PostgreSQL Scale
高并发和大数据下的PG实践

阿里云

digoal

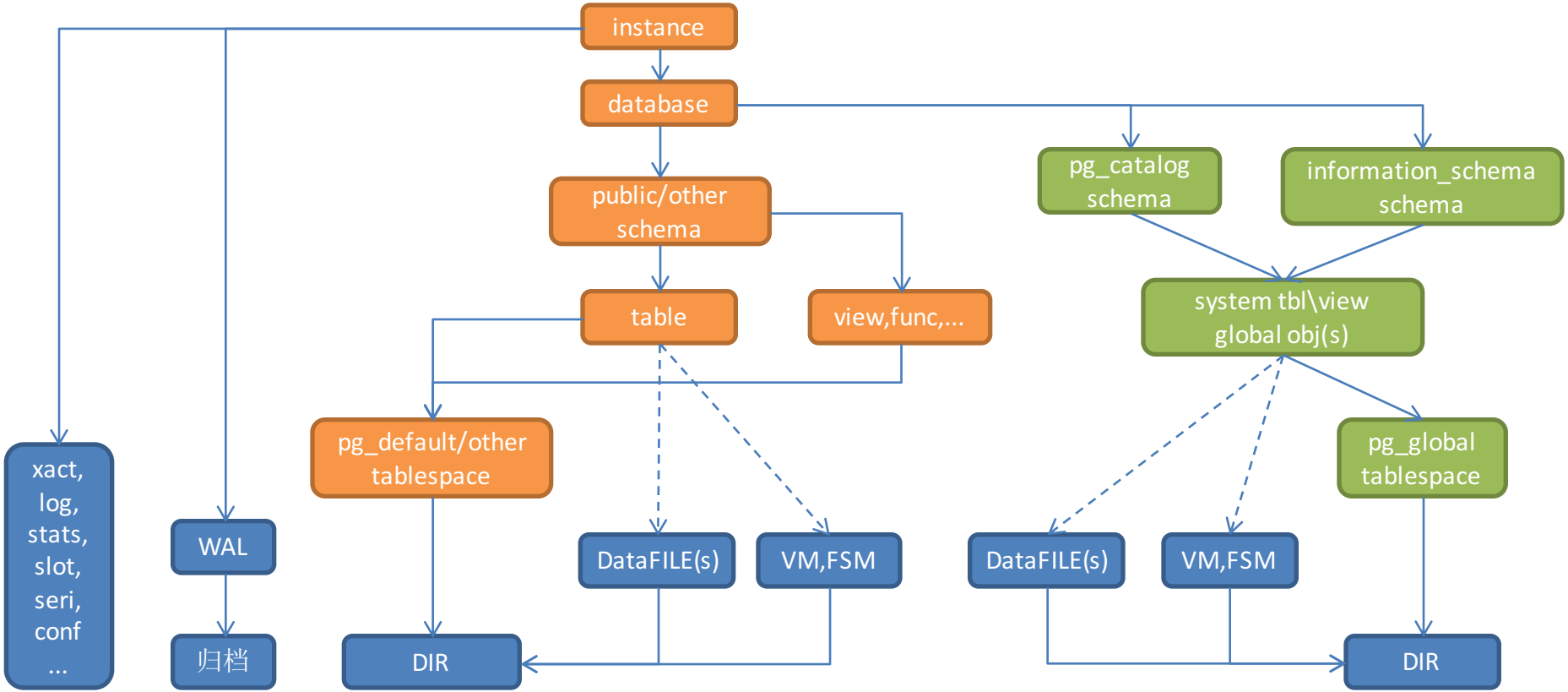
目录

- PostgreSQL数据库架构
- 高并发下的问题与实践
- 大数据下的问题与实践
- 未来的发展方向展望

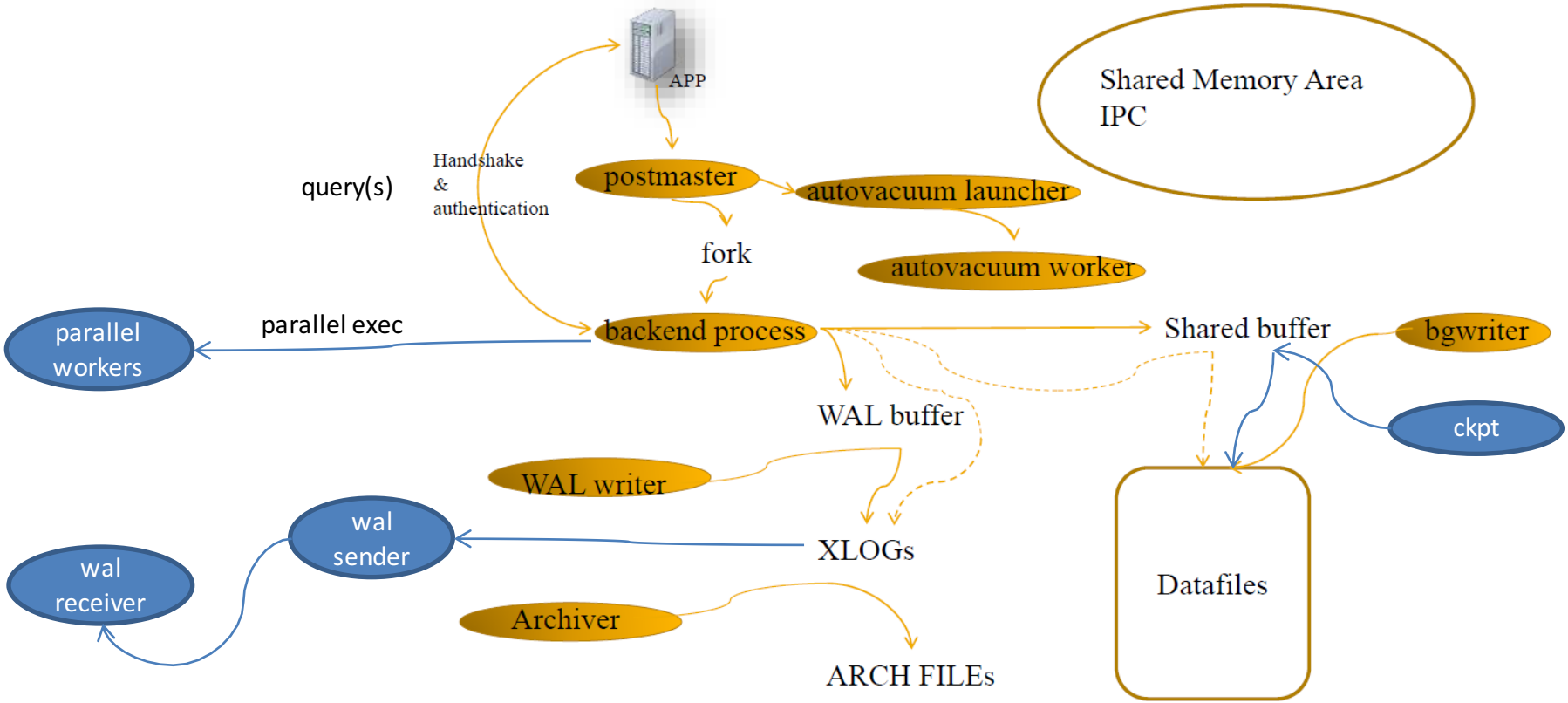
架构

- 物理结构
- 物理流复制
- 逻辑流复制（订阅）
- 单元化
- 多副本
- 读写分离
- sharding

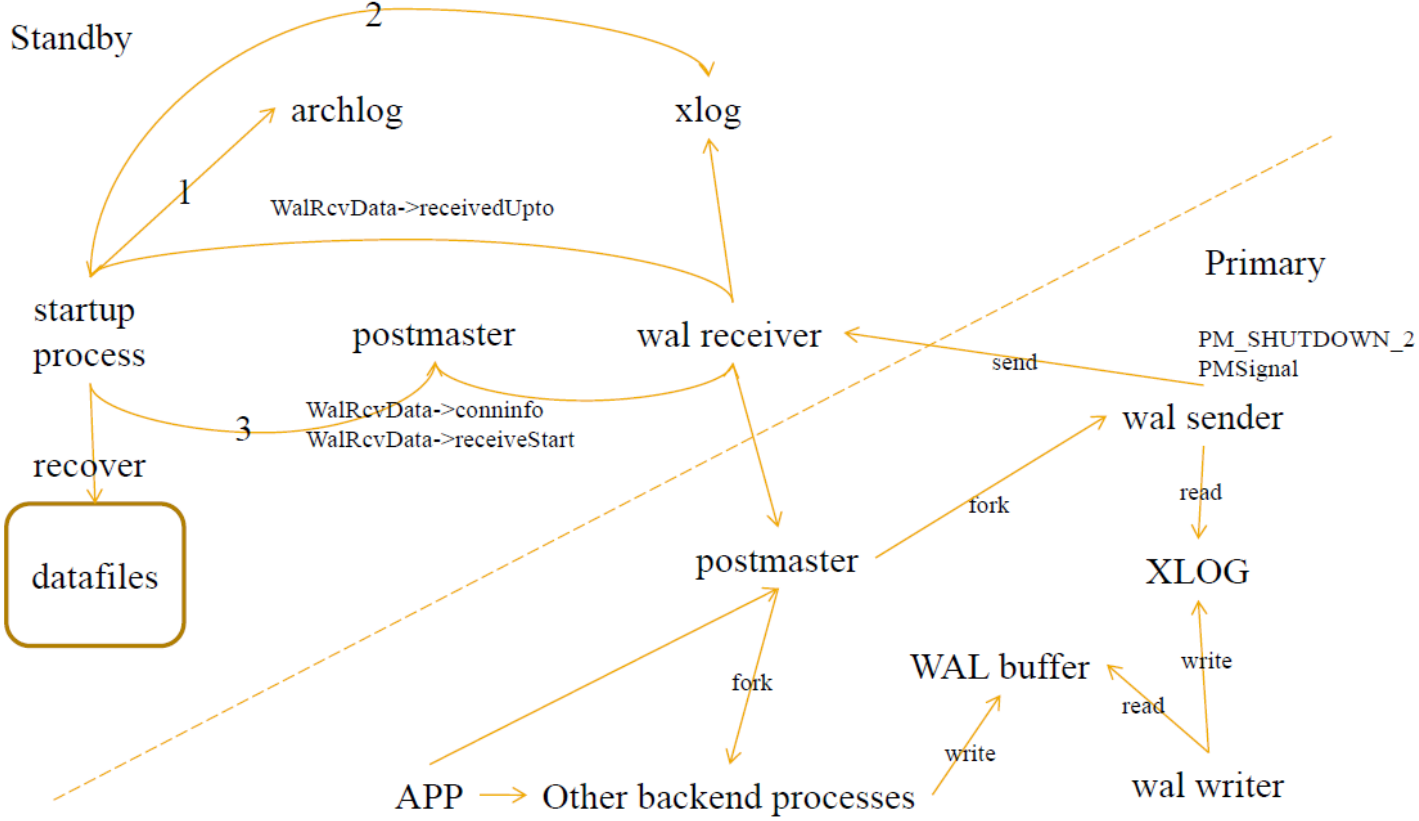
数据库物理架构


instance
database
public/other schema
table
view,func,...
pg_default/other tablespace
DataFILE(s)
VM,FSM
DIR
pg_catalog schema
information_schema schema
