PostgreSQL
高并发数据库应用数据

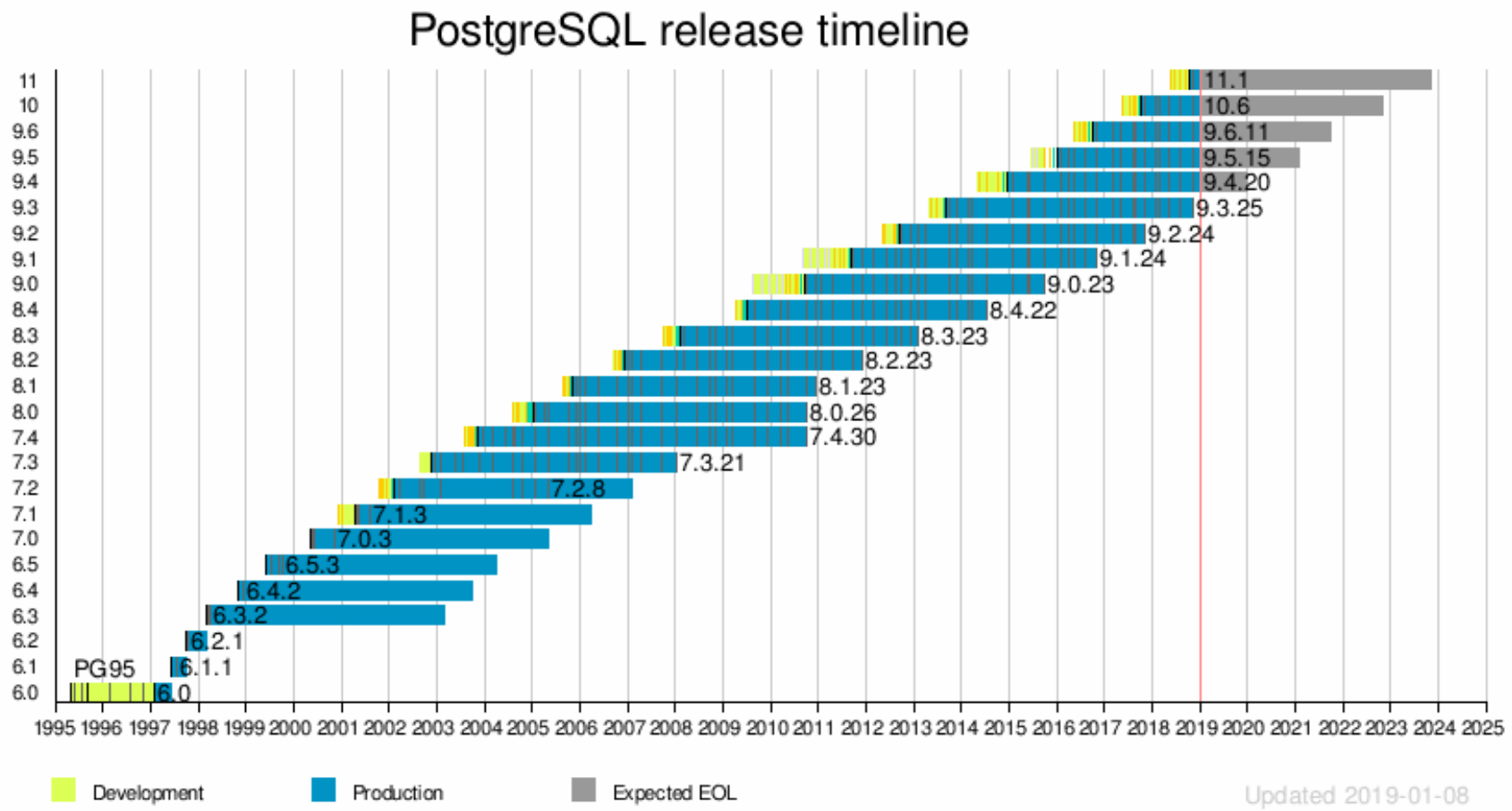
阿里云
digoal

目录

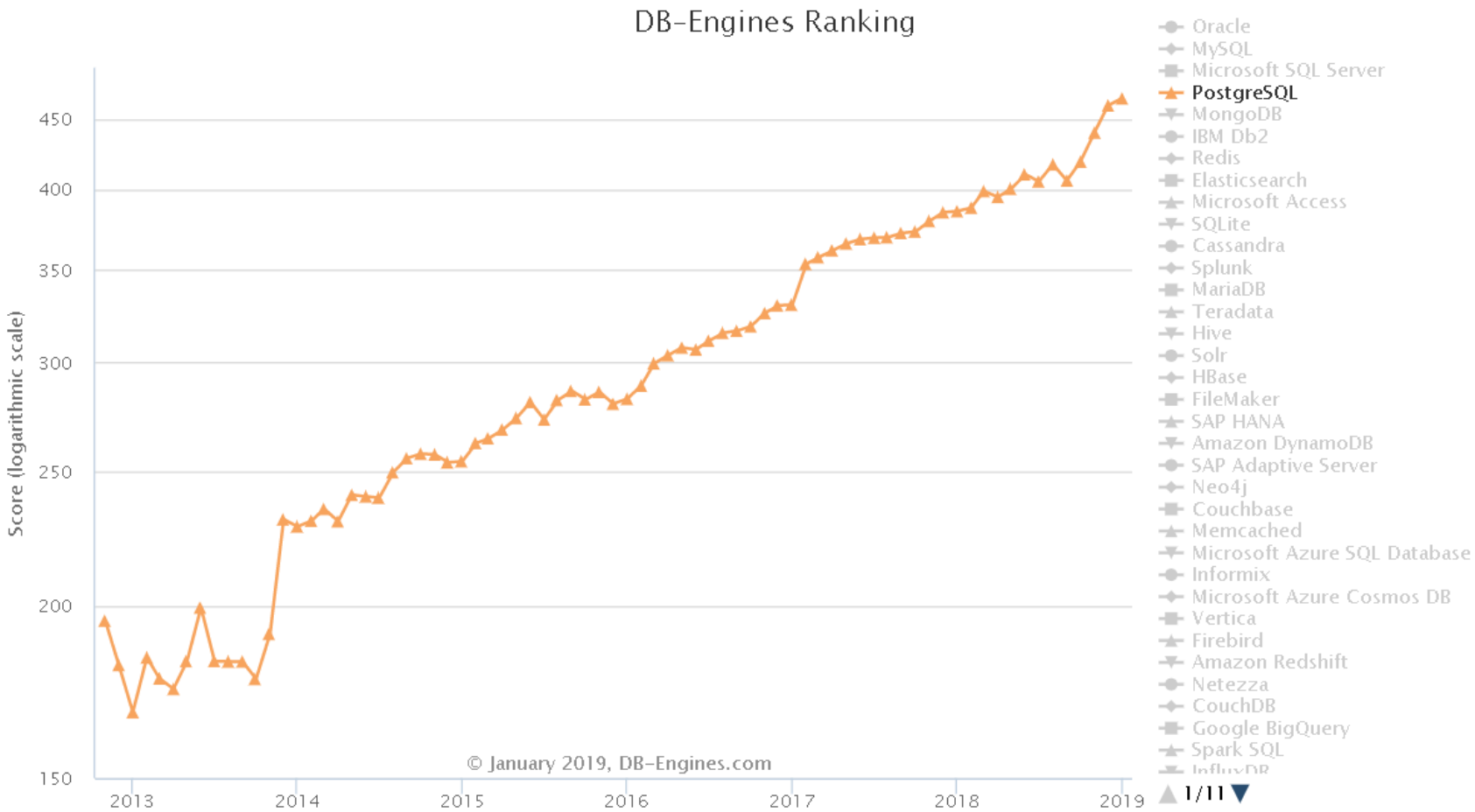
- PG生态
- 常见的高并发场景与业务
- 高并发场景带来的挑战
- 高并发场景数据库设计与优化
- 阿里RDS PG在高并发场景的内核改进
- 案例

