PostgreSQL 11,12 重大新特性

阿里云

digoal

PG 11

- 分区表增强
- 并行计算增强
- btree index include索引叶子附加属性
- 添加字段（含默认值）更快
- 支持存储过程

分区表增强

- hash分区
- 支持触发器
- 支持默认分区
- 允许修改分区字段

并行计算增强

- https://github.com/digoal/blog/blob/master/201903/20190318_05.md

- 36个CASE, 平均20倍提升
- create table|mview as,create index
- hash join, hash agg, parallel multi-phase agg

PostgreSQL 11 并行计算使用场景、性能提升倍数

场景	数据量	关闭并行	开启并行	并行度	开启并行性能提升倍数
全表扫描	10亿	53.4秒	1.8秒	32	29.7倍
条件过滤	10亿	53.4秒	1.87秒	32	28.6倍
哈希聚合	10亿	142.3秒	4.8秒	30	29.6倍
分组聚合	10亿	142.3秒	4.8秒	30	29.6倍
select into	10亿	54.5秒	1.9秒	32	28.7倍
create table as	10亿	54.7秒	2秒	30	27.35倍
CREATE MATERIALIZED VIEW	10亿	54.7秒	2秒	30	27.35倍

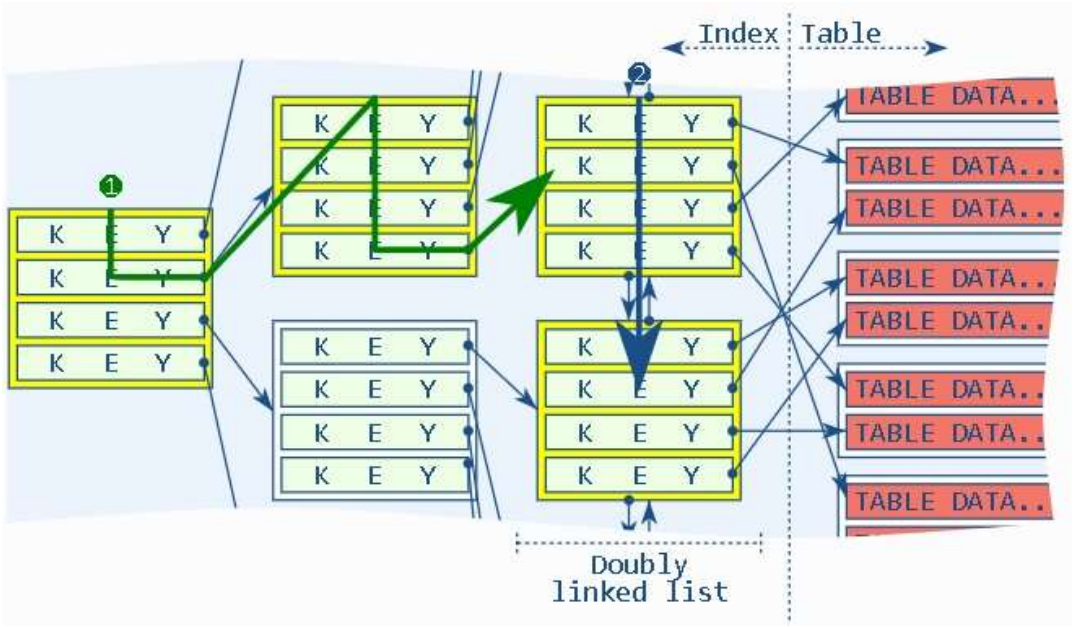
create index	10 亿	964 秒	252 秒	32	3.83 倍
parallel CREATE INDEX CONCURRENTLY - 不堵塞读写	10亿	509.6 秒	355 秒	16	1.44 倍
排序	10 亿	76.9 秒	2.75 秒	32	28 倍
自定义并行聚合1(求 distinct 数组 字段元素、以及count distinct)	10 亿	298.8 秒	8.7 秒	36	34.3 倍
自定义并行聚合2(求 distinct 普通 字段元素、以及count distinct)	10 亿	96.5 秒	3.43 秒	36	28 倍
自定义并行函数(UDF)	10 亿	456 秒	16.5 秒	30	27.6 倍
普通并行(gather)	10 亿	70.2 秒	2.5 秒	30	28.1 倍
归并并行(gather merge)	10 亿	78.2 秒	2.76 秒	30	28.3 倍
rc (ud agg count distinct)	10 亿	107 秒	3.65 秒	30	29.3 倍
rr (ud agg count distinct)	10 亿	107 秒	3.65 秒	30	29.3 倍