system tbl\view global obj(s)
pg_global tablespace
DataFILE(s)
VM,FSM
DIR
xact, log, stats, slot, seri, conf ...
WAL
归档

数据库进程结构

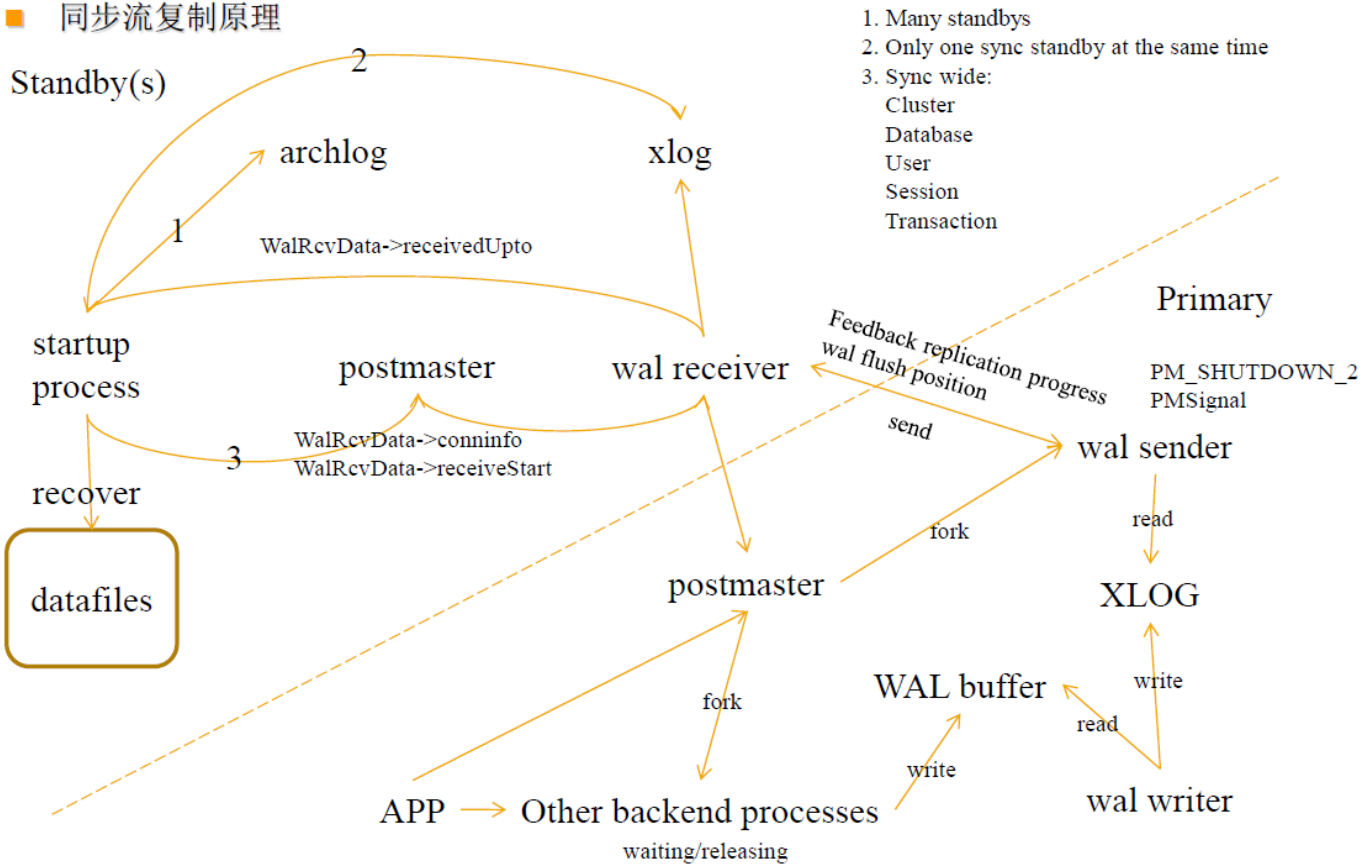

APP
Shared Memory Area
IPC
query(s)
Handshake
&
authentication
postmaster
autovacuum launcher
fork
autovacuum worker
parallel workers
parallel exec
backend process
Shared buffer
bgwriter
WAL buffer
ckpt
WAL writer
wal sender
wal receiver
XLOGs
Datafiles
Archiver
ARCH FILEs