PG生态

PG 版本发布周期

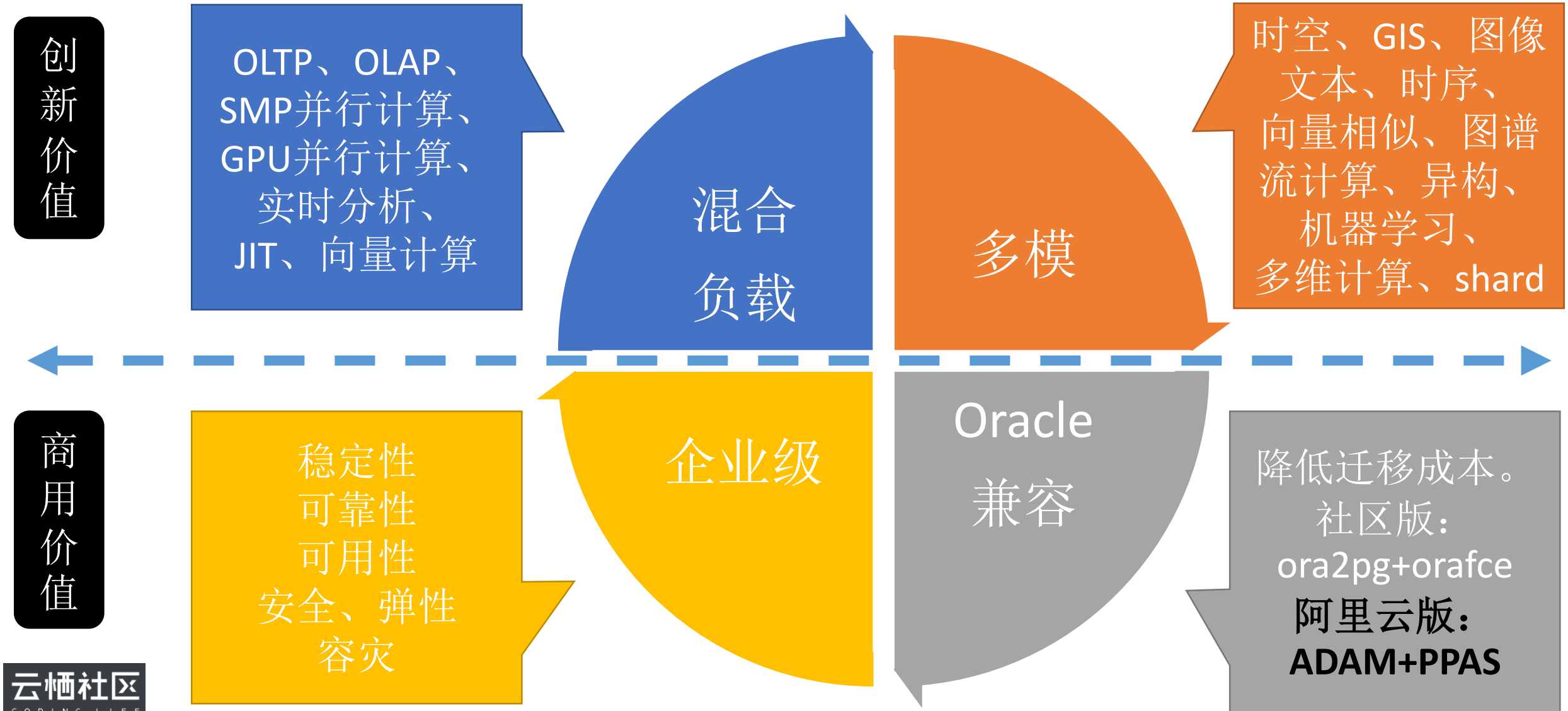


2017,2018被评为年度数据库



2018 中国PG用户会

PG 定位-企业数据库



PG 解决了企业最急迫的问题

- **合规、合法、可控(BSD-like许可)**
  - 都是开源，许可大不同。
- **企业级业务对数据库的基本诉求**
  - 可靠性、可用性、稳定性、安全性、扩展性、弹性、性能、合规
- **企业“去O”推进超车道**
  - 兼容Oracle
  - 媲美Oracle的优化器
    - 高并发，烂SQL，复杂SQL，框架生成SQL，计算型SQL通吃
      - **解决传统行业开发者水平参差不齐问题**

PG 典型企业用户

PostgreSQL 全球财富1000的用户

Accenture 埃森哲
ADP
Aetna 安泰保险
AT&T
AutoZone
BAE Systems
Banco do Brasil 巴西银行
Boeing 波音公司
Bouygues Telecom
Broadcom 博通公司
Cisco Systems 思科公司
Citigroup 花旗集团
Cognizant Technology Solutions
Computer Associates
Computer Science Corporation
Deere & Company
Dell 戴尔公司
Deutsche Boerse AG 德国证券交易所
eBay 易趣网
EMC Corporation 易安信公司
Emerson Electric
Ericsson
Fujitsu 富士通公司
General Electric (GE) 通用公司
Google 谷歌公司
Grupo BBVA
HP 惠普公司
IBM 公司