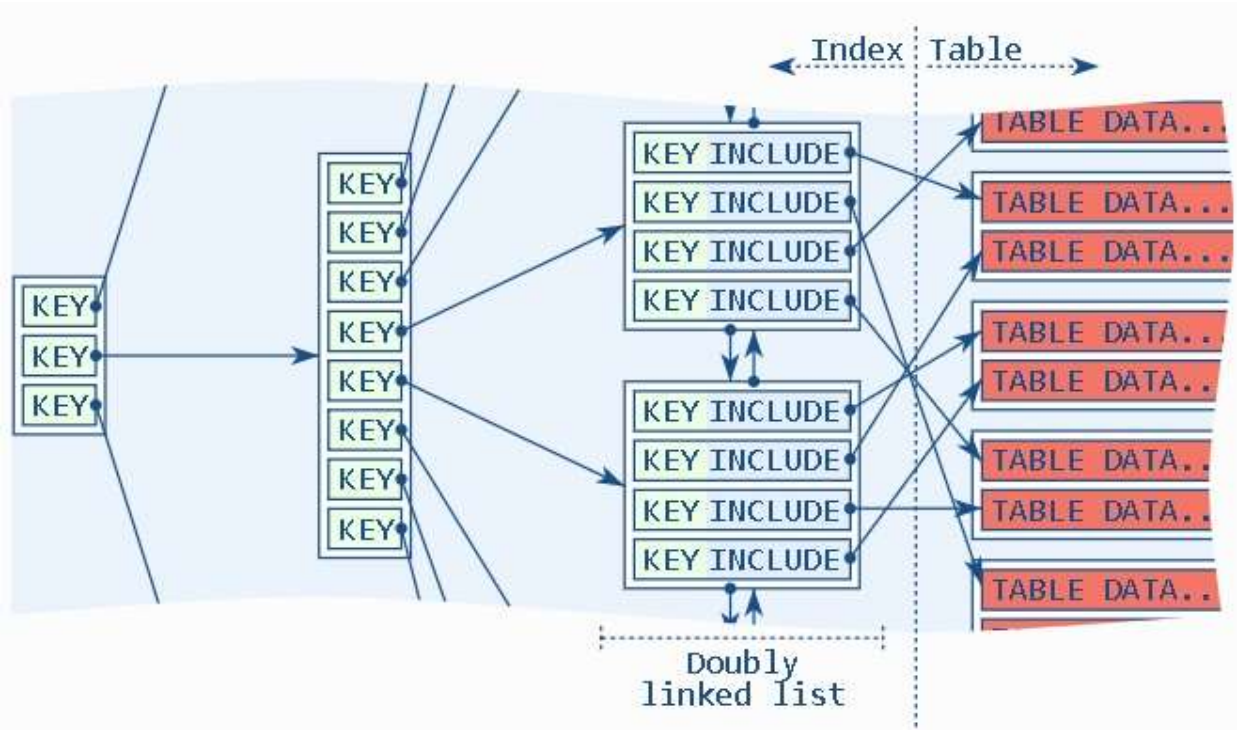
parallel OLAP : 中间结果 parallel with unlogged table ; unlogged table并行求avg case	10 亿	73.6 秒	2.5 秒	30	29.44 倍
parallel index scan	10 亿	19 秒	1.58 秒	20	12 倍
parallel bitmap scan	10 亿	23.98 秒	15.86 秒	20	1.5 倍
parallel index only scan	10 亿	8 秒	0.6 秒	20	13.33 倍
parallel nestloop join	10亿 join 10亿 using (i) where t1.i<10000000	14.4 秒	4.6 秒	8	3.13 倍
parallel merge join	10亿 join 10亿 using (i) where t1.i<10000000	3.2 秒	1 秒	8	3.2 倍
parallel hash join	10亿 join 10亿 using (i) where t1.i<10000000 and t2.i<10000000	8.1 秒	1 秒	20	8.1 倍
parallel hash join	10亿 join 10亿 using (i)	1071 秒	92.3 秒	20	11.6 倍
parallel partition table wise join	10亿 join 10亿 using (i)	1006 秒	76 秒	24	13.2 倍
parallel partition table wise agg	10亿	191 秒	8 秒	24	23.9 倍