异步流复制


Standby
2
archlog
xlog
1
WalRcvData->receivedUpto
startup
process
postmaster
wal receiver
Primary
PM_SHUTDOWN_2
PMSignal
send
3
WalRcvData->conninfo
WalRcvData->receiveStart
wal sender
recover
fork
read
datafiles
postmaster
XLOG
WAL buffer
write
read
fork
write
APP
Other backend processes
wal writer

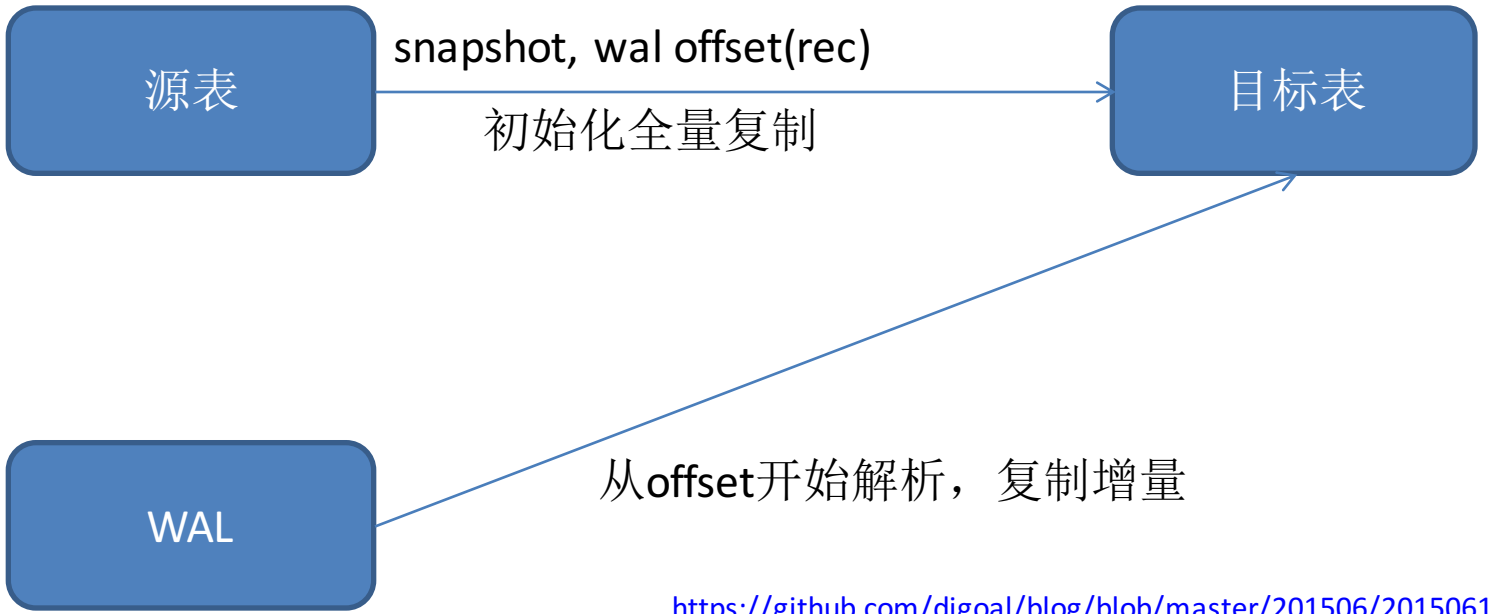
同步流复制

- 同步流复制原理



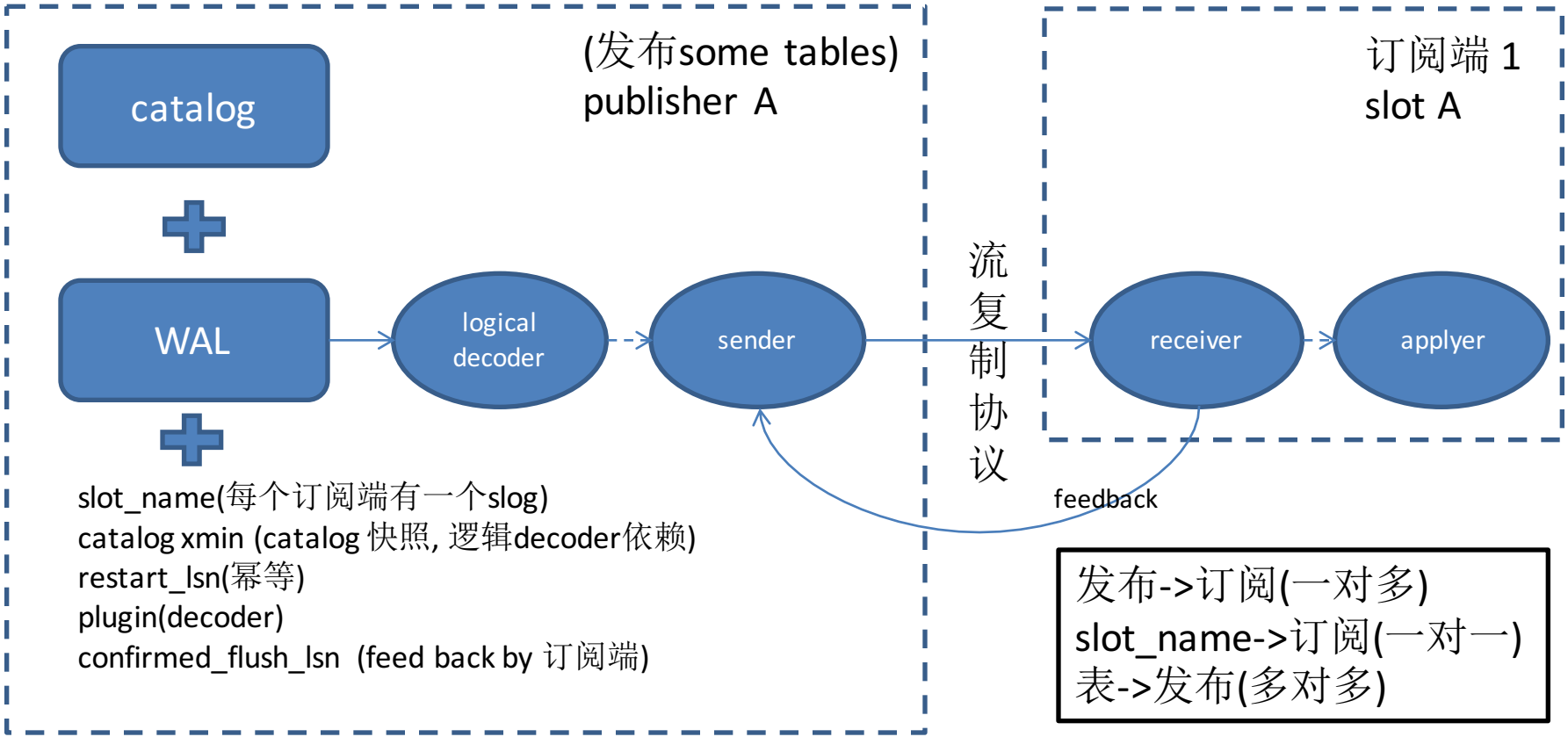
1. Many standbys
2. Only one sync standby at the same time
3. Sync wide:
   - Cluster
   - Database
   - User
   - Session
   - Transaction

