2018PostgreSQL中国技术大会

Your Topic Name

董红禹

donghongyu@cstech.ltd

杭州乘数科技有限公司

背景

- 高额的license费用
  - 每年将近几十万licens费用，按CPU收费并做了硬限制
- 特定业务场景不能满足
  - 特定业务场景性能不能满足，扩展性不友好，与其他平台不能很好融合
- 开源自主可用
  - 随着发展企业将向开源自主可控转型

迁移工作流程

调研
- 数据量、差异、存储过程、函数、视图

迁移方式
- 工具调研

逐步迁移
- 部分业务、odbc_fdw

正式割接
- 全部业务切换完成

数据类型对比

SQL Server	PostgreSQL
VARCHAR(N)	TEXT/VARCHAR
DATETIME	TIMESTAMP CURRENT_TIMESTAMP(0)::TIMESTAMP CLOCK_TIMESTAMP()::TIMESTAMP
MONEY	NUMERIC(15,4)
IMAGE	BYTEA
UNIQUEIDENTIFIER	UUID
VARBINARY	BYTEA
TIMESTAMP	INTERVAL

函数类型转换

SQL Server	PostgreSQL
ISNULL()	COALESCE()
DATALENGTH	OCTET_LENGTH()
LEN()	LENGTH()
CONVERT()	::TYPE / (CAST())
SUBSTRING('DFADF', 1, 2)[DF]	SUBSTR()
CHARINDEX(';', 'FASD;FDS', 2)	POSITION(';' IN 'FASD;FDS')
GETDATE()	NOW();
DATEADD	NOW()+INTERVAL '3 MONTHS';
DATEDIFF(DAY, STARTTIME, ENDTIME)	DATE_PART('DAY', ENDTIME - STARTTIME))
DATEPART()	DATE_PART('WEEK', NOW());

临时表

- SQL Server中临时表
  - CREATE TABLE #Temp (id int, customer_name nvarchar(50), age int)
  - select * into #t12 from table01;

- PG中使用方式
  - create temp table tmp_t12 on commit drop as select * from table01;
    - on commit
      - PRESERVE ROWS
      - DELETE ROWS
      - DROP

视图、外键

- 视图
  - 查询方式相同，需要注意PG中基表的数据类型发生变化后，视图需要重建

```
begin;  -- 开始事务
set local lock_timeout = '1s';  -- 设置锁超时
drop view v_test;  -- 删除依赖视图
alter table test alter column a type varchar(32);  -- 修改字段长度
create view v_test as select id,c1 from test;  -- 创建视图
end;  -- 结束事务
```

- 外键
  - SQL Server中外键可以临时禁用
  - PG中在创建表时设置是否可延迟约束
    - DEFERRABLE
      - INITIALLY DEFERRED
      - INITIALLY IMMEDIATE

索引

- 索引
  - PostgreSQL中没有聚集索引
  - 对于选择性底的索引可以创建条件索引
    - CREATE INDEX IDX_Job_CompanyId ON Job (CompanyId) WHERE IsDeleted = false;

部分索引

```
postgres=# create table t5(a int,name character varying);
postgres=# insert into t5  select 1,'test'||i from generate_series(1,100000) as t(i);
postgres=# insert into t5  select i,'test'||i from generate_series(1,1000) as t(i);
postgres=# explain select * from t5 where a=1;
                            QUERY PLAN
-------------------------------------------------------------------------
 Seq Scan on t5  (cost=10000000000.00..10000001808.50 rows=100027 width=13)
   Filter: (a = 1)

postgres=# create index idx_a_t5 on t5(a) where a<>1;
postgres=# explain select * from t5 where a=100;
                          QUERY PLAN
---------------------------------------------------------------
 Index Scan using idx_a_t5 on t5  (cost=0.28..8.33 rows=3 width=13)
   Index Cond: (a = 100)
(2 rows)
```