parallel append	10亿	70.5 秒	3.16 秒	24	22.3 倍
parallel append merge	10亿	99.4 秒	5.87 秒	24	16.93 倍
parallel union all	10亿	99 秒	5.6 秒	24	17.68 倍
parallel CTE	10亿	65.65 秒	3.33 秒	24	19.7 倍
parallel 递归查询, 树状查询, 异构查询, CTE, recursive CTE, connect by	异构数据1亿，日志数据10亿	5.14 秒	0.29 秒	24	17.7 倍
parallel scan mult FDW tables (通过继承表方式)	10亿	180 秒	7.8 秒	24	23.1 倍
parallel scan mult FDW tables (通过union all)	10亿	165.6 秒	27.8 秒	5	6 倍
parallel leader process	10亿	186 秒	95 秒	1	2 倍
parallel subquery	20亿	179.7 秒	6.5 秒	28	27.6 倍

btree index include索引叶子附加属性

- https://github.com/digoal/blog/blob/master/201905/20190503_03.md
- 类似B+tree数据聚集，
- create index idx_t1_1 on t1 (id) include(c1,c2,c3,info,crt_time);





添加字段（含默认值）更快

- https://github.com/digoal/blog/blob/master/201805/20180518_01.md

```
postgres=# \x
Expanded display is on.
postgres=# select * from pg_attribute where attrelid='aaa'::regclass and attname='c1';
-[ RECORD 1 ]-+---------
attrelid      | 99498
attname       | c1
atttypid      | 25
attstattarget | -1
attlen        | -1
attnum        | 4
attndims      | 0
attcacheoff   | -1
atttypmod     | -1
attbyval      | f
attstorage    | x
attalign      | i
attnotnull    | f
atthasdef     | t
atthasmissing | t
attidentity   |
attisdropped  | f
attislocal    | t
attinhcount   | 0
attcollation  | 100
attacl        |
attoptions    |
attfdwoptions |
attmissingval | {digoal}

Time: 0.470 ms
```

```
postgres=# create unlogged table aaa(id int, info text, crt_time timestamp);
CREATE TABLE
Time: 6.259 ms
postgres=# insert into aaa select generate_series(1,1000000),repeat(md5(random()::text),10), now();
INSERT 0 1000000
Time: 2151.531 ms (00:02.152)
postgres=# insert into aaa select * from aaa;
INSERT 0 1000000
Time: 1235.480 ms (00:01.235)
postgres=# insert into aaa select * from aaa;
INSERT 0 2000000
Time: 2688.409 ms (00:02.688)
postgres=# insert into aaa select * from aaa;
INSERT 0 4000000
Time: 4782.437 ms (00:04.782)
postgres=# insert into aaa select * from aaa;
INSERT 0 8000000
Time: 11367.010 ms (00:11.367)


postgres=# \dt+ aaa
                    List of relations
 Schema | Name | Type  |  Owner   |  Size   | Description
--------+------+-------+----------+---------+-------------
 public | aaa  | table | postgres | 5618 MB |
(1 row)
```