ICICI
Infosys 公司
JPMorgan Chase
KDDI
KT 韩国电信
Kubota
Kyocera
Lockheed Martin

MasterCard International
McKesson
Mizuho Information & Research Institute
Mosaic ATM
Motorola 摩托罗拉公司
NEC 日电公司
NTT 日本电信
Nokia 诺基亚

Northrop Grumman
Nucor
ONGEI
Panasonic 飞利浦
QUALCOMM
Raytheon
RSA
SAP 公司
Schneider Electric
Seagate 希捷
Siemens 西门子
SK Telecom
Softbank 日本软件银行
Sony 索尼公司
Swisscom 瑞士电信
Symantec 赛门铁克
Syngenta Crop Protection
Tata Consultancy Services
Telstra 澳洲电信
The GAP 时尚服饰
Tokio Marine & Nichido Fire Insurance
Toyota
Union Pacific Railroad
VMWare 公司
Walt Disney
Wipro
Xerox
Yahoo 雅虎公司

国际典型用户

- 制造业：大量日系、德系汽车及其另配件生产线使用PostgreSQL
- 电信业：以亚太区例 NTT、KT、台湾大哥大
- 金融业：星展银行、mastercard、德国证券交易所、西班牙储蓄银行、荷兰ABN集团
- 政府：NASA、欧洲宇航局、美国海空军、法国政府、波兰政府
- 互联网：苹果、Skype、Yahoo、SOE
- 公共软件：SAP、saleforce
- 其他：思科、EMC、软银、英国乐透、SAP、

常见的高并发场景与业务

- 2C
- 秒杀
- 运营活动
- 热点事件
- 游戏
- 物联网（IoT）
- 车联网

高并发场景带来的挑战

- 高并发短连接挑战：建立连接成本高，效率低
- 进程模式
  - 高并发场景：进程调度开销大，效率低下
- 锁竞争问题
  - 隔离级别越高，问题可能越明显
- 死锁隐患
- 雪崩隐患
- 单实例多业务（在线、分析）混合使用、攻击，问题SQL：干扰、攻击、抖动隐患
- 高并发小事务挑战：IO刷盘频率高
- 计算能力挑战
- 读写分离，高压下的从库延迟，成本挑战等
- 高并发写压力下的索引IO 引入RT增加
- 内存挑战（长连接霸占会话级缓存、PROC touch shared buffer hashtable、分区relcache）

高并发场景数据库设计与优化

- 内置、外置连接池、长连接
- Huge page
- Release session memory context
- PGA
- pg_pathman

- AD Lock
- 乐观锁

- 异步提交
- 组提交

- 锁超时

- 预计算、流计算(pipelinedb)

- 用户、DB级资源隔离

- Sharding
- 读写分离
- 计算存储分离(POLARDB PG)

- GIN fast update

阿里云PG产品线

支持Oracle\PG两套协议

(企业级+Oracle兼容)