逻辑订阅流程



https://github.com/digoal/blog/blob/master/201506/20150616_02.md
(原理：PG逻辑订阅已封装)

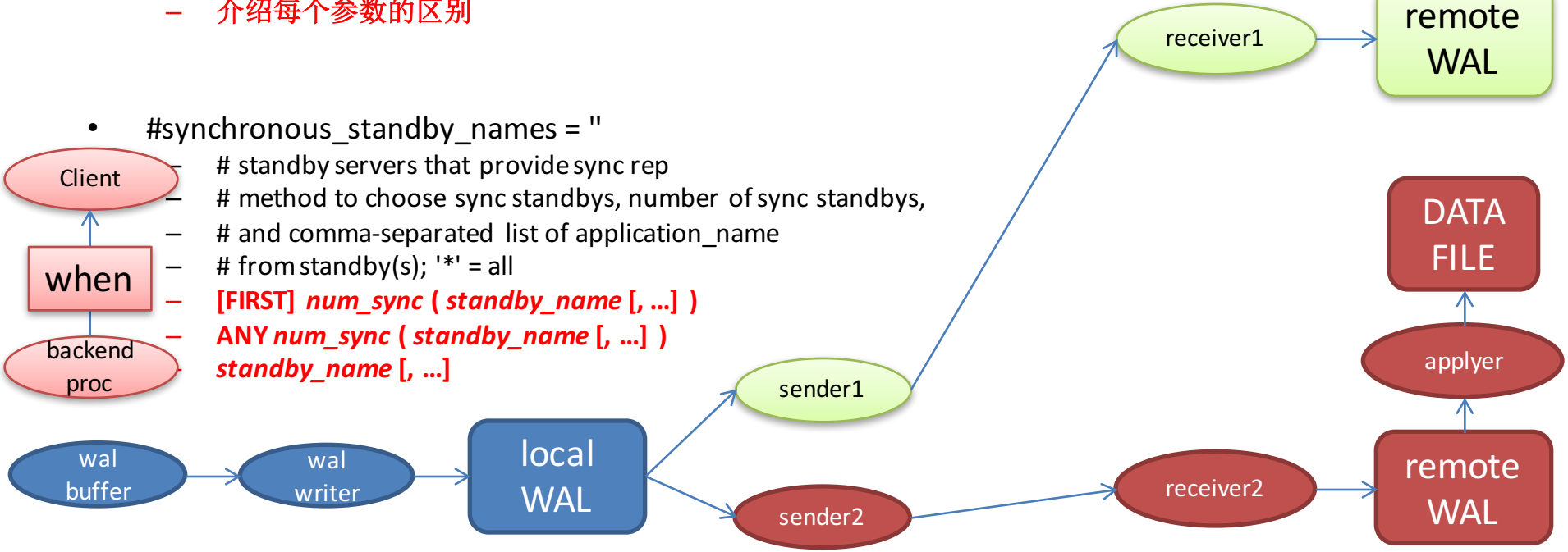
逻辑订阅(增量复制)


(发布some tables)
publisher A
catalog
WAL
logical decoder
sender
流复制协议
receiver
applyer
订阅端 1
slot A
feedback
slot_name(每个订阅端有一个slog)
catalog xmin (catalog 快照, 逻辑decoder依赖)
restart_lsn(幂等)
plugin(decoder)
confirmed_flush_lsn (feed back by 订阅端)
发布->订阅(一对多)
slot_name->订阅(一对一)
表->发布(多对多)

多副本介绍

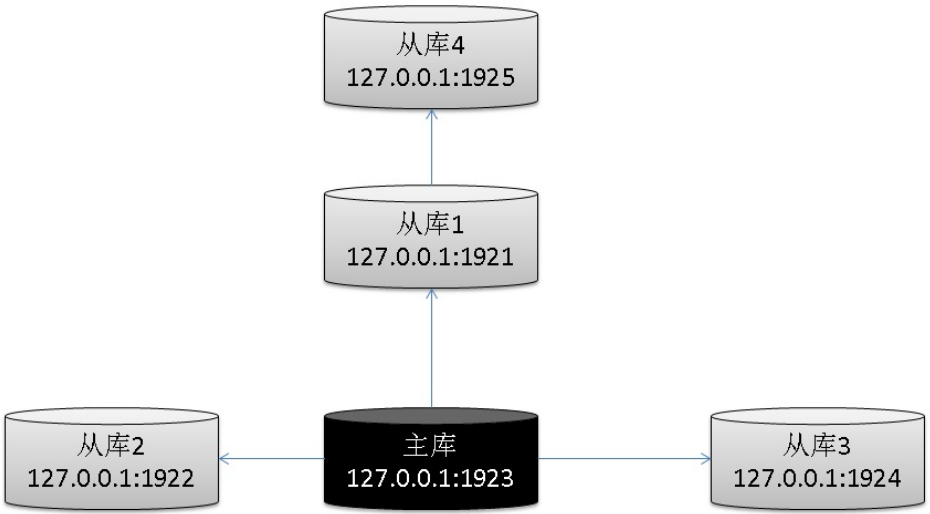
- #synchronous_commit = on
  - # synchronization level;
  - # **off, local, remote_write, remote_apply, or on**
  - **介绍每个参数的区别**

- #synchronous_standby_names = ''
  - # standby servers that provide sync rep
  - # method to choose sync standbys, number of sync standbys,
  - # and comma-separated list of application_name
  - # from standby(s); '*' = all
  - **[FIRST] *num_sync* (*standby_name* [, ...])**
  - **ANY *num_sync* (*standby_name* [, ...])**
  - ***standby_name* [, ...]**



多副本介绍

- https://github.com/digoal/blog/blob/master/201803/20180326_01.md



TPC-B 性能

```
latency average = 1.005 ms
latency stddev = 1.494 ms
tps = 55695.800213 (including
tps = 55700.524316 (excluding
```