2、新增字段，并添加默认值，由于只需要修改元数据，瞬间完成。

```
postgres=# alter table aaa add column c1 text default 'digoal';
ALTER TABLE
Time: 3.013 ms
```

支持存储过程

- https://www.postgresql.org/docs/11/sql-createprocedure.html

- CREATE [OR REPLACE] PROCEDURE
- name ([[argmode] [argname] argtype [{ DEFAULT | = } default_expr] [, ...]])
- { LANGUAGE lang_name
- | TRANSFORM { FOR TYPE type_name } [, ...]
- | [EXTERNAL] SECURITY INVOKER | [EXTERNAL] SECURITY DEFINER
- | SET configuration_parameter { TO value | = value | FROM CURRENT }
- | AS 'definition'
- | AS 'obj_file', 'link_symbol'
- } ...

- https://github.com/digoal/blog/blob/master/201905/20190531_01.md

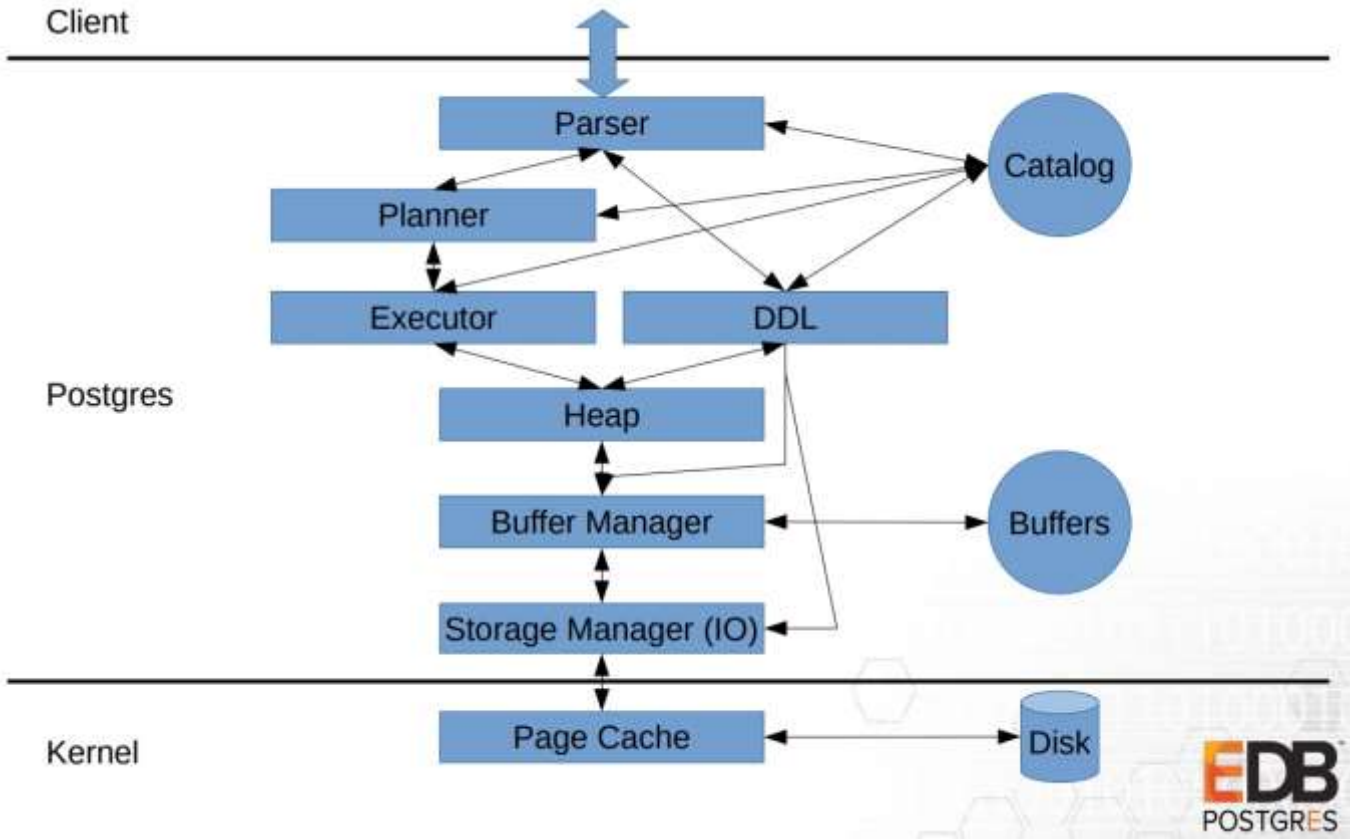
- 支持事务
- commit|rollback

PG 12

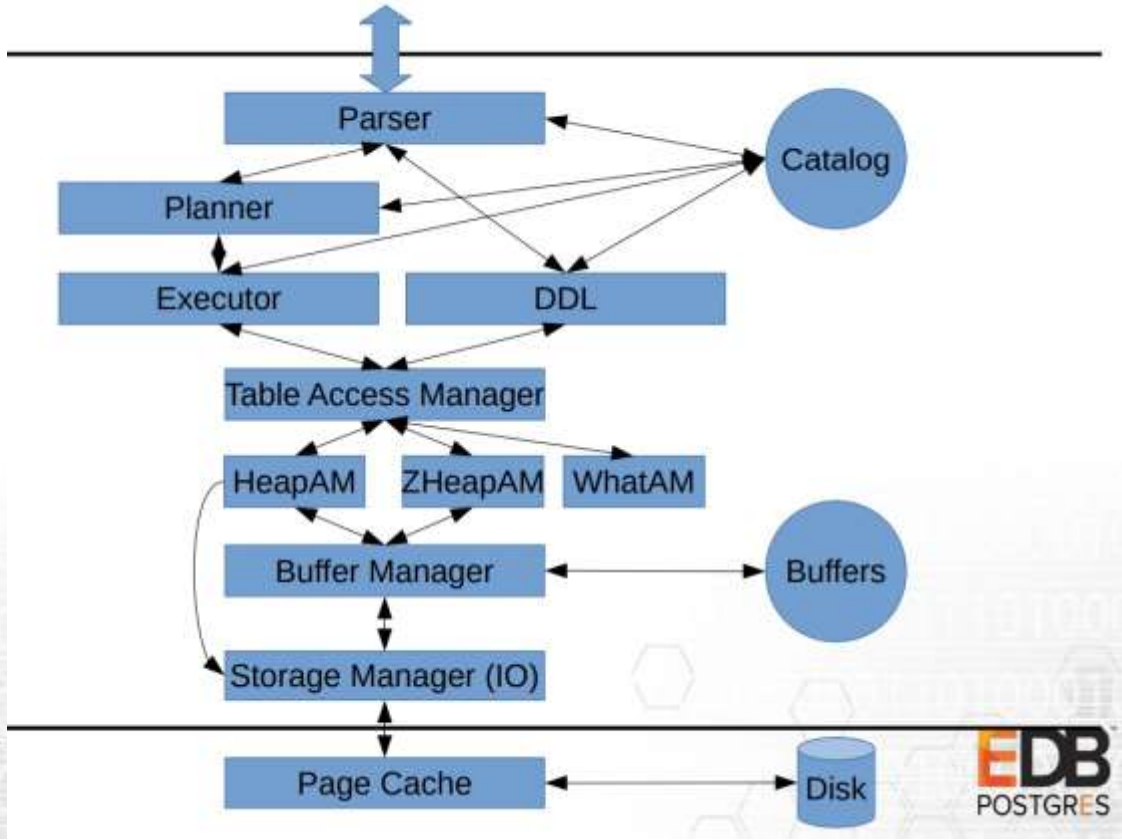
- AM接口
- 分区表-大量分区性能提升
- GiST index include索引叶子附加属性
- CTE 物化、非物化
- 日志采样
- COPY WHERE