云数据库 PPAS 版

POLARDB 双机版

支持Oracle\PG两套协议
计算存储分离
计算、存储横向弹性扩容、缩容

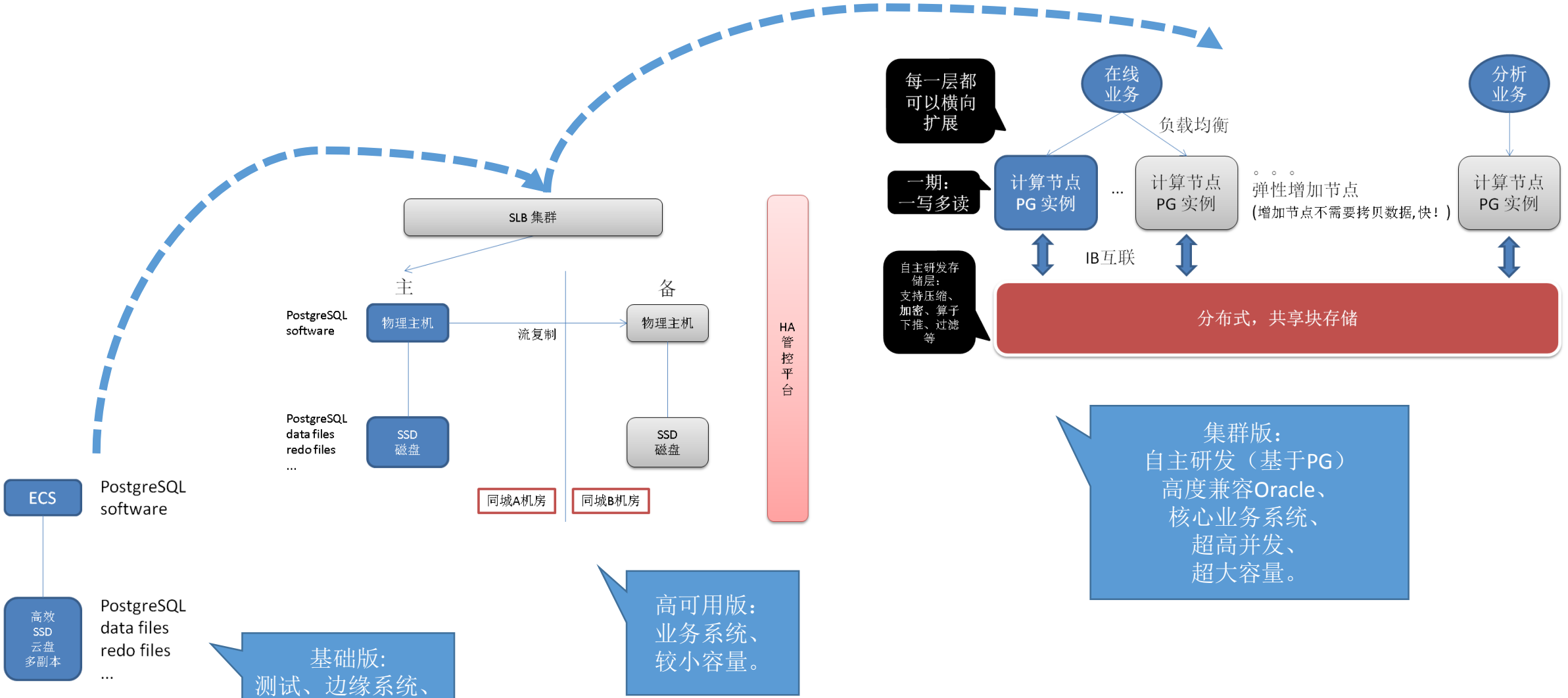
100TB OLTP+OLAP+多模混合处理

POLARDB for PG

POLARDB 集群版

开源增强版
SMP、GPU
6TB OLTP+OLAP

产品架构



阿里RDS PG在高并发场景的内核改进

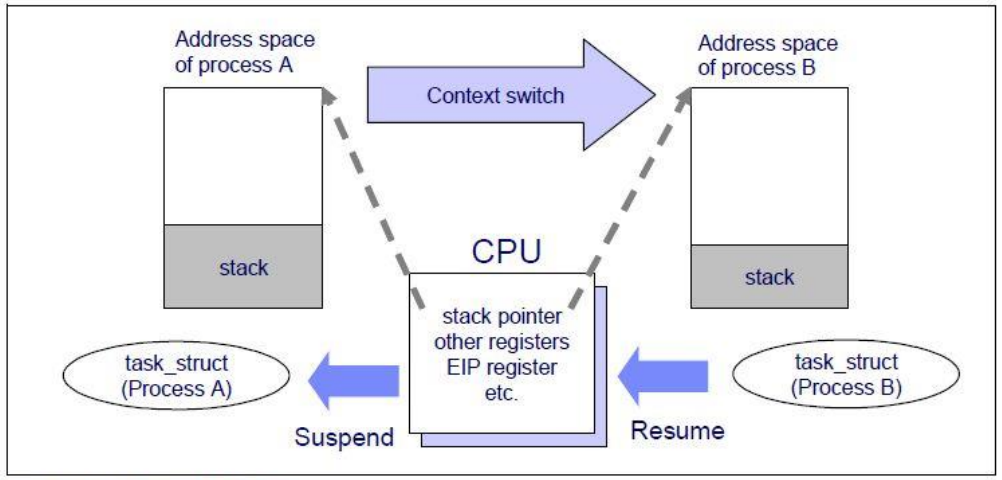


Figure 1-5 Context switching

调度开销

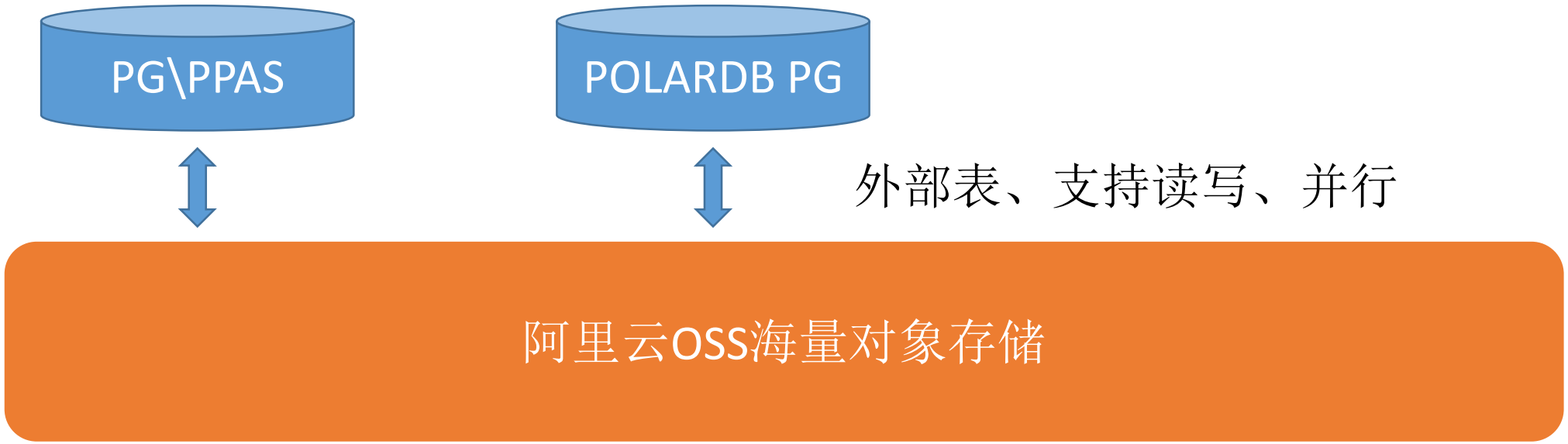
连接数	社区版本RT	阿里云版本RT
64	0.475 ms	0.501 ms
128	0.934 ms	0.854 ms
256	2.109 ms	1.842 ms
512	4.656 ms	4.587 ms
1024	9.837 ms	8.69 ms
2048	36.882 ms	7.928 ms
4096	67.513 ms	7.522 ms
8192	201.208 ms	6.536 ms
16384	65428.243 ms	4.811 ms

池化，优化

关注社区版与ali版量级的差别。
单次测试，
RT受其他干扰，
不必纠结。

存储 - 支持冷热数据分离

- 通过OSS扩展无限容量



为什么要研发POLARDB PG

- 计算存储分离集群版

这些问题，你都遇到过吗？

老板要看活动运营数据，有一个复杂大SQL查询非常慢，半小时了还没出结果。。。

活动上线，压力突增，数据库来不及升级了

使用读写分离，刚更新的数据，查询不到。。。

添加一个字段，备库延迟3小时。。

主从复制经常中断，1236 1042错误

数据库即将超过3T，业务发展很快，没有时间做分库分表的改造

备份时会锁表，必须在业务低峰期操作

1T数据量，一次备份竟然需要十几个小时

上百台ECS连接同一台数据库，高并发下性能问题凸显

问题根源？

- SQL只能用到**单CORE**，慢
- 升级需要**迁移数据**，慢
- 增加只读节点，需要**迁移数据**，慢
- 读写分离依靠**SLAVE增量复制、回放**，延迟高
- 添加字段带来**数据重写**，逻辑SLAVE回放需要**等待上游事务结束**再开始，延迟高
- **主节点写压力大**，逻辑SLAVE容易中断
- 存储使用**本地磁盘**，容易达到**空间上限**
- **逻辑备份**时，表结构不能变更，可能出现**锁冲突**
- 备份需要拖全量，**实例越大，耗时越久**
- **高并发**访问时，实例扛不住