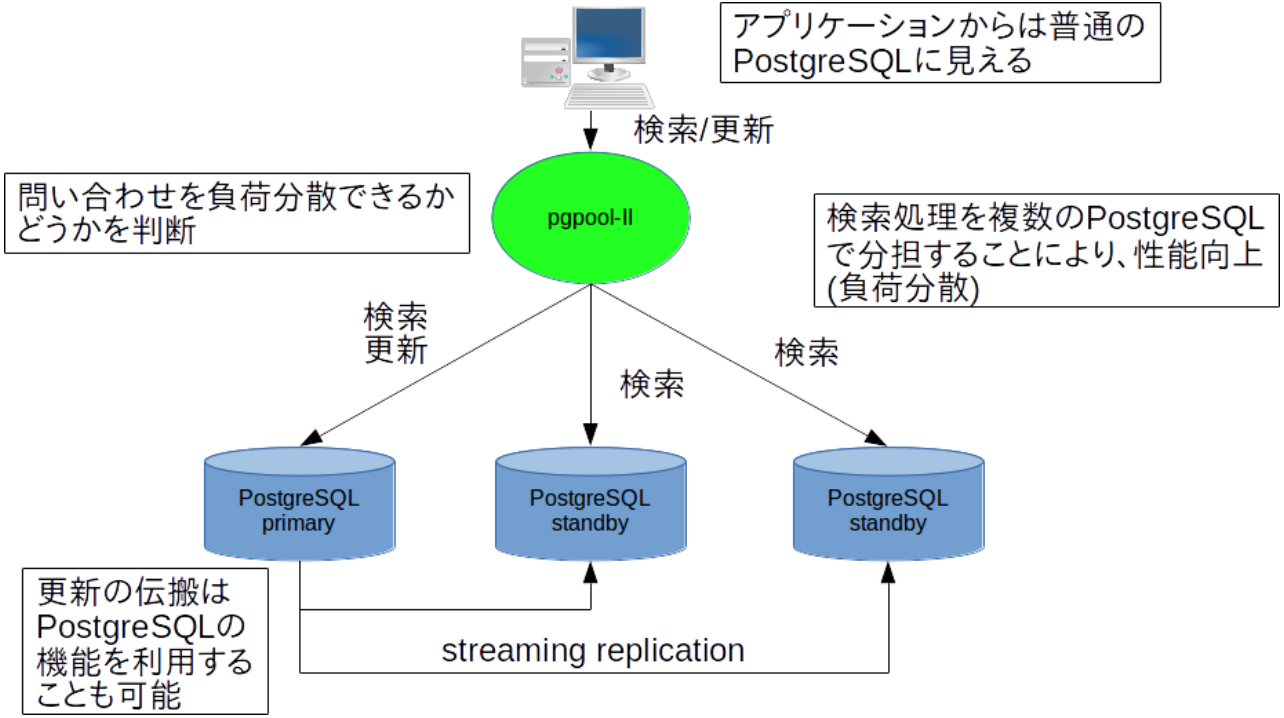
```
latency average = 1.248 ms
latency stddev = 4.118 ms
tps = 44840.272218 (including
tps = 44853.145859 (excluding
```

OLTP写场景本地持久化 VS 多副本性能对比：

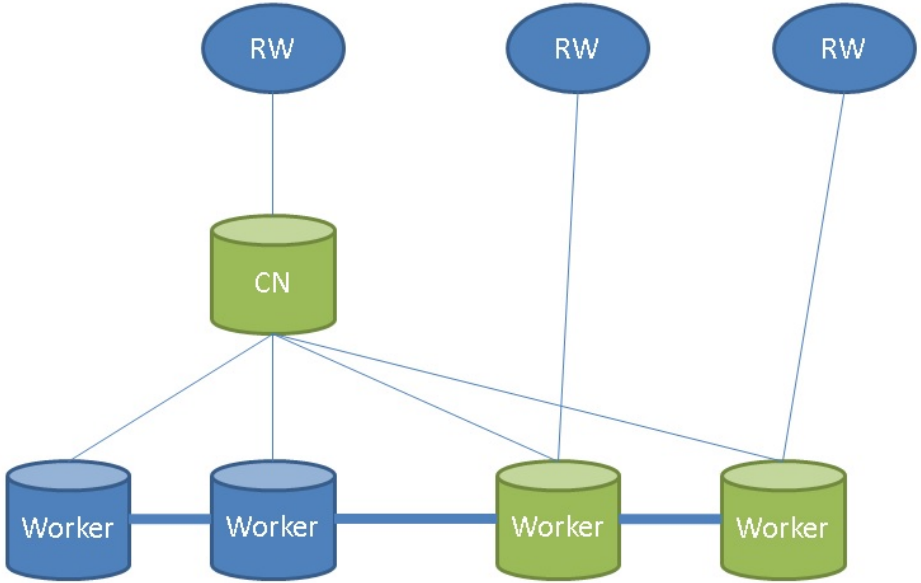
1、多副本模式，RT 提高了0.24毫秒，约20%

2、多副本模式，TPS 下降了10855，约20%

读写分离(pgpool-II)



sharding (citus)



citus
MX特性
指定worker携带metadata

高并发下的问题与实践

- 进程模型开销
  - 内存拷贝、上下文切换、进程间通讯、shared buffer - page table flood
- 锁冲突

- 连接池
  - pgbouncer (事务级连接池)
    - https://github.com/digoal/blog/blob/master/201005/20100511_03.md
- 内置连接池/队列
  - https://github.com/digoal/blog/blob/master/201805/20180521_03.md
  - 阿里云版本
    - https://github.com/digoal/blog/blob/master/201805/20180505_07.md
- 读写分离
  - https://github.com/digoal/blog/blob/master/201608/20160824_03.md
- 大页
  - https://github.com/digoal/blog/blob/master/201803/20180325_02.md

高并发下的问题与实践

- 组提交
  - postgres=# show commit_delay ;
  - commit_delay
  - ------------
  - 10
  - (1 row)

  - postgres=# show commit_siblings ;
  - commit_siblings
  - ----------------
  - 5
  - (1 row)
- 异步提交
  - synchronous_commit = on | off | local | remote_write | remote_apply
- 秒杀
  - https://github.com/digoal/blog/blob/master/201801/20180105_03.md
- perf, oprofile
  - https://github.com/digoal/blog/blob/master/201611/20161129_01.md
  - https://github.com/digoal/blog/blob/master/201505/20150509_01.md

大数据下的问题与实践

- 存储量、计算量

- 存储问题
- 垂直
  - 分区表
    - https://github.com/digoal/blog/blob/master/201805/20180524_05.md
    - https://github.com/digoal/blog/blob/master/201801/20180122_03.md
  - 压缩
    - 类型压缩(变长类型, https://commitfest.postgresql.org/20/1294/)
    - 文件系统压缩(zfs)
  - 冷热存储分离
    - 阿里云oss_fdw
    - https://help.aliyun.com/document_detail/44461.html
- 水平
  - sharding(citus)

大数据下的问题与实践

- 计算问题
- 垂直
  - 列存
  - 向量计算
    - 阿里云 gpdb 内置向量计算加速
    - https://github.com/digoal/blog/blob/master/201702/20170225_01.md
  - CPU并行
    - 9.6开始支持
  - GPU加速
    - 阿里云版本，空间计算内置GPU加速。
- 水平
  - sharding(citus)
  - mpp(gpdb)

TPC-B (ECS 32C, 512G, SSD)

- https://github.com/digoal/blog/blob/master/201809/20180916_01.md
- **1、100亿TPCB初始化**
- 耗时：8385秒。
- 速度：约119万行/s。
- **2、100亿TPCB创建索引（24并行）**
- 耗时：43分50秒。
- 速度：约380万行/s。
- **3、100亿TPCB空间占用**
- 表：1.251 TB
- 索引：209 GB
- **4、100亿TPCB只读3600秒**
- TPS: 118053
- QPS: 118053
- **5、100亿TPCB读写3600秒**
- TPS: 42058
- QPS: 210290

1、10亿TPCB初始化
耗时：879秒。
速度：约113万行/s。
2、10亿TPCB创建索引（24并行）
耗时：145秒。
速度：约690万行/s。
3、10亿TPCB空间占用
表：125 GB
索引：21 GB
4、10亿TPCB只读3600秒
TPS: 109万
QPS: 109万
5、10亿TPCB读写3600秒
TPS: 93729
QPS: 468645