AM接口

- https://github.com/digoal/blog/blob/master/201903/20190331_03.md
- https://github.com/digoal/blog/blob/master/201905/20190531_03.md



PG 12



- zedstore(列存)
- zheap(支持回滚段)

```
postgres=# SELECT AVG(i199) FROM (select i199 from layout offset 0) x; -- heap
         avg
---------------------
 500000.500000000000
(1 row)

Time: 4679.026 ms (00:04.679)

postgres=# SELECT AVG(i199) FROM (select i199 from zlayout offset 0) x; -- zedstore
         avg
---------------------
 500000.500000000000
(1 row)

Time: 379.710 ms
```

分区表-大量分区性能提升

- 1000个分区、469倍

- https://github.com/digoal/blog/blob/master/201905/20190521_01.md
- https://github.com/digoal/blog/blob/master/201903/20190331_01.md

case	PG 11	PG 12 beta1	**性能提升倍数**
单表查询 qps	1161671	1160909	持平
单表upsert qps	317379	332552	持平
分区表(1024分区)查询 qps	1163	545602	469倍
分区表(1024分区)upsert qps	2885	246627	85倍

GiST index include索引叶子附加属性

- 轨迹，时空搜索
- 按结果集（索引）聚集存储，消除回表IO放大。

CTE 物化、非物化

- https://github.com/digoal/blog/blob/master/201903/20190309_04.md

•非物化-条件下推

- WITH w AS NOT MATERIALIZED (
- SELECT * FROM big_table
-)
- SELECT * FROM w AS w1 JOIN w AS w2 ON w1.key = w2.ref
- WHERE w2.key = 123;

```
with NOT MATERIALIZED (不使用物化，允许外面条件推进去)

with MATERIALIZED (使用物化)
```

日志采样

- https://github.com/digoal/blog/blob/master/201904/20190405_09.md
- log_statement_sample_rate ， 当设置了log_min_duration_statement时， 如果log_statement_sample_rate也设置了， 它表示百分之多少的超时SQL被记录。

- log_transaction_sample_rate ， 不管其他任何设置， 它表示百分之多少的事务被记录duration。（事务为最小粒度单位， 如果一个事务被触发了记录duration， 这个事务中的所有SQL都会被记录。）

COPY WHERE

- https://github.com/digoal/blog/blob/master/201903/20190331_11.md

```
COPY table_name [ ( column_name [, ...] ) ]
    FROM { 'filename' | PROGRAM 'command' | STDIN }
    [ [ WITH ] ( option [, ...] ) ]
    [ WHERE condition ]
```