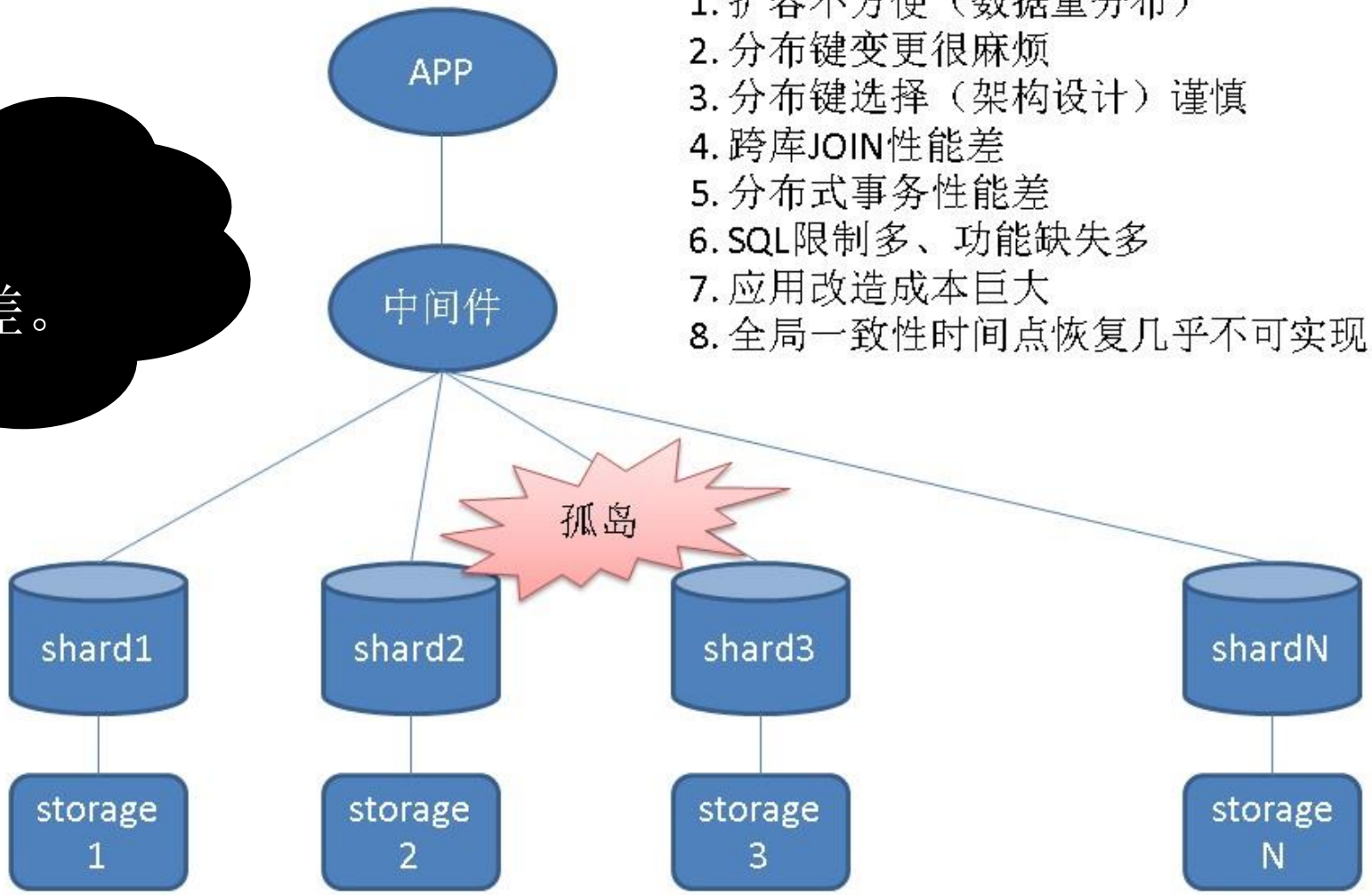
POLARDB PG vs 社区版

	POLARDB 集群版	开源版
成本	低 1、存储按量收费 2、只读节点只收计算资源的钱	一般 1、存储预收费 2、只读节点需要复制数据，节点越多，价格越贵
计算弹性	好 1、分钟级添加节点	一般 1、取决于实例大小，以及是否需要跨机伸缩
存储弹性	好 1、近乎无限扩容，对业务无影响 2、容量上限100T	一般 1、扩容可能需要迁移数据 2、容量上限几T
只读节点延迟	低 1、共享数据，延迟低	一般 1、需要复制、增量回放。
备份	快 1、秒级快照备份	一般 1、取决于数据库大小，越大越慢，备份时可能影响性能
恢复	快 1、快照克隆，秒级	一般 1、取决于数据库大小，越大恢复可能越慢
可靠性	高 1、存储多副本，parallel raft协议，高效，保证强一致。RPO=0	高 1、依靠STANDBY，异步模式有数据丢失可能性 2、依靠STANDBY，同步模式quorumbased sync replica，RPO=0，损耗一定性能。
可用性	高 1、异常failover：由于共享数据，切换时不需要等待恢复，切换时间非常短暂。 2、未来支持multi-writer模式，做到zero downtime。	一般 1、异常failover切换时，需要等待备库同步完成，激活，切换。流程较长。
性能	高 1、分布式块存储，RDMA网络，横向的能力扩展。 2、支持存储级计算、压缩、filter下推。 3、计算节点支持mpp模型，加速复杂计算SQL。 4、支持列存。 5、支持GPU加速。	一般 1、存储依赖本地设备能力。