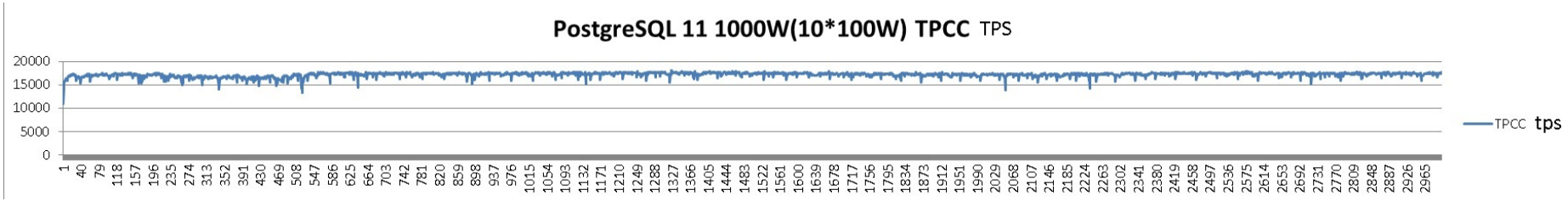
TPC-B (ECS 32C, 512G, ESSD)

- https://github.com/digoal/blog/blob/master/201809/20180917_01.md
- **1、1000亿TPCB 初始化**
- 耗时：25小时 52分。
- 速度：约107万行/s。
- **2、1000亿TPCB 创建索引（64并行）**
- 耗时：10小时 50分。
- 速度：约250万行/s。
- **3、1000亿TPCB 空间占用**
- 表：12.51 TB
- 索引：2 TB

单表数据量	TEST CASE	QPS	TPS
1000亿	tpcb 活跃数据10亿 只读	998818	998818
1000亿	tpcb 活跃数据100亿 只读	597877	597877
1000亿	tpcb 活跃数据500亿 只读	66678	66678
1000亿	tpcb 活跃数据1000亿 只读	67295	67295
1000亿	tpcb 活跃数据10亿 读写	475595	95119
1000亿	tpcb 活跃数据100亿 读写	426390	85278
1000亿	tpcb 活跃数据500亿 读写	191505	38301
1000亿	tpcb 活跃数据1000亿 读写	175945	35189

TPC-C (ECS 32C, 512G, SSD)

- https://github.com/digoal/blog/blob/master/201809/20180913_01.md
- (1000W数据) 10*100W
- 103万 tpmC



TPC-H (ECS 32C, 512G, SSD)

- https://github.com/digoal/blog/blob/master/201808/20180823_01.md
- SF=10 TPCH 150秒
- SF=200 TPCH 39 min

SF	q1	q2	q3	q4	q5	q6	q7	q8	q9	q10	q11	q12	q13	q14	q15	q16	q17	q18	q19	q20	q21	q22
10	3	2	2	5	3	4	2	2	9	3	1	2	4	2	6	10	27	46	2	10	13	2
200	18	38	25	32	57	8	52	24	66	38	24	26	98	13	58	114	732	595	12	213	124	14

OLAP Benchmark(56 Core ECS, PG 10)

CASE	数据量	并发	TPS	平均响应时间
多表JOIN	10张1000万的表	112	10万	1.08毫秒
大表JOIN、统计	2张1亿，1张100万	56	2.2万	2.5毫秒
大表OUTER JOIN、统计	1千万 OUTER JOIN 1亿			1千万 left join 1亿：11秒 反之：8秒
用户画像-数组包含、透视	1亿，每行16个标签	56	1773	31毫秒
用户画像-数组相交、透视	1亿，每行16个标签	56	113	492毫秒
用户画像-varbitx	1万行，2000亿BIT ，与或非			2.5秒
用户画像-多字段任意搜索\聚合、透视	1亿，32个字段，任意字段组合查询	56	3.6万	1.56毫秒
物联网-线性数据-区间实时聚合、统计	1万传感器，10亿记录	56	6266	8.9毫秒

OLAP Benchmark(56 Core ECS, PG 10)

CASE	数据量	并发	TPS	平均响应时间
排序	1亿	32		1.4秒
建索引	1亿			PG 10: 38秒； PG11: 15.5秒
并行扫描	1亿	32		0.88秒
并行聚合	1亿	32		0.9秒
并行过滤	1亿	32		1秒
并行JOIN+聚合	1000 万 JOIN 1000 万	32		PG 10: 5.4秒； PG 11: 1秒
并行JOIN+聚合	1亿 JOIN 1亿 (双表过滤到1000万)	32		PG 10: 6.3秒； PG 11: 1.2秒
并行JOIN+聚合	1亿 JOIN 1亿 (单表过滤到1000万)	32		PG 10: 8.5秒； PG 11: 2秒
并行JOIN+聚合	1亿 JOIN 1亿 (无条件JOIN)	32		PG 10: 58.3秒； PG 11: 10.7秒
并行JOIN+聚合	10亿 JOIN 10亿 (双表过滤到1000万)	32		PG 10: 12秒； PG 11: 1秒

OLAP Benchmark(56 Core ECS, PG 10)

CASE	数据量	并发	TPS	平均响应时间
并行HASH AGG	10亿 (PG 11)	64		11秒
VOPS + 异步并行	10亿，聚合查询（PG 10）	56		2秒
多表并行扫描 (parallel append)	2亿 (PG 11)	64		0.6 秒
求TOP-K	100亿 (PG 11)	64		40秒

