ALIBABA CLOUD

Alibaba Cloud

云原生数据仓库 AnalyticDB PostgreSQL 版

产品简介

文档版本：20220704

阿里云

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。

2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。

3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。

4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。

5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。

6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
危险	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	危险 重置操作将丢失用户配置数据。
警告	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	警告 重启操作将导致业务中断，恢复业务时间约十分钟。
注意	用于警示信息、补充说明等，是用户必须了解的内容。	注意 权重设置为0，该服务器不会再接受新请求。
说明	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	说明 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	单击**设置**> **网络**> **设置网络类型**。
粗体	表示按键、菜单、页面名称等UI元素。	在**结果确认**页面，单击**确定**。
Courier字体	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *Instance_ID*
[] 或者 [a|b]	表示可选项，至多选择一个。	`ipconfig [-all|-t]`
{} 或者 {a|b}	表示必选项，至多选择一个。	`switch {active|stand}`

目录

1.产品概述

云原生数据仓库AnalyticDB PostgreSQL版是一种大规模并行处理（MPP）数据仓库服务，可提供海量数据在线分析服务。

AnalyticDB PostgreSQL版基于开源项目Greenplum构建，由阿里云深度扩展，兼容ANSI SQL 2003，兼容PostgreSQL/Oracle数据库生态，支持行存储和列存储模式。既提供高性能离线数据处理，也支持高并发在线分析查询，是各行业有竞争力的PB级实时数据仓库方案。