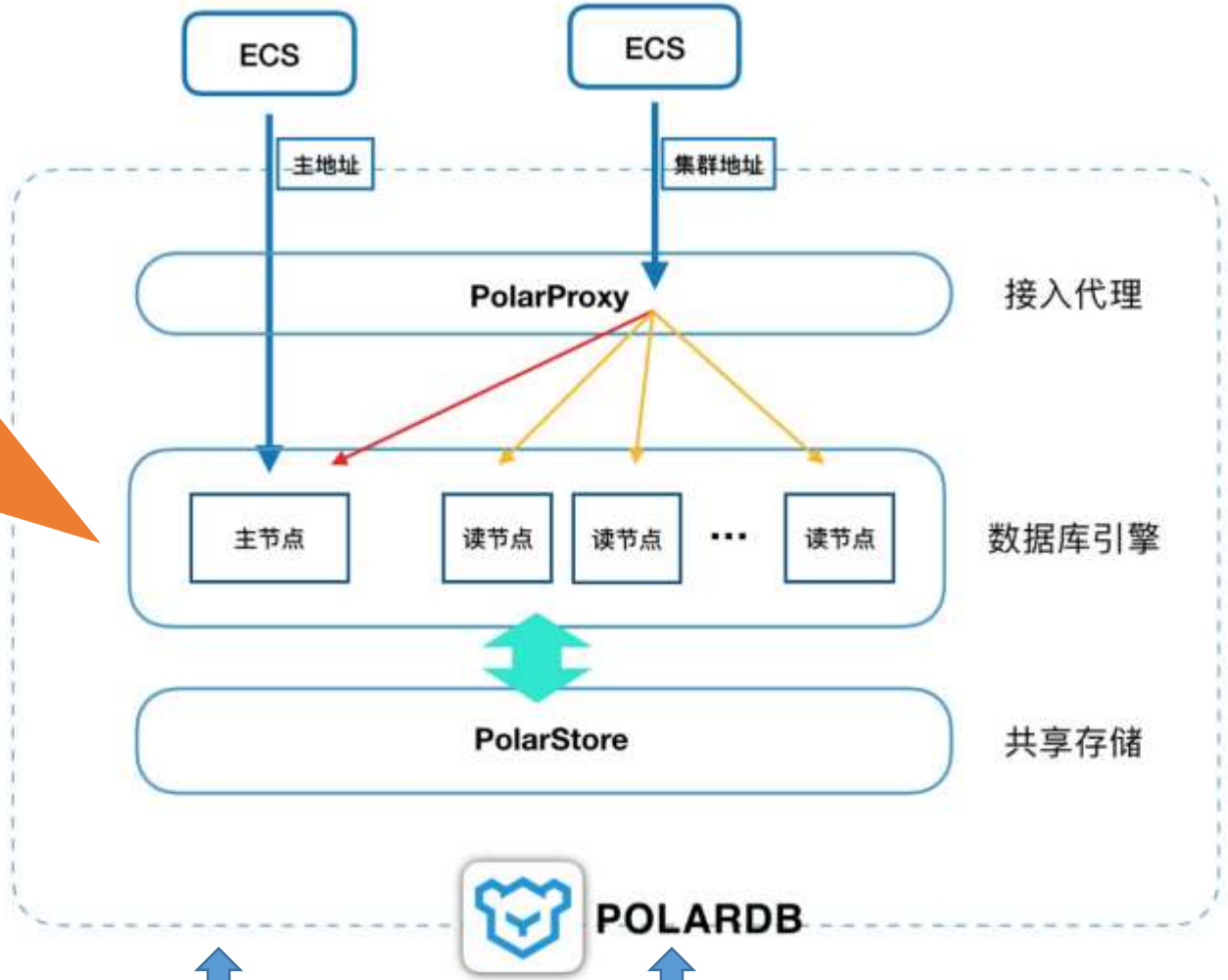
```
postgres=# create table t_to (id int , info text, crt_Time timestamp);
CREATE TABLE
postgres=# insert into t_to select generate_series(1,100000), md5(random()::Text), clock_timestamp();
INSERT 0 100000
postgres=# copy t_to to '/tmp/t_to';
COPY 100000
postgres=# create table t_from (like t_to);
CREATE TABLE
postgres=# copy t_from from '/tmp/t_to' where id<100;
COPY 99
```

阿里云POLARDB v2.0 重磅发布兼容Oracle|PostgreSQL

【Oracle深度兼容】
内置ORACLE兼容(for Oracle)

【OLTP+OLAP混合负载】
内置并行计算
内置会话级资源隔离

【智能驾驶】
内置AAS性能洞察
内置SQL防火墙
内置索引推荐



【冷热分离】
分级存储。
历史数据想存多久就存多久。

阿里云OSS海量对象存储

云生态
无缝对接

ADB

函数计算

MAX Compute

还有很多很多很多牛逼的特性!

- Preview
- https://github.com/digoal/blog/blob/master/README.md

- News
- https://www.postgresql.org/about/news/1894/
- https://www.postgresql.org/about/news/1943/

- Release Notes
- https://www.postgresql.org/docs/11/release-11.html
- https://www.postgresql.org/docs/12/release-12.html

- 功能矩阵
- https://www.postgresql.org/about/featurematrix/

PG技术进阶
每周直播