传统分库分表架构 - 缺陷

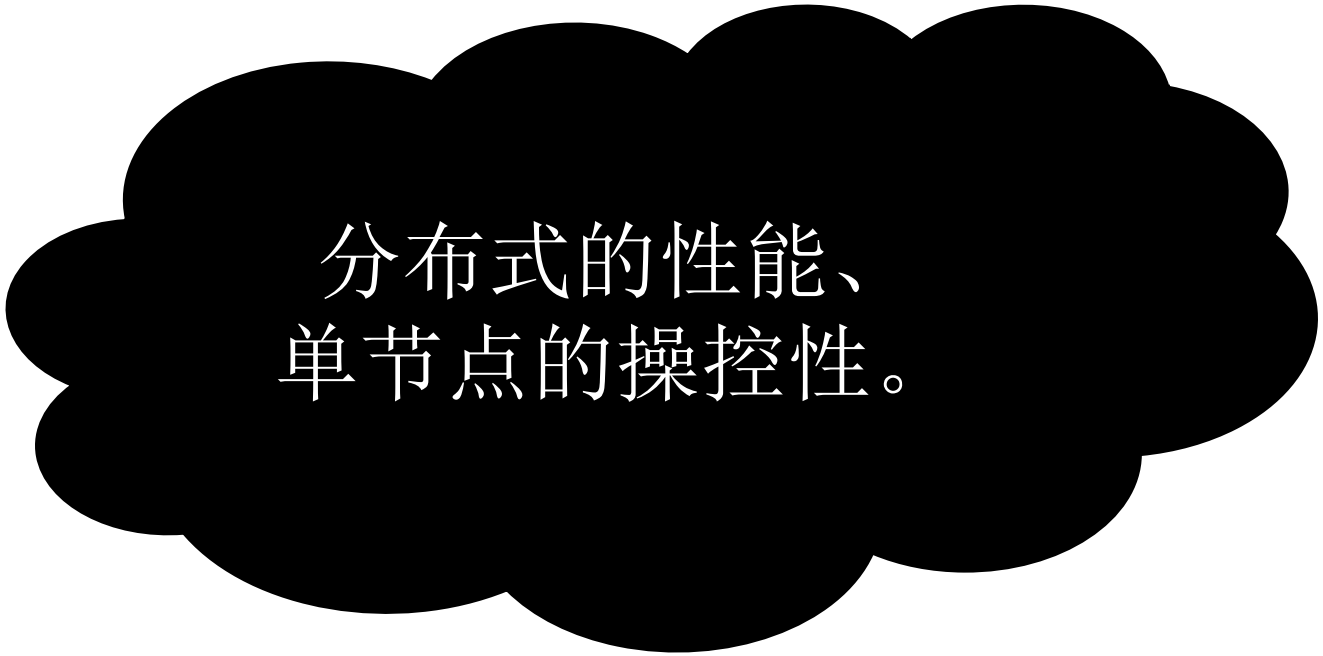
传统分库分表架构：
限制、缺陷、操控差。

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

POLARDB PG



案例 -乐观锁提高处理吞吐

https://github.com/digoal/blog/blob/master/201901/20190118_02.md

性能对比

数据量	隔离级别	TPS(吞吐能力)	QPS	正常事务	冲突比例
5亿	rc	94528	472640	11343739	0%
5亿	rr	87957	439785	638001	93.95%
5亿	ssi	43058	215290	411820	92.03%