主要功能

- 易适配，免调优

  支持SQL 2003，部分兼容Oracle语法，支持PL/SQL存储过程。新一代SQL优化器，实现复杂分析语句免调优。

- PB级数据秒级分析

  MPP水平扩展架构，支持PB级数据查询秒级响应。向量化计算及列存储智能索引，相比较传统数据库引擎在性能方面约有十倍的提升。

- 高可用，服务永远在线

  支持分布式事务，数据ACID一致性支持，所有节点和数据跨机器冗余部署，任意硬件故障，自动化监控切换，保持服务在线。

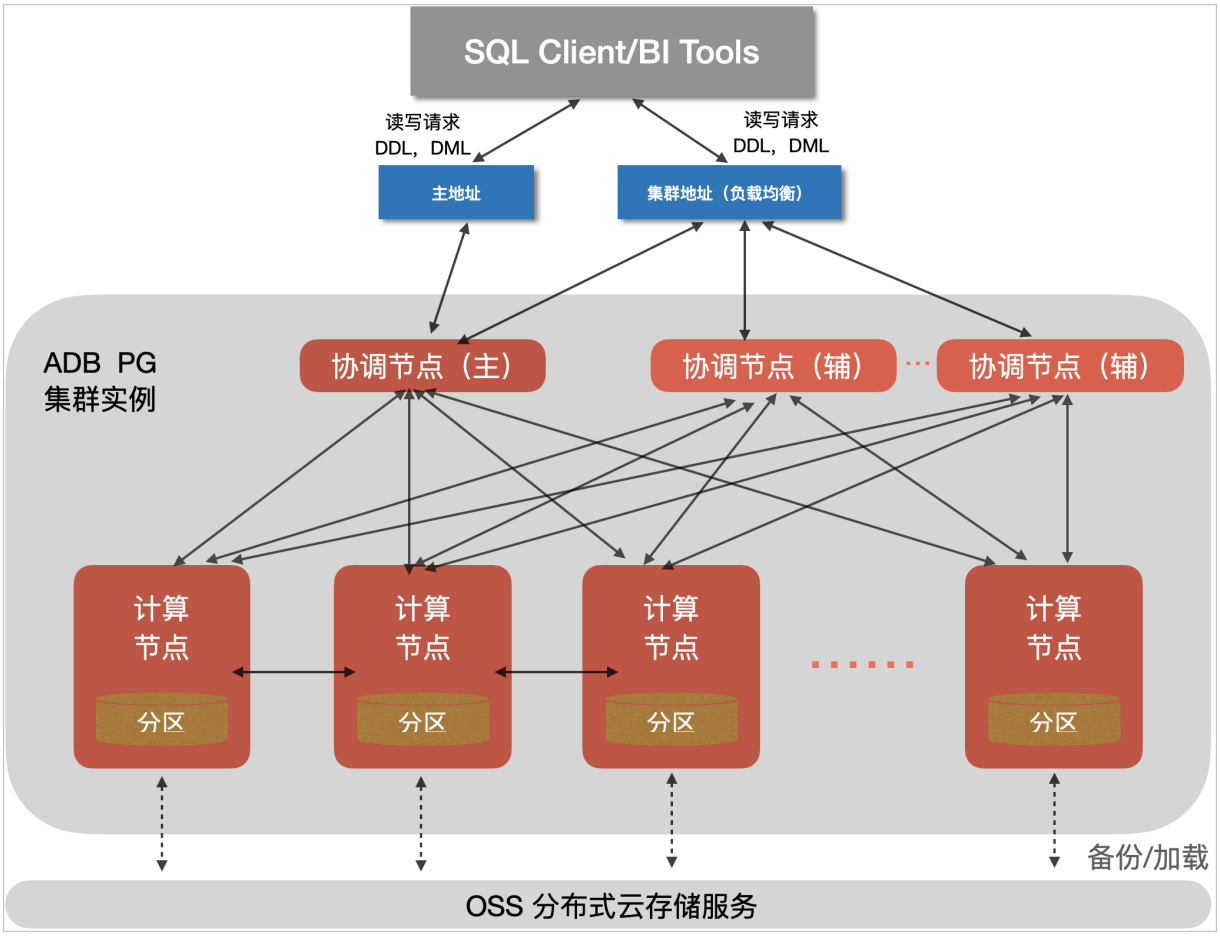
- 广泛生态兼容

  支持主流BI、ETL工具。通过PostGIS插件支持地理信息数据分析，MADlib库内置超过300个机器学习算法库。

- 数据互联互通

  支持通过DTS、DataWorks等工具，同多种数据源同步；支持高并行访问OSS，构筑数据湖分析。

产品架构



AnalyticDB PostgreSQL版采用MPP架构，实例由多个计算节点组成，存储磁盘类型支持高效云盘和ESSD云盘，计算和存储分离，可以独立增加节点或扩容，且保持查询响应时间不变。集群实例包括的组件有：

- 协调节点（Master Node）
  - 接收请求，制定分布式执行计划。
- 计算节点（Compute Groups）
  - 全并行分析计算。
  - 数据分区双副本存储。
  - 定期自动备份至OSS。

区别于Greenplum， 2021年2月8日，AnalyticDB PostgreSQL版正式开放多Master的能力，支持通过水平扩展协调节点（Master Node）来突破原架构单Master的限制，在计算节点不存在瓶颈的情况下，系统连接数及读写能力可以随着Master节点数增加实现线性扩展，从而进一步提升系统整体能力，更好的满足实时数仓及HTAP等业务场景的需求。对多master实例，在原有主地址基础之上新增了集群地址，详细信息请参见主地址和集群地址。

获取更多信息

- 云栖社区、AnalyticDB PostgreSQL版云栖号及技术支持钉钉群，详情请参见AnalyticDB for PostgreSQL实时数据仓库上手指南。
- 提交工单以获取人工帮助。
- Greenplum Database开源社区官方资料，请参见Greenplum Database官网。

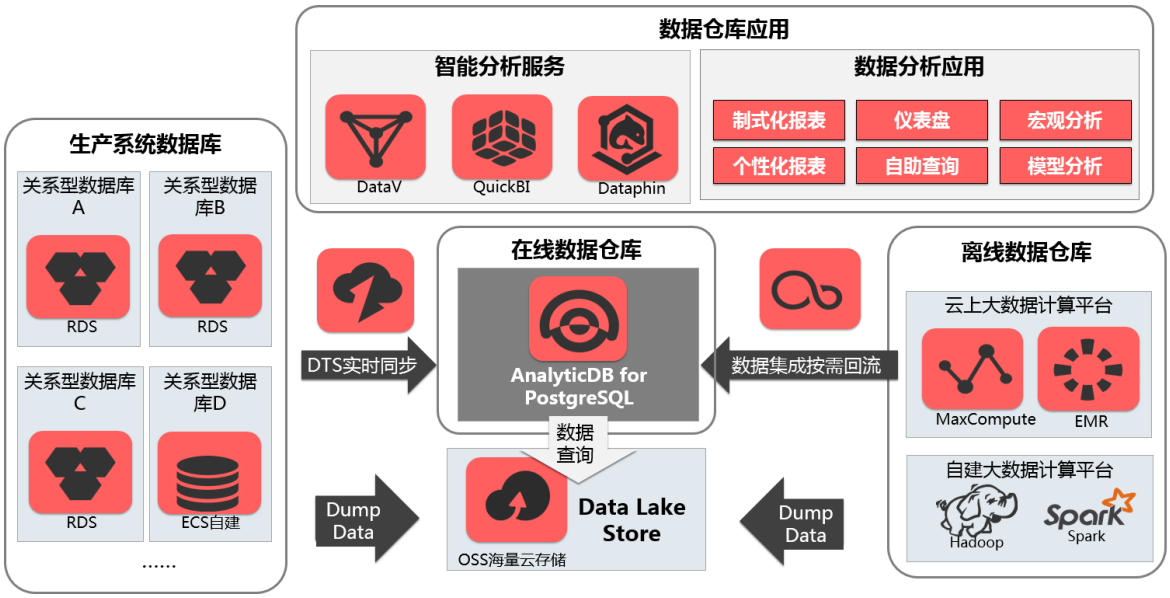
注意事项

AnalyticDB PostgreSQL版基于开源GreenPlum进行了深度的改造和扩展，鉴于AnalyticDB PostgreSQL版团队对GreenPlum的深度理解和维护经验，AnalyticDB PostgreSQL版禁用了部分GreenPlum的功能，例如触发器等，更多使用限制，请参见约束与限制。

2.典型场景

本文将介绍云原生数据仓库AnalyticDB PostgreSQL版的典型场景及产品功能优势。

典型场景



- **数据仓库服务**

  您可以通过数据传输服务（DTS）或数据集成服务（DataX），将云数据库（例如RDS、PolarDB）或自建数据库批量同步到云原生数据仓库AnalyticDB PostgreSQL版。云原生数据仓库PostgreSQL版支持对海量数据的复杂ETL进行处理，这些操作任务也可以被DataWorks调度。同时它还支持高性能的在线分析能力，可以通过Quick BI、DataV、Tableau、帆软等即时查询数据，并将数据以报表形式展现。