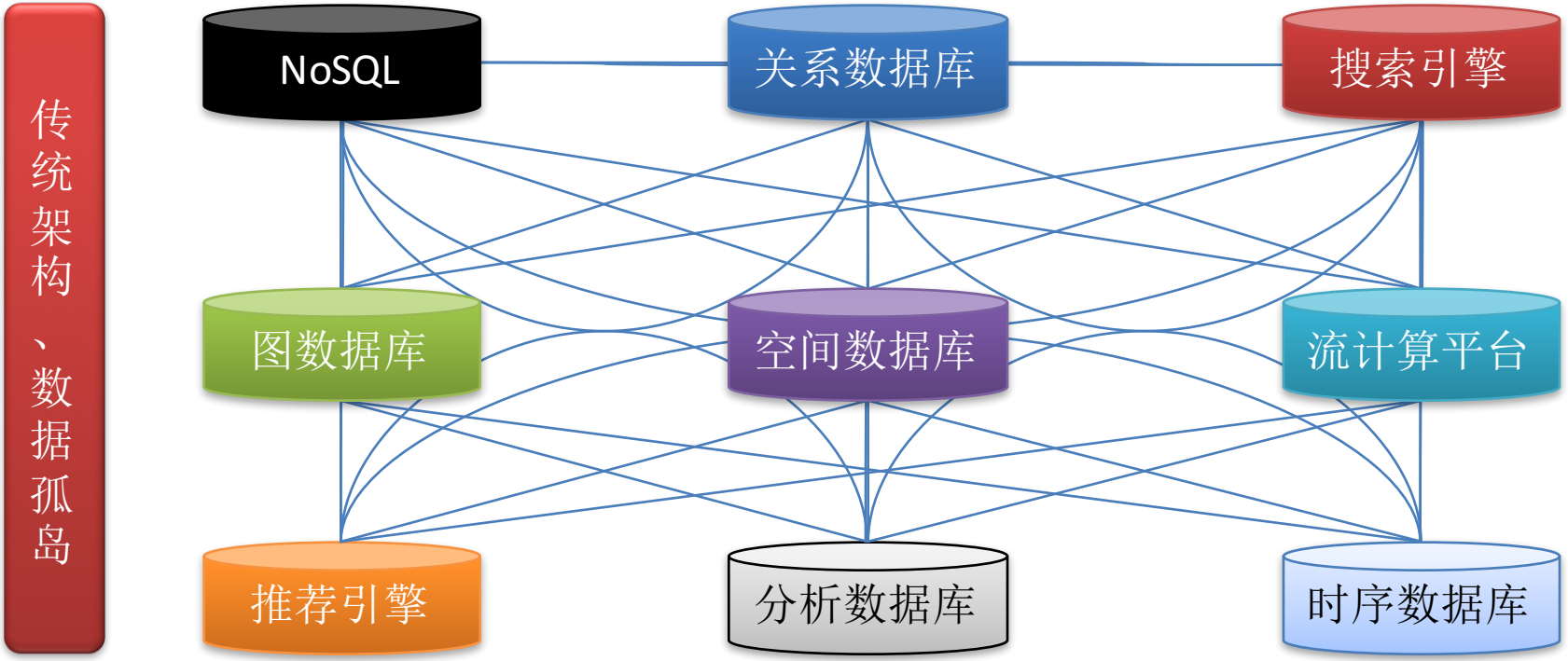
未来的发展方向展望

- 全栈数据库
- 一份存储 HTAP
- 计算存储分离
- 弹性扩容
- 计算下推

传统架构痛点

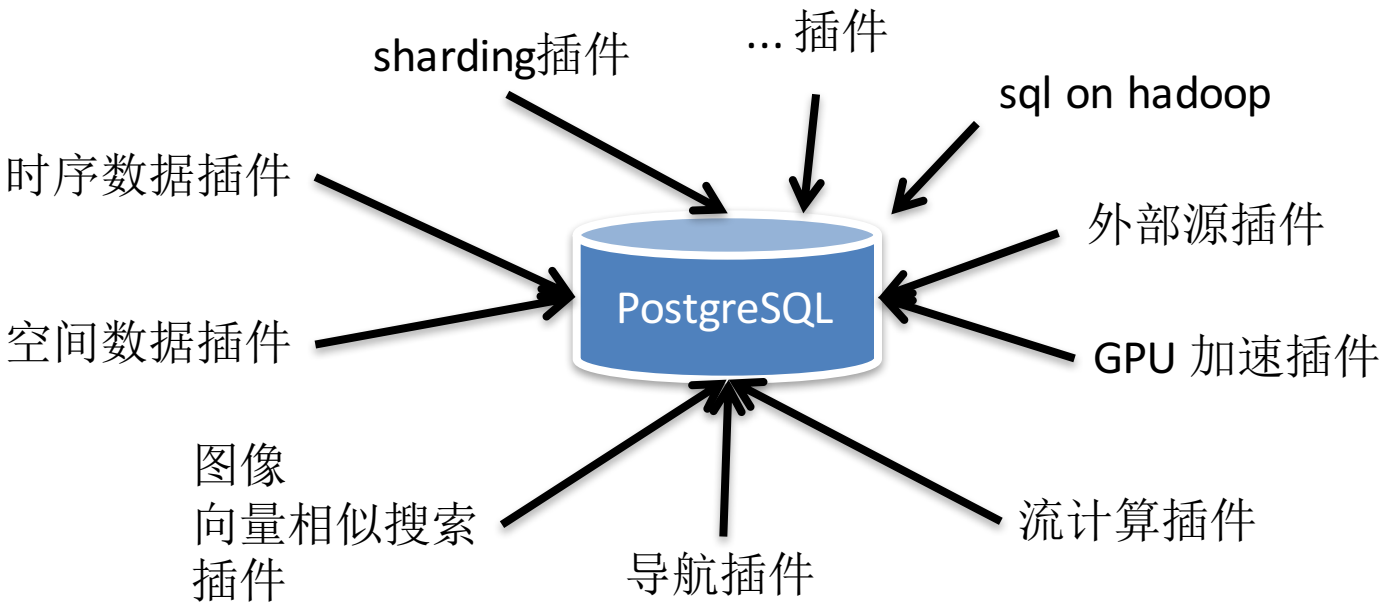
跨数据库痛点：同步延迟、一致性、原子性、可靠性等问题。

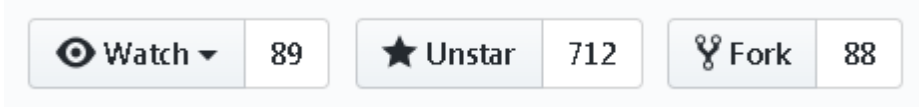
成本痛点：软件成本、硬件成本、研发成本。



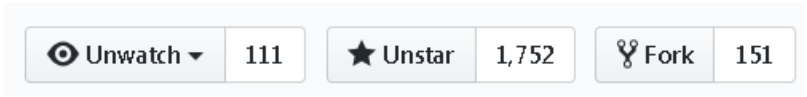
PostgreSQL 优势

- PG支持热插拔扩展

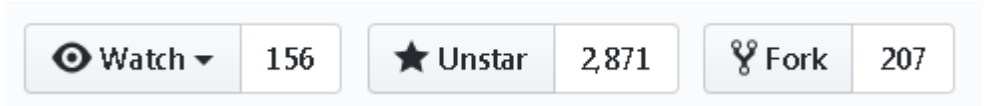




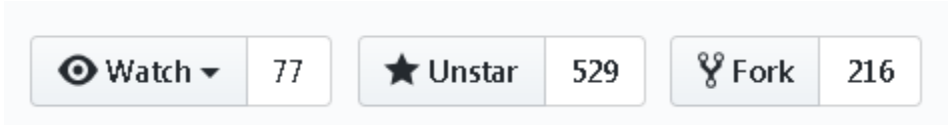
pg-strom GPU加速



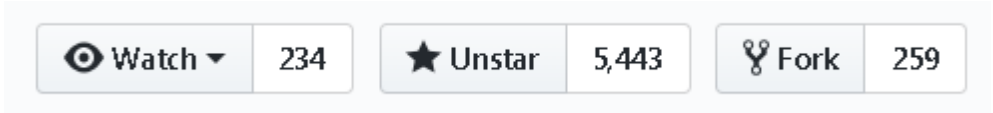
pipelinedb 流计算



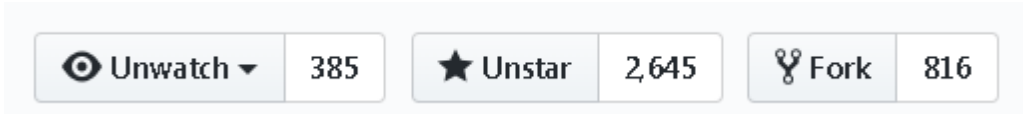
citus sharding



PostGIS 时空



timescaleDB 时序

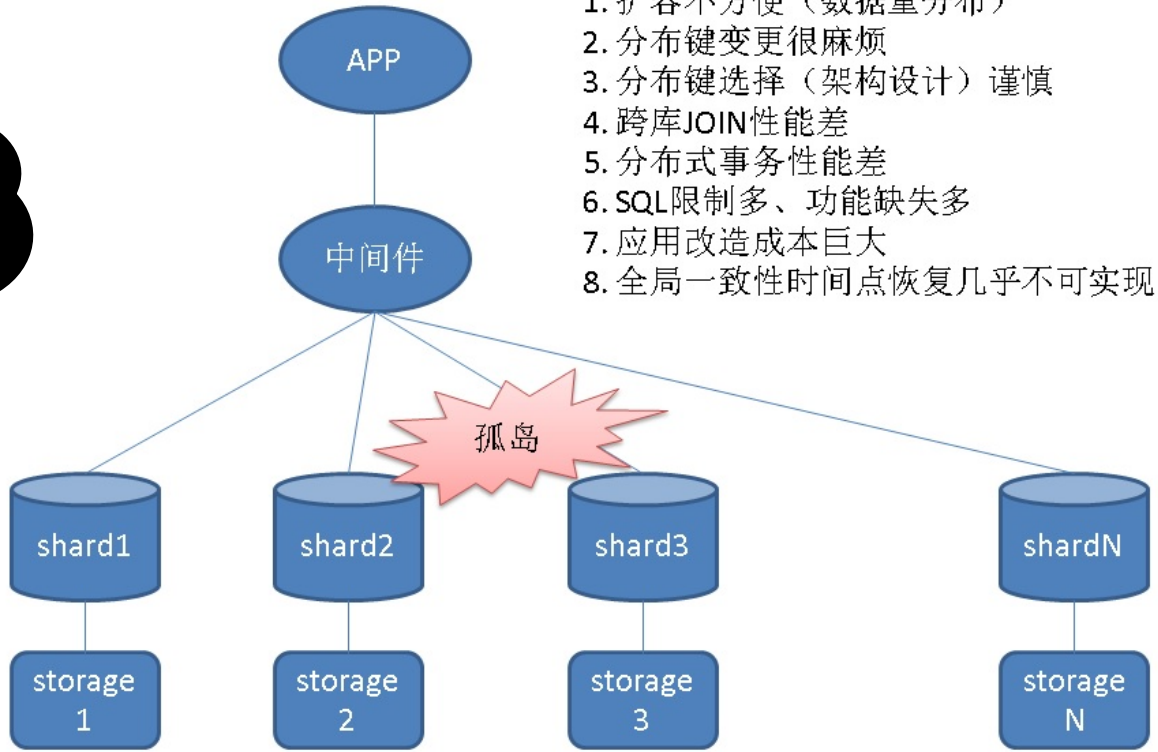


GPDB MPP

- 完全兼容PostgreSQL。
- 计算存储分离版；
- 存储接口自定义（支持下推计算）；
- 计算节点支持水平扩展；
- (定位高端用户、HTAP混合应用)

传统分库分表架构缺陷

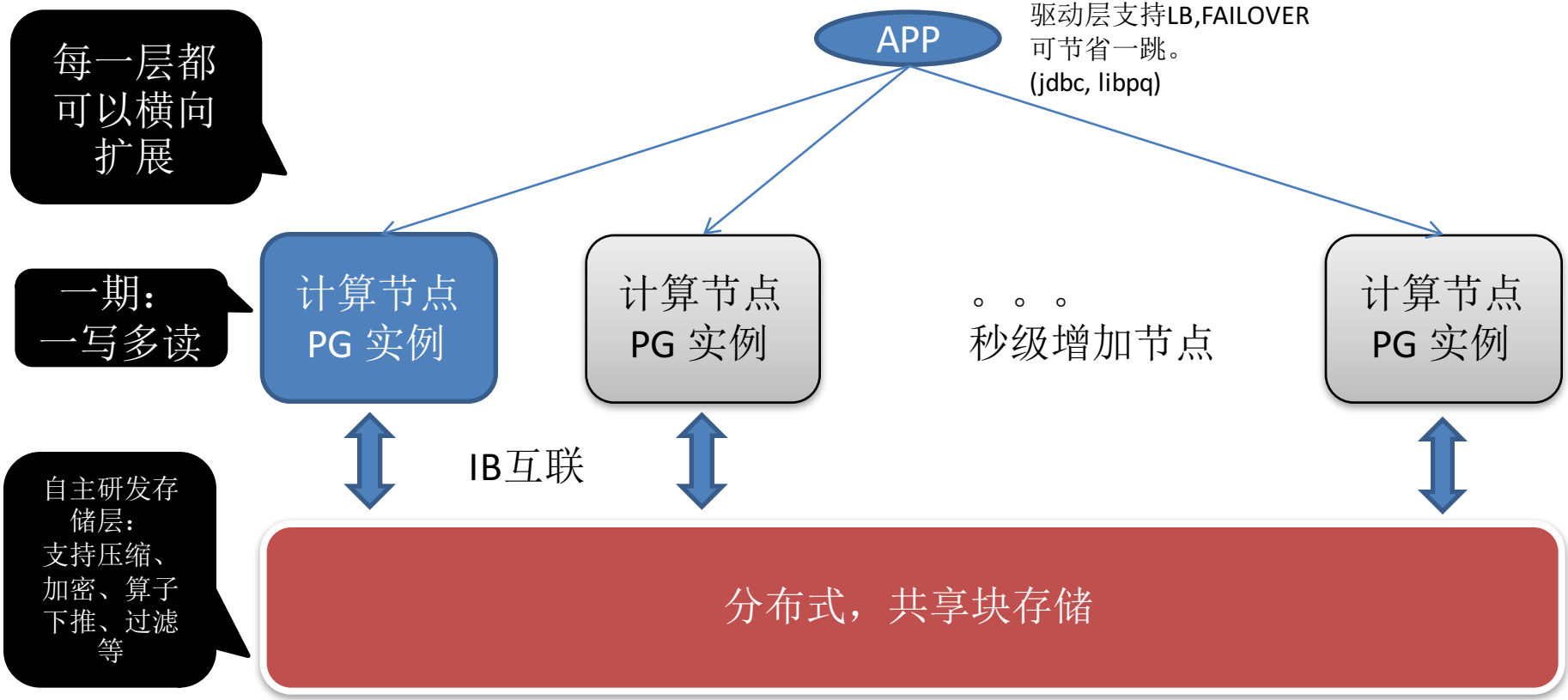
传统分库分表
架构：
限制、缺陷。



1. 扩容不方便（数据重分布）