非索引列的使用

```
postgres=# explain  select * from t5 where a=1 and name='test1';
                          QUERY PLAN
---------------------------------------------------------------------------
 Seq Scan on t5  (cost=10000000000.00..10000002061.00 rows=1 width=13)
   Filter: ((a = 1) AND ((name)::text = 'test1'::text))

postgres=# create index idx_a_name_t5 on t5(a) where name='test100';

postgres=# explain  select * from t5 where a=1 and name='test100';
                          QUERY PLAN
---------------------------------------------------------------------------
 Index Scan using idx_a_name_t5 on t5  (cost=0.13..8.14 rows=1 width=13)
   Index Cond: (a = 1)
```

表达式索引

```
postgres=# explain select * from t5 where a+1=100;
                          QUERY PLAN
---------------------------------------------------------------------
 Seq Scan on t5  (cost=10000000000.00..10000002061.00 rows=505 width=13)
   Filter: ((a + 1) = 100)

Time: 0.962 ms
postgres=# create index idx_a_t5 on t5((a+1));
CREATE INDEX
postgres=# explain select * from t5 where a+1=100;
                          QUERY PLAN
---------------------------------------------------------------------
 Bitmap Heap Scan on t5  (cost=12.21..577.48 rows=505 width=13)
   Recheck Cond: ((a + 1) = 100)
   ->  Bitmap Index Scan on idx_a_t5  (cost=0.00..12.08 rows=505 width=0)
         Index Cond: ((a + 1) = 100)
```

应用SQL修改

- SQL相关
  - 字符串拼接
    - SELECT FirstName + LastName FROM..
    - SELECT FirstName || LastName FROM...
  - 大小写敏感问题
    - SQL Server中不区分大小写
    - PG中可以使用LOWER/UPPER函数
- Order by
  - SQL Server中Order by 首先会选择NULL值
  - PG中可以选择先读取NULL值还是后读取NULL值
    - select * from t order by name NULLS last;
    - select * from t order by name NULLS FIRST;

UUID

- SQL Server中
  - 用newid()函数生成uuid的
- PG中使用方式
  - create extension uuid-ossp
    - uuid_generate_v1()生成uuid

存储过程及函数

- 获取受影响行数

```
while 1=1 loop --批量删除数据
        delete from t_inofaout_zcq  where trade_id=in_trade_id
and
in_trade_id>0 and rq> in_start_date and rq<=in_end_date ;
        GET DIAGNOSTICS v_count = ROW_COUNT;
        if v_count<10000 then
                exit;     --while循环中退出
        end if;
end loop;
```

存储过程及函数

- 返回单行多列

```
create or replace function GetDate(
  in in_month int,
  out v_date1 date,
  out v_date2 date
) returns record
as $$ BEGIN
    v_date1 :=now();
    v_date2 :=now()+make_interval(months => in_month);
    return;
END;
$$ LANGUAGE plpgsql;
```

存储过程及函数

- 返回多行多列

```
create or replace function sp_get_multiple_set1(
in f_id int,
refcursor,
refcursor
) returns setof refcursor
as
$tt$
declare r1 alias for $2;
            r2 alias for $3;
begin
  open  r1 for select * from j1 where id =f_id;
  return next r1;
  open r2 for select * from tmp_2 where id = f_id;
  return next r2;
end;
```

存储过程及函数

- 存储过程中调用存储过程
  - SQL Server中是exec
  - PG中转为如果这个函数是返回单值的，就直接调用函数即可
    - v_info := f_getinfo(‘aaaaa’);
  - 如果是返回多行的，用

```
for rec in select * from func_name('xxxx') loop
...
end loop;
```

存储过程及函数

- 不想处理函数返回值用perform关键字

```
CREATE OR REPLACE FUNCTION f_test16(in in_id1 int)
RETURNS int
AS $$
DECLARE
rec record;
BEGIN
    perform f_test15(10);
    return 10;
END;
$$ LANGUAGE plpgsql;
```

迁移工作

- 迁移工具
  - 调研迁移工具及项目具体情况
    - 最后选择自己使用python程序开发
- 存储过程修改
  - 1500个存储过程迁移
- 演练测试
  - 多次演练测试评估迁移时间窗口