- **大数据分析平台**

  对于MaxCompute、Hadoop和Spark中保存的海量数据，可通过采用数据集成服务（DataX）或通过对象存储服务（OSS），快速批量导入到云原生数据仓库AnalyticDB PostgreSQL版，帮助您实现高性能分析处理和在线数据探索。

- **数据湖分析**

  云原生数据仓库AnalyticDB PostgreSQL版可以通过外部表机制，高并行直接访问海量云存储OSS上的数据，构筑阿里云统一数据湖分析平台。

产品功能优势

针对主要的OLAP业务，云原生数据仓库AnalyticDB PostgreSQL版具备以下优势。

- **ETL离线数据处理**

  面对复杂SQL优化和海量数据大规模聚合分析等挑战，云原生数据仓库AnalyticDB PostgreSQL版具有如下技术优势：

  - 支持标准SQL、OLAP窗口函数和存储过程。
  - ORCA分布式SQL优化器，复杂查询免调优。
  - MPP多节点全并行计算，PB级数据秒级响应。

- 基于列存储的高性能大表扫描，极高压缩比。

- **在线高性能查询**

  面对任意维度数据即时探索和数据实时入库更新等挑战，云原生数据仓库AnalyticDB PostgreSQL版具有如下技术优势：

  - 高吞吐数据写入及更新（如INSERT、UPDATE、DELETE）。
  - 行存储及多种索引（如B-tree、Bitmap），点查询毫秒级返回。
  - 支持分布式事务，标准数据库隔离级别，支持HTAP混合负载。

- **多模数据分析**

  面对多种非结构化数据源的挑战，云原生数据仓库AnalyticDB PostgreSQL版具有如下技术优势：

  - 支持PostGIS插件扩展，实现地理数据分析处理。
  - 通过MADlib插件扩展，内置多种机器学习算法，实现AI Native DB。
  - 支持通过向量检索，实现非结构化数据（如图片、语音、文本）的高性能检索分析。
  - 支持JSON等格式，支持日志等半结构化数据处理分析。

3.规格及选型

本文将介绍如何选择云原生数据仓库AnalyticDB PostgreSQL版实例规格。

实例资源类型

AnalyticDB PostgreSQL版推荐使用存储弹性模式和Serverless版本两种实例资源类型：

- 存储弹性模式

  产品功能完整，使用存储计算一体架构，支持计算节点垂直升降配，横向扩容和存储灵活调整。

  购买时需要指定**实例系列**、**计算节点规格**、**计算节点数**、**存储节点类型**和**单节点存储大小**。

- Serverless版本

  自研存储计算分离的架构，支持秒级扩缩容、按需存储，轻松覆盖业务存在明显闲忙的资源需求场景。

  购买时需要指定**实例系列**、**计算节点规格**和**计算节点数**。

具体信息如下表所示：

存储弹性模式（推荐）

实例系列	节点规格	存储磁盘类型	适配场景
高可用	2C16G	ESSD云盘 高效云盘	POC测试。 个人学习使用，体验测试产品能力。
	4C32G		适合计算存储均衡场景，60%用户的选择。
	8C64G		适合计算密集型场景，支持大量高复杂度数据分析，高并发等场景。
	16C128G		适合企业级平台建设，适用于高并发场景，大规模企业核心数据平台推荐选择。
高性能（基础版）	2C8G	ESSD云盘	POC测试。 个人学习使用，体验测试产品能力。
	4C16G		适合计算存储均衡场景，适合离线分析业务。 **注意** 高性能（基础版）不提供高可用，请谨慎选择该系列。
	8C32G		
	16C64G		

Serverless版本（新）

实例系列	节点规格	存储磁盘类型	适配场景
高可用	4C16G	共享存储	新产品形态，支持秒级变配、按需存储及数据共享。 适合如下场景： • 业务负载变动较大。 • 新业务无法明确资源计划。 • 业务负载隔离明确。
	8C32G		

核心能力选型推荐

典型数据入仓场景

场景	存储弹性模式	Serverless版本
RDS	支持	暂不支持
Flink	支持	暂不支持
Kafka	支持	暂不支持
自建数据库（MySQL，PostgreSQL等）	支持	暂不支持
MaxCompute	支持	支持
OSS数据（JSON，AVRO，CSV等）	支持	支持
JDBC、ODBC客户端写入	支持	支持

数据分析场景

场景	存储弹性模式	Serverless版本
标准SQL能力	支持	支持
时空分析场景 PostGIS、GANOS	支持	暂不支持
机器学习场景	支持	支持
数据湖分析（OSS，ODPS）	支持	支持
联邦分析（OSS，ODPS）	支持	支持
大量离线数据批处理	支持	支持
向量分析场景	支持	暂不支持

负载管理场景

场景	存储弹性模式	Serverless版本
资源隔离	支持	支持
多租户分库分仓	支持	支持
多业务实例间关联分析	暂不支持	支持
分时弹性	暂不支持	暂不支持