2. 分布键变更很麻烦
3. 分布键选择（架构设计）谨慎
4. 跨库JOIN性能差
5. 分布式事务性能差
6. SQL限制多、功能缺失多
7. 应用改造成本巨大
8. 全局一致性时间点恢复几乎不可实现

shard之间隔绝、需要跨节点JOIN的话要在中间件层实现、效率低

阿里云PolarDB for PostgreSQL



谢谢

天天象上活动预告

标题	活动时间
PostgreSQL生态、原理、应用案例、开发与管理实践 - 天津站	2019-06-15 09:15 ~ 18:00
PostgreSQL生态、原理、应用案例、开发与管理实践 - 济南站	2019-05-11 09:15 ~ 18:00
PostgreSQL生态、原理、应用案例、开发与管理实践 - 重庆站	2019-04-14 09:15 ~ 18:00
PostgreSQL生态、原理、应用案例、开发与管理实践 - 成都站	2019-04-13 09:15 ~ 18:00
PostgreSQL生态、原理、应用案例、开发与管理实践 - 长沙站	2019-03-09 09:15 ~ 18:00
PostgreSQL生态、原理、应用案例、开发与管理实践 - 武汉站	2019-01-12 09:15 ~ 18:00
PostgreSQL生态、原理、应用案例、开发与管理实践 - 南京站	2018-12-15 09:15 ~ 18:00
PostgreSQL生态、原理、应用案例、开发与管理实践 - 上海站	2018-11-17 09:15 ~ 18:00
PostgreSQL生态、原理、应用案例、开发与管理实践 - 深圳站	2018-10-28 09:15 ~ 18:00
PostgreSQL生态、原理、应用案例、开发与管理实践 - 广州站	2018-10-27 09:15 ~ 18:00

谢谢