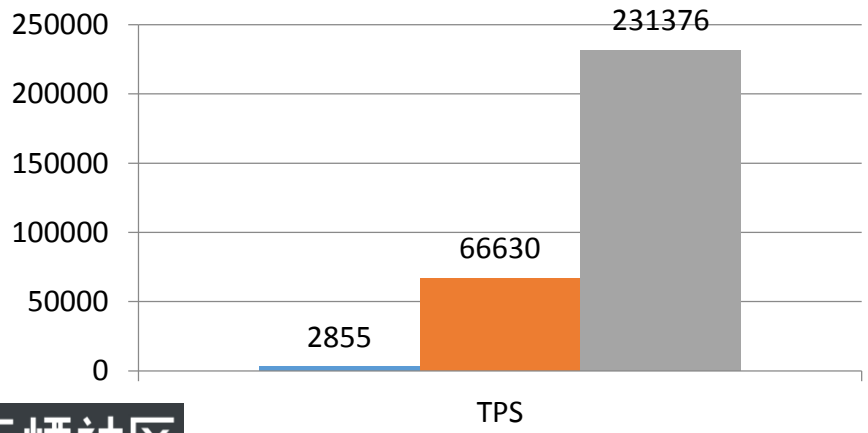
案例 - 秒杀

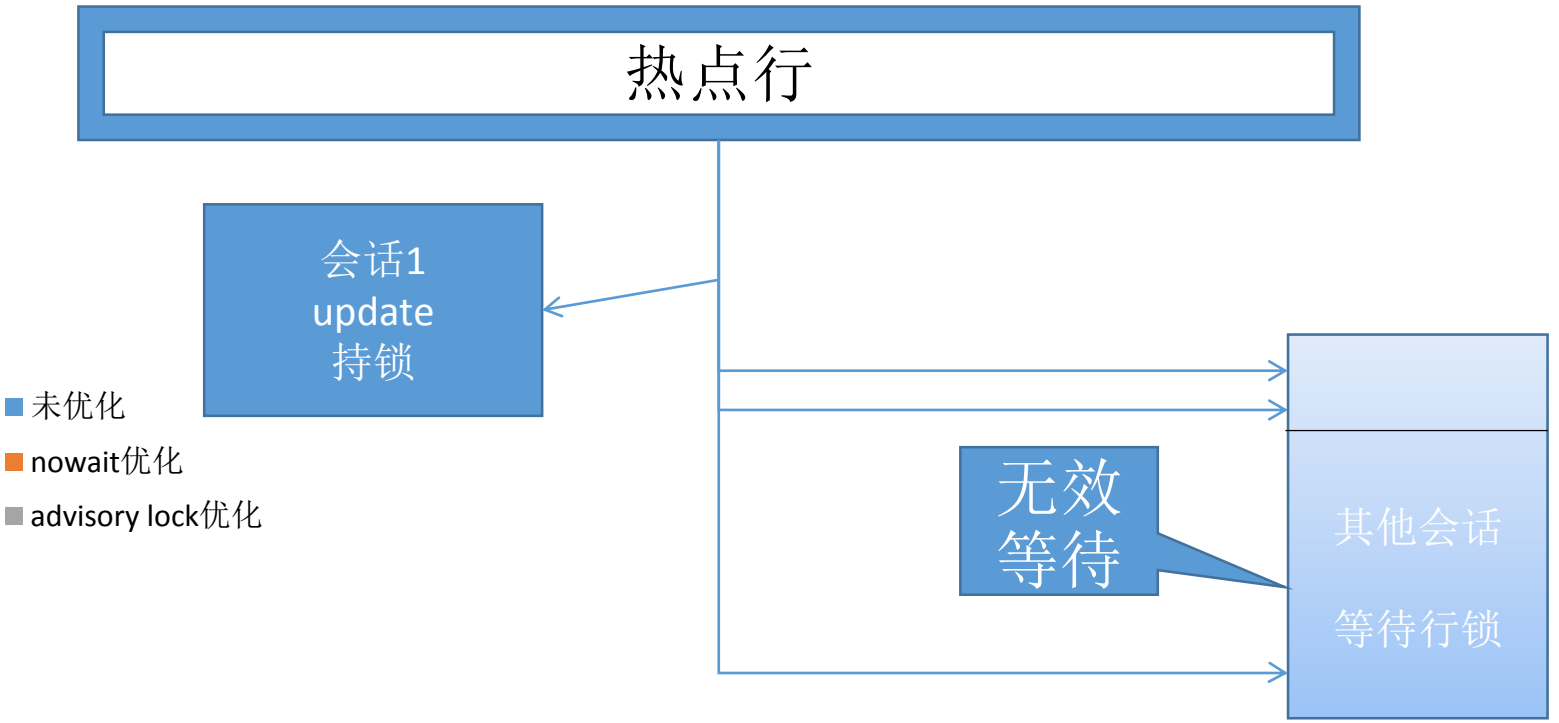
- 秒杀

秒杀

- 超轻锁 (advisory LOCK) 解决高并发锁竞争问题
  - 手段： 在CPU运算发现行锁之前就知道是不是有冲突，大大缩短CPU计算资源，等待资源

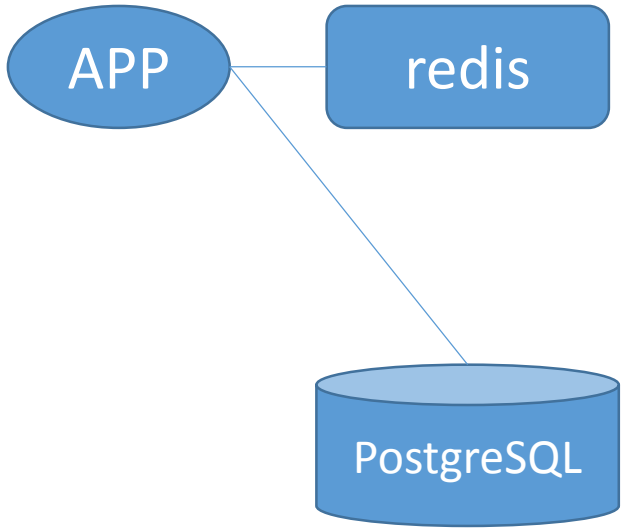
传统 - 行锁弊端
1. 无效等待多
2. 无效等待用户
长时间占用会话资源
3. 发现锁冲突的代码路径长
需要进行大量CPU运算

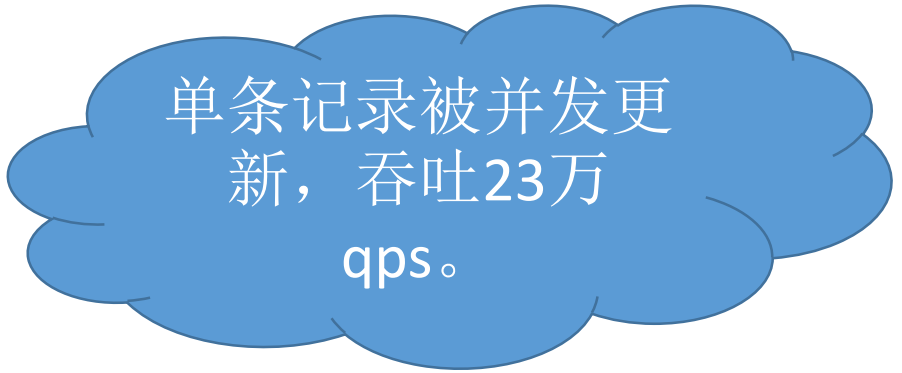




ADLock代替行锁 - 秒杀

- 高并发扣减库存
- 高并发争抢锁
  - update tbl set x=x where id=? and pg_try_advisory_xact_lock(id) returning *;