实例配置选型案例

案例一：互联网用户和制造业用户

用户为互联网客户和制造业客户，目前自建业务数据库和Greenplum数仓，希望能够完成云化部署。

建议：使用AnalyticDB PostgreSQL版存储弹性模式进行部署。

优势：AnalyticDB PostgreSQL版兼容所有Greenplum、PostgreSQL语法和开源生态，可完成能力的全无缝对接并按需进行资源调整。

案例二：互联网SaaS用户

用户为互联网SaaS客户，需要建立数据中台，涉及多数据源包括RDS、Flink、OSS等，期待在平台上完成数据的ETL流程，实现多源处理以及分析侧不同场景的混合负载支持，需要高稳定性保证，同时平台对接数据应用支持报表和企业级数据服务。

建议：使用AnalyticDB PostgreSQL版存储弹性模式， 实例系列为高可用版，计算节点规格为4C32G以上，计算节点数量为4个以上。

优势：存储弹性模式打通了阿里云产品生态，支持导入阿里云或其他云产品进行的数据，提供企业级能力，例如通过UDF或Resource Queue进行负载管理。可实现仓内的ETL全流程，同时提供高于传统数仓系统约3倍的计算性能，支持分析侧的应用变化，支持垂直升降配和扩容，具有较高的平台扩展性。

案例三：传统企业数字化转型

用户为传统企业，需进行数字化转型，替代IDC上的Teradata、Oracle、DB2、传统Greenplum等传统数仓。

建议：使用AnalyticDB PostgreSQL版存储弹性模式，实例系列可根据业务类型选择高可用版或基础版，计算节点规格为4C32G以上，计算节点数量为4个以上。

优势：AnalyticDB PostgreSQL版是替代Teradata、Oracle的标杆产品，已在数百家金融、运营商、政企验证完备的替代能力和解决方案。

案例四：自动驾驶企业

用户为自动驾驶领域企业，需要基于车采数据进行地理位置和时序的采集数据分析，要求对JSON格式的友好兼容和时空数据的分析能力，构建业务看板并支持特征工程。

建议：使用AnalyticDB PostgreSQL版存储弹性模式，实例系列为基础版，计算节点规格为4C32G以上。

优势：存储弹性模式具备PostGIS和Ganos的时空分析引擎，同时可实现在MPP架构下的查询加速；支持JSON等半结构化数据分析；支持数据湖分析，可实现最大程度的数据分析灵活性。

案例五：互联网游戏企业

用户为互联网游戏企业，需要构建数据中台，对行为数据进行分析。平台通过清洗业务日志和数据关联分析，实时支持运营工具。存在工作时间的混合业务负载和资源隔离需求。

建议：使用AnalyticDB PostgreSQL版Serverless版本，计算节点规格为4C16G以上，计算节点数量为4个以上。

优势：Serverless版本可以灵活的根据不同的时段进行资源的调整。对于日志数据提供SLS+OSS成熟的解决方案，能够实现仓内的高效数据清洗。Serverless版本分析能力完备，具备更强的单点计算能力。

案例六：新零售企业

用户为新零售企业，需要构建CDP平台，平台需要完备的多数据源汇入能力，并提供CDP下游人群圈选的成熟解决方案。

建议：使用AnalyticDB PostgreSQL版存储弹性模式，实例系列可根据需求选择高可用版或基础版，计算节点规格为4C32G以上，计算节点数量为4个以上。

优势：存储弹性模式支持多种数据格式，例如JSON、CSV、AVRO、PARQUET等，可实现数据快速汇聚并完成标签生成。支持阿里云自研的Quick Audience等产品，可快速实现云上平台的一站式搭建。

案例七：大型互联网企业

用户为大型互联网企业，各业务线具有各自独立的业务中台，企业存在统一的数据中台，希望可以快速部署独立资源支持不同的业务负载，且未来不会产生数据孤岛。

建议：使用AnalyticDB PostgreSQL版Serverless版本，计算节点规格为4C16G以上，计算节点数量为2个以上，可部署多个实例。

优势：Serverless版本快速的资源部署和弹性的特性规避了对业务的资源部署进行繁重的前期规划，可动态适配业务负载。多个实例间可实现数据共享，无需担心业务中台的发展和数据体系建设会造成数据孤岛；独立的实例可完全保证资源隔离；每个业务的使用情况也可以直接反映在账单上。

案例八：构建数据开发平台

用户需要构建一个数据开发平台，期望减少开发过程中对线上业务的影响同时增加开发效率。

建议：使用AnalyticDB PostgreSQL版Serverless版本，计算节点规格为4C16G以上，计算节点数量为2个以上，可部署多个实例。

优势：生产库使用AnalyticDB PostgreSQL版Serverless版本，在需要进行数据开发时，通过数据共享和测试实例实现生产数据的实时同步可用，同时避免了开发时对生产环境的影响，也可用高时效性的数据进行数据开发。