保持原有业务访问方式

- 原有业务通过函数方式访问数据库
  - 同时考虑读写分离
    - 使用plproxy创建读写集群

```
CREATE SERVER write_cluster FOREIGN DATA WRAPPER plproxy options
(connection_lifetime '1800',
p0 'dbname=db0 hostaddr=172.16.3.150 port=5432');
```

```
CREATE SERVER read_cluster FOREIGN DATA WRAPPER plproxy options
(connection_lifetime '1800',
p0 'dbname=db0 hostaddr=172.16.3.151 port=5432',
p1 'dbname=db1 hostaddr=172.16.3.152 port=5432',
p2 'dbname=db2 hostaddr=172.16.3.153 port=5432',
p3 'dbname=db3 hostaddr=172.16.3.151 port=5432');
```

主库

保持原有业务访问方式

- 原有业务通过函数方式访问数据库
  - 同时考虑读写分离
    - 创建函数时根据读写操作指定对应路由的cluster

```
#代理proxydb上插数据的函数:
CREATE OR REPLACE FUNCTION insert_user(i_username text, i_emailaddress text)
RETURNS integer AS $$
    CLUSTER 'write_cluster';
    RUN ON ANY;
$$ LANGUAGE plproxy;
#代理proxydb上查询数据的函数
CREATE OR REPLACE FUNCTION get_user_email(i_username text) RETURNS SETOF text AS $$
    CLUSTER 'read_cluster';
    RUN ON ANY;
$$ LANGUAGE plproxy;
```

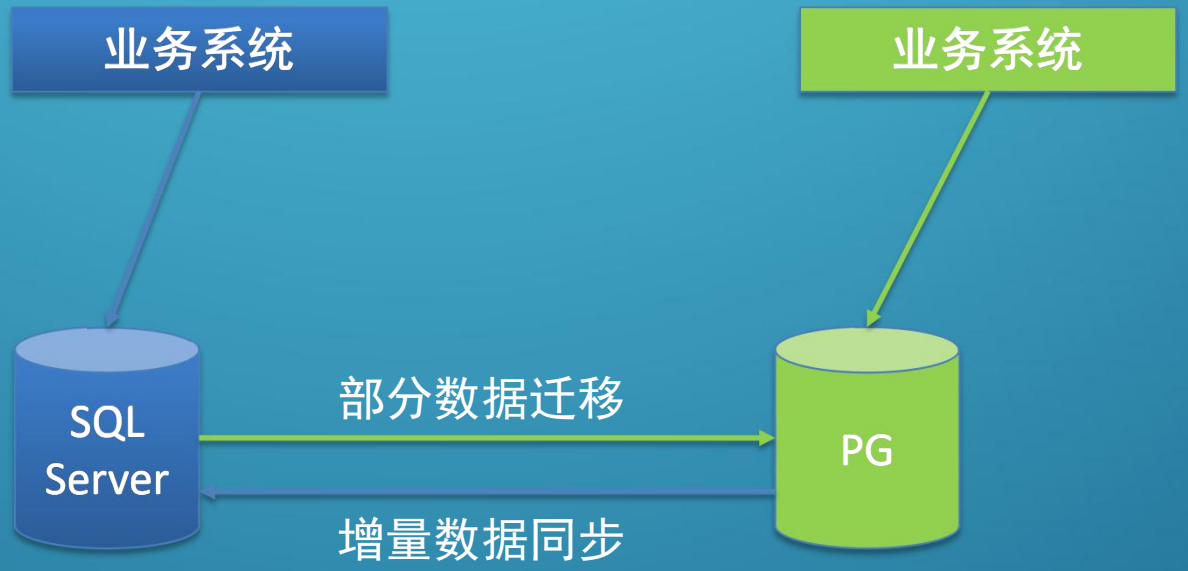
保持原有业务访问方式

- 原有业务通过函数方式访问数据库
  - 同时考虑读写分离
    - 调用函数时可直接路由完成读写分离

```
CREATE or REPLACE FUNCTION sumtest(
    username text,
    out out_result INT,
    out em text,
    out im INT
)returns record
as $$
BEGIN
    BEGIN
        em :=get_user_email(username);
        im :=insert_user('xiaoming', em);
        out_result:=1;
EXCEPTION
```

过度迁移

- 迁移部分数据
  - PG与SQL Server同时使用
  - 使用ODBC_FDW做数据同步
    - https://github.com/hangzhou-cstech/odbc_fdw



Thanks