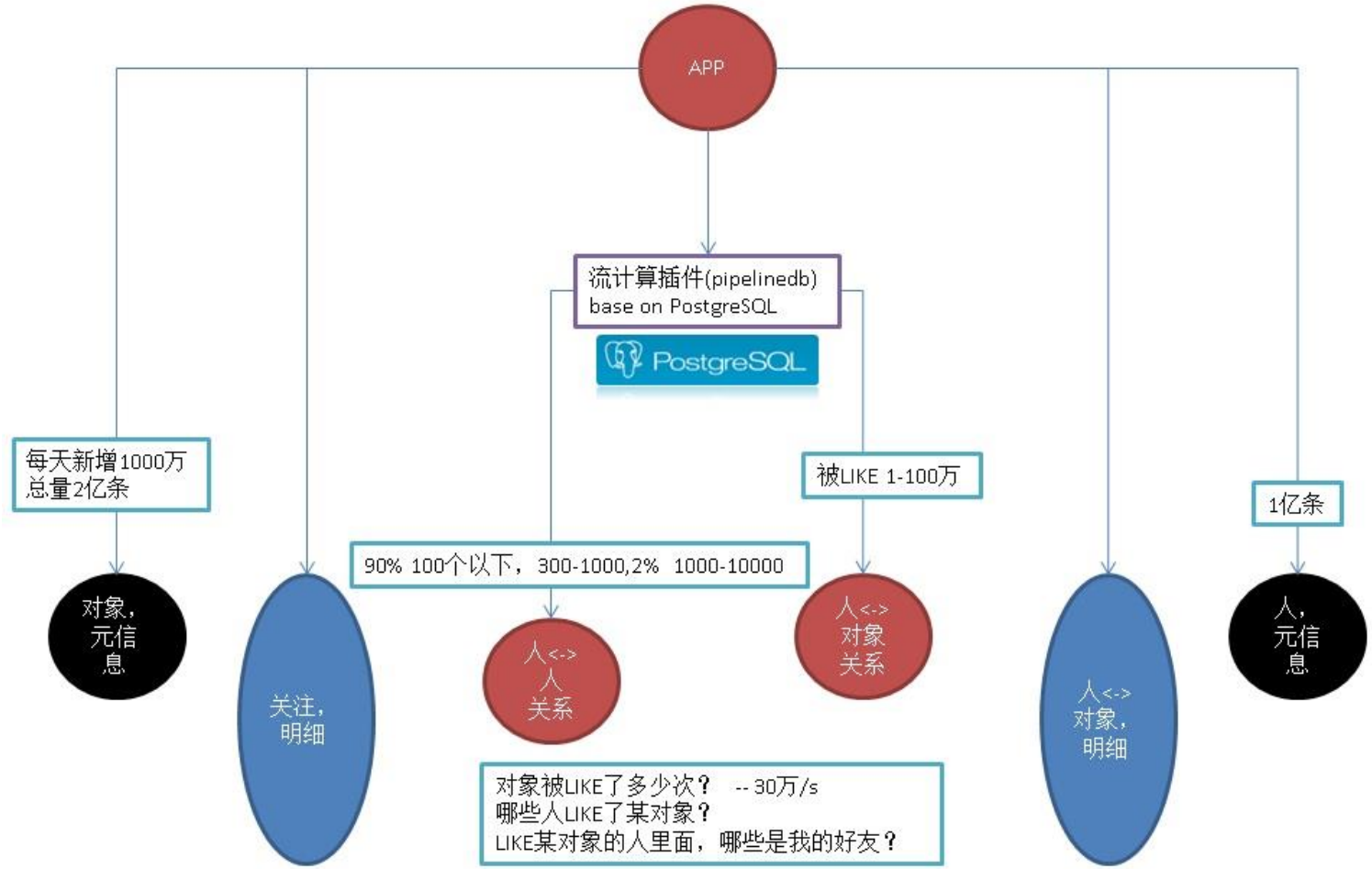




1. 连接redis判断是否还有库存
2. 有，去PG扣减(ADLock)。没有则直接返回。
3. 扣减成功，去redis更新库存

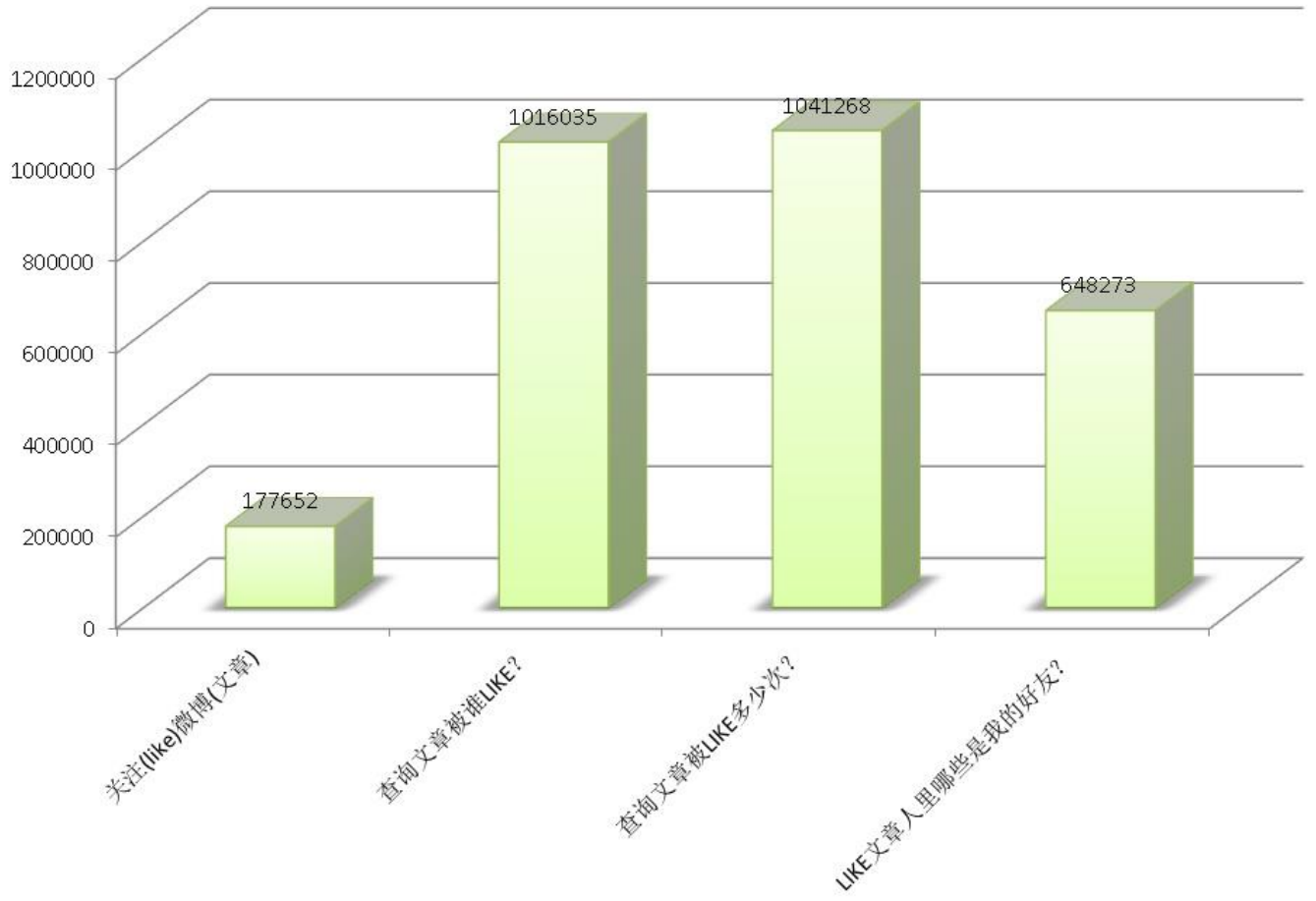
案例-流计算，PCC大赛

https://github.com/digoal/blog/blob/master/201705/20170512_02.md



案例-流计算，PCC大赛

https://github.com/digoal/blog/blob/master/201705/20170512_02.md



案例-流计算，时空轨迹实时聚合

- https://github.com/digoal/blog/blob/master/201811/20181101_02.md
- 大并发点上报
- 点在HEAP中散落存储
- 大量轨迹查询（多点聚合），离散IO严重
- 相似轨迹、时态分析

- 流计算，实时聚合：
- 阿里PostgreSQL Ganos插件。

谢谢

digoal's 微信

PG进阶钉钉群
每周技术直播
专家问答