4.基础版实例

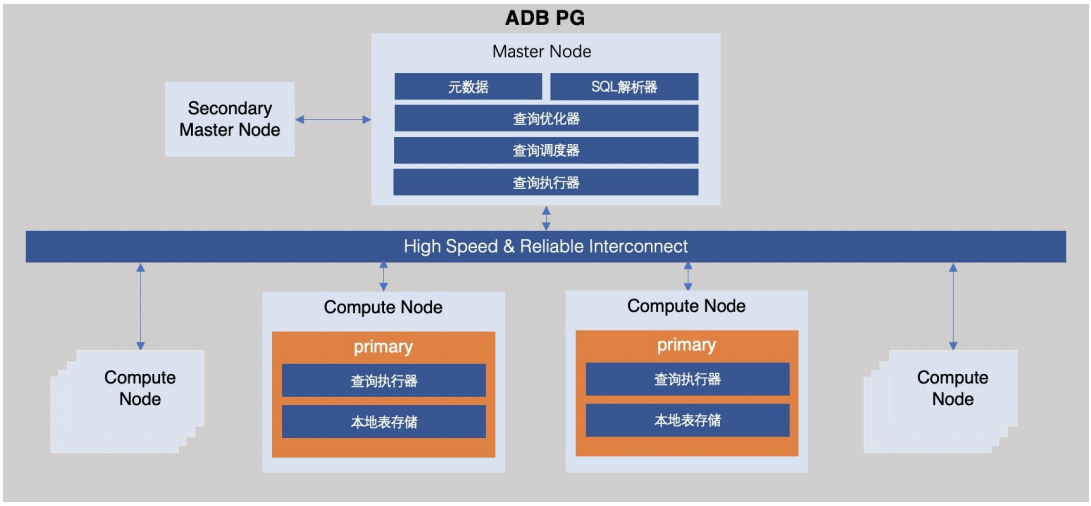
AnalyticDB PostgreSQL版基础版采用单副本存储模式，大幅降低了数据存储成本及建仓门槛，并提供了较高的I/O能力。

> **说明** AnalyticDB PostgreSQL版基础版实例，适用于大部分业务分析场景。对于企业核心业务，依然推荐采用高可用版本。

架构介绍

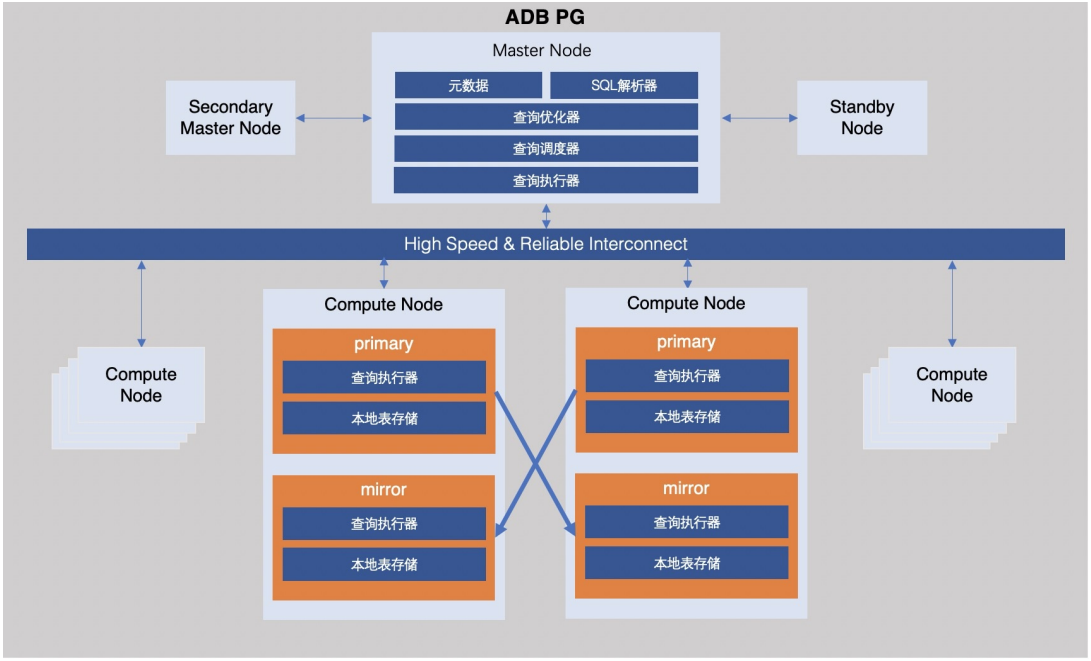
AnalyticDB PostgreSQL版基础版的Master和Segment节点均采用了单节点部署，架构图如下。

基础版架构图



相比较下图中的高可用版，基础版取消了Master Node的副本Standby Node，以及Compute Node中Primary的副本Mirror。

高可用版架构图



取消上述副本后基础版具有如下优势：

- 取消了Master Node的副本Standby Node，节省了Standby Node占用的存储空间。
- 取消了Compute Node中的副本Mirror，节省了一半的存储空间。
- 取消了Compute Node中Primary与Mirror的数据同步过程，提升了写入场景下的I/O性能。

费用说明

价格信息详情，请参见云原生数据仓库PostgreSQL版详细价格信息。

基础版优势

- **成本优势**

  基础版成本优势主要体现在如下两个方面：

  - 相同规格下，节省了一个副本的存储空间，降低了50%的存储成本。
  - 计算节点在拥有相同算力的情况下价格下降。

配置	存储价格			计算节点价格			总价格		
	基础版	高可用版	价格下降	基础版	高可用版	价格下降	基础版	高可用版	价格下降
入门配置	22.4美元/月	100美元/月	77.6%	175.55美元/月	352.05美元/月	50.13%	197.95美元/月	452.05美元/月	56.21%
常用配置s	89.6美元/月	200美元/月	55.2%	668.65美元/月	700.28美元/月	4.52%	758.25美元/月	900.28美元/月	15.78%

