发布的周期，目前计划是约每隔三个月做一次跟进，下期计划时间是 2021 年 4 月 19 日~4 月 30 日，如果有相关的信息提供请发送至邮箱：newsletter@dbaplus.cn

感谢名单

最后要感谢那些提供宝贵信息和建议的专家朋友，排名不分先后。

贡献者单位/职务	贡献者	贡献领域
dbaplus 社群	杨建荣、林林	
甲骨文资深解决方案工程师	阮蓉	Oracle
凡普金科和爱钱进 DBA 团队负责人	贺春旸	MySQL
PostgreSQL 中国社区发起人	德哥	PostgreSQL
蚂蚁集团 OceanBase 运营专家	李琦	OceanBase
CRUG 主席	张冬洪	Redis
前饿了么框架工具部监控平台负责人	黄杰	RocksDB
阿里云产品经理	刘军民	Cassandra
PingCAP 联合创始人兼 CTO	黄东旭	TiDB
巨杉数据库市场部	史新龙	SequoiaDB
Elastic Stack 国内顶尖实战专家	李猛	Elastic
项目经理	段旻	Greenplum
光大银行核心团队数据库专家	洪烨	openGauss
极数云舟 ArkFAE 负责人	许子文	ArkDB
易鲸捷市场经理	秦浩越	QianBase™
阿里云数据库品牌负责人	李京	阿里云数据库
腾讯云数据库高级品牌经理	周璐璐	腾讯云数据库
京东云资深产品经理	杨牧	京东智联云数据库
青云数据库团队核心研发	Andy	RadonDB

欢迎提供 Newsletter 信息，发送至邮箱：newsletter@dbaplus.cn
Github 地址：https://github.com/dbaplus/DBAplus_Newsletter
欢迎技术文章投稿，发送至邮箱：editor@dbaplus.cn


扫码关注