  - 入门配置：入门配置即最低配置，基础版为2核、50 GB存储容量、2个计算节点，高可用版为2核、50 GB存储容量、4个计算节点。相比高可用版，基础版的入门价格降低了59%。
  - 常用配置：基础版和高可用版均为4核、100 GB存储容量、4个计算节点。相比高可用版，相同配置下，价格降低了22%。

- **性能优势**

  基础版相比较高可用版，I/O性能有比较明显的提升。2核规格的实例，I/O性能最高可达高可用版相同规格实例的2.5倍。在大量数据写入的场景中，基础版省去了向副本同步数据和流复制的过程，该场景下又有额外的接近1倍的I/O提升。

  对计算节点规格为2核、存储容量为400 GB、4个计算节点的基础版和高可用版实例进行本地复制测试和TPC-H测试：

- 本地复制测试

  对约90 GB的行存表进行本地复制测试，示例SQL如下：

  ```
  create table lineitem2 as (select * from lineitem);
  ```

  基础版和高可用版执行耗时如下：

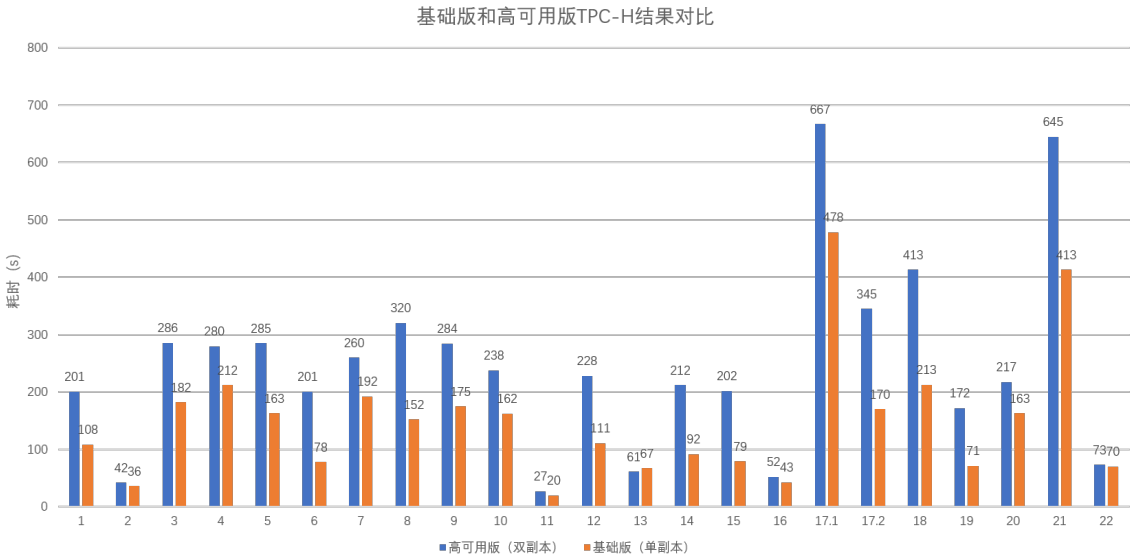
  - 基础版：249s
  - 高可用版：1307s

  通过如上测试可以看出，I/O密集型场景（例如本地表CTAS、INSERT INTO SELECT）性能提升明显，上述示例中有接近5倍性能提升。

- TPC-H测试

  > **说明** 本文的TPC-H的实现基于TPC-H的基准测试，并不能与已发布的TPC-H基准测试结果相比较，本文中的测试并不符合TPC-H基准测试的所有要求。

  对数据集总大小为100 GB的TPC-H数据集进行基准测试，TPC-H的22个SQL结果如下图所示。



  由于I/O性能的提升，相比较高可用版，基础版的TPC-H集群测试用时降低了40%左右。

可用性

- **数据可靠性**

  AnalyticDB PostgreSQL版使用阿里云ESSD云盘作为存储介质，可保证在单副本模式下，依然可以提供超高的数据可靠性。即使计算节点发生故障，也可以保证实例无数据丢失。

- **高可用**

  AnalyticDB PostgreSQL版基础版由于减少了一个副本，在高可用方面出现了一些下降，在物理机故障等极端情况下，集群恢复的时间会变长。基础版通过ESSD多副本技术，保留了完整的数据可靠性，并且阿里云团队通过更改基础版CheckPoint机制的方式，减少了恢复的时间。

  以下内容为AnalyticDB PostgreSQL版实例常见故障场景中基础版和高可用版的对比：

- **恢复（Recovery）模式**

  根据以往AnalyticDB PostgreSQL版运行情况，恢复模式为出现概率最大的故障场景，远大于另外两种场景，该场景下基础版恢复速度远高于高可用版。

  SQL崩溃时，主要为出现Coredump或Out of Memory等情况，会使AnalyticDB PostgreSQL版进入恢复模式。恢复模式中，系统会对残留的锁和内存执行一些清理操作，并通过回放WAL文件来保证数据的完整性。恢复期间，实例会暂时无法服务，完成恢复后，实例会恢复正常。高可用版实例恢复一般耗时5~10分钟，而基础版实例通过更改CheckPoint机制等方式，恢复的时间可缩短至10s左右。

  WAL和CheckPoint介绍如下：

  - WAL（Write Ahead Log）

    AnalyticDB PostgreSQL版中，事务的每次修改数据的操作都需要先记录到WAL文件中，即每次事务提交时，会保证WAL日志已落盘。当数据库需要恢复数据时，可以通过回放WAL日志的方法来恢复已经提交但是尚未写入磁盘的数据库的数据更改。

  - CheckPoint

    CheckPoint相当于在WAL日志中写入的一个恢复点标记，并将该标记之前的修改全部落盘。数据库恢复数据时，只需要回放到最近一次恢复点即可。AnalyticDB PostgreSQL版会定期执行CheckPoint操作，当WAL日志过长时，也会自动执行CheckPoint进行落盘。

- **计算节点故障**

  基础版实例减少了一个副本，必然带来可用性的下降。高可用版的某个计算节点故障后，会立刻无缝切换至对应副本，实例可以正常运行，故障的计算节点会切换为副本，在后台自动重启；而基础版实例单个节点故障会导致整个实例不可用，必须重启整个实例恢复。

- **计算节点宿主机故障**

  计算节点宿主机故障属于比较少见的极端情况，会触发宿主机的自动迁移。对于高可用版实例，仍然可以触发副本自动切换，实例可以正常运行，同时后台自动完成宿主机的迁移；基础版实例则需要等待宿主机迁移成功后，再重启恢复实例，耗时一般在15分钟左右。

相关文档

- 【通知】AnalyticDB PostgreSQL版发布基础版实例
- 创建实例

5.Serverless模式

AnalyticDB PostgreSQL版全新推出Serverless模式，利用云基础设施提供的资源池化和海量存储能力，结合传统MPP数据库技术、离在线一体化技术和Serverless技术，实现了计算存储分离、秒级扩缩容和多实例数据实时共享的特性。

简介

AnalyticDB PostgreSQL版基于云原生架构的Serverless模式完全解耦计算与存储，解决了计算存储必须等比例缩放的问题。赋能用户面向业务峰谷时对计算能力进行快速且独立的扩缩要求，同时保证存储持续按需付费。做到快速响应业务变化的同时，合理优化使用成本，进一步助力企业降本增效。

相比存储弹性模式，Serverless模式具有以下优点：

- 大幅度降低存储成本，实现按需使用。您的历史数据无需再迁移到其他存储介质上，让数据分析更简单、高效、低成本，一站式解决金融、互联网等行业快速增长的数据分析需求。
- 对高吞吐写入场景和高性能跑批业务进行了设计优化，同时提供了弹性伸缩能力，适合业务数据量大、并具有典型的业务访问波峰波谷场景。
- 在存储计算分离基础上，提供了数据共享功能，打破了物理机的边界，让云上的数据流动了起来。 一存多读的使用模式，打破了传统数仓之间数据访问需要先导入再访问的孤岛，简化操作，提高效率，降低成本。

注意事项

> **注意** 目前国际站仅支持创建按量付费的AnalyticDB PostgreSQL版Serverless模式实例。

支持创建AnalyticDB PostgreSQL版Serverless模式的地域及可用区如下：

- 中国：
  - 华北2（北京）：可用区H和可用区I
  - 华东1（杭州）：可用区I和可用区J
  - 华东2（上海）：可用区G和可用区F
  - 华北3（张家口）：可用区C
  - 华南1（深圳）：可用区E和可用区F
- 亚太：

  新加坡：可用区A和可用区C

产品形态对比

Serverless模式作为一个新的形态，兼容存储弹性模式大部分功能，两种模式在产品功能方面的对比如下。

类别	功能	存储弹性模式	Serverless模式
	实例基本信息	支持	支持
	登录数据库（DMS）	支持	支持
	创建实例	支持	支持
	释放实例	支持	支持

类别	功能	存储弹性模式	Serverless模式
实例管理	重启实例	支持	支持
	实例升降配	支持	暂不支持
	扩缩Master节点	支持	支持
	扩容实例	支持	支持
	缩容实例	不支持	支持
	小版本升级	支持	支持
账号管理	创建账号	支持	支持
	重置密码	支持	支持
数据库连接	连接基本信息	支持	支持
	申请外网地址	支持	支持
监控与报警	监控	支持	支持
	报警规则	支持	支持
数据安全	白名单	支持	支持
	SQL审计	支持	支持
	SSL	支持	支持
	备份恢复	支持	支持
配置	参数设置	支持	支持

功能及约束

Serverless模式兼容存储弹性模式95%以上的功能，大多数情况下您可以按照原有语法使用本产品；JDBC接口、ODBC接口以及psql等工具在Serverless模式的使用方法与存储弹性模式一致。您在使用Serverless模式时需要注意部分功能方面的约束，具体信息如下。

类别	功能	约束及说明
	ALTER TABLE	• 支持大部分ALTER TABLE的功能，例如修改表名、删除列约束、增删列等。 • 不支持修改列类型以及修改分布列。
	索引	暂不支持
	PRIMARY KEY	暂不支持
	UNIQUE CONSTRAINT	暂不支持

类别	功能	约束及说明
基本功能	INSERT ON CONFLICT（覆盖写）	暂不支持
	UNLOG表	不支持
	触发器	暂不支持
	HEAP表/AO/AOCS	不支持
	自定义类型	暂不支持
	显式游标	支持
计算引擎	ORCA优化器	支持
	Laser引擎	支持
事务能力	子事务	支持
	事务隔离级别	支持RC和RR隔离级别
高级功能	备份恢复	支持
	物化视图	暂不支持
	AUTO VACUUM	支持
	AUTO ANALYZE	支持
	在线扩容	支持
	在线缩容	支持
	GIS/GANOS	不支持
	数据共享	支持

数据迁移

您可以将现有数据迁移至Serverless模式中，AnalyticDB PostgreSQL版存储弹性模式和存储预留模式迁移至Serverless模式请参见从存储弹性或存储预留实例迁移到Serverless实例。

更多数据迁移支持情况，请参见下表。

迁移类型	文档	是否支持
数据写入	使用INSERT ON CONFLICT覆盖写入数据	暂不支持
	使用COPY ON CONFLICT覆盖导入数据	暂不支持
	基于Client SDK数据写入	支持
	通过DataWorks导入数据	支持

迁移类型	文档	是否支持
表级迁移	通过DTS从云数据库同步数据	暂不支持
	通过DTS从自建数据库同步数据	暂不支持
	通过实时计算Blink写入数据	暂不支持。 您可以通过外表文件中转导入。
	使用\COPY命令导入本地数据	支持
	使用OSS外表高速导入OSS数据	支持
	通过外表在HDFS上读写数据	支持
仓级迁移	自建Greenplum迁移到AnalyticDB PostgreSQL版	暂不支持。 您可以通过外表文件中转导入。
	Teradata应用迁移至AnalyticDB PostgreSQL	暂不支持。 您可以通过外表文件中转导入。
	Amazon Redshift应用和数据迁移至AnalyticDB PostgreSQL	暂不支持。 您可以通过外表文件中转导入。
	Oracle应用迁移至AnalyticDB PostgreSQL	暂不支持。 您可以通过外表文件中转导入。
	从自建Oracle迁移至云原生数据仓库AnalyticDB PostgreSQL	暂不支持。 您可以通过外表文件中转导入。

弹性扩缩容

Serverless模式支持分钟级别的在线弹性扩缩容。实验室测试扩缩容性能如下：

- 16个节点及以内的扩缩容耗时不超过60秒。
- 16个节点以上的扩缩容耗时不超过5分钟。

利用Serverless模式的分钟级别弹性扩缩容能力，您可以在预期的应用访问高峰期到来前（例如双十一购物节），临时将计算节点规模扩大，当应用访问高峰结束后再减少计算节点规模。AnalyticDB PostgreSQL版的计费模块会按照实际应用的时长和规格进行计费（以小时为单位）。通过这种方式可以达到性能和成本的平衡。

目前Serverless模式每个计算节点都拥有相对应的最高存储容量，如果您需要进行缩容操作，请务必保证总数据量不能超过缩容后节点规模的最高存储容量和。例如您的计算节点规格为2C8G，该节点对应的最高存储容量为960 GB，您需要缩容至4个计算节点，则您的总数据量不能超过3840 GB（960 GB*4）。

Serverless模式的不同规格节点对应的最高存储容量分别如下。

规格	最高存储容量
2C8G	960 GB
4C16G	2200 GB
8C32G	5400 GB
16C64G	11800 GB

在扩缩容过程中，只有扩缩容前后会发生临时的闪断，其他时间段业务依然处于可读可写状态，保证了系统的持续可用性。

数据共享（Beta）

相比较传统数仓共享数据使用的数据导入导出方式，Serverless模式的数据共享具有如下优点：

- 存储成本：无需在多个AnalyticDB PostgreSQL版实例间复制或移动数据。分布式存储中仅存放一份数据，不占用额外存储空间，多个实例可在设定的共享范围内访问同一份数据。
- 易用性：只需要简单的创建共享、授权和导入共享操作，即可在数据消费者实例上访问数据，无需处理表结构迁移，可以像访问本地数据一样访问共享数据。共享中增减共享对象和授权变化能自动同步到消费者实例。
- 数据一致性：数据消费者实例对数据的访问性能接近于数据生产者实例，同时消费者实例可以读到生产者实例最新已提交的写入数据，保证事务的ACID能力。

数据共享可以帮助您解决以下问题：

- 复杂组织权限隔离：例如公司总部和分部各有一个实例，总部实例的部分数据需要允许分部的实例进行访问。
- 复杂业务资源隔离：例如ETL和AdHoc业务通过实例实现物理资源隔离，ETL结果通过数据共享给AdHoc的实例。
- 跨业务协作：例如数据研发、销售、运营、财务在需要分析同一份数据时，该数据可以通过数据共享允许组织内不同业务组的访问。

目前数据共享处于测试阶段，使用时存在以下约束：

- 数据共享仅支持普通表，不支持分区表、外表、View、Schema和函数的数据共享。
- 数据共享仅支持Hash分布表，不支持复制表和随机表。
- 数据共享不支持子事务。
- 当源实例中有多个共享时，目标库只能订阅其中一个共享。
- 共享的表无法进行DDL操作，如果需要进行DDL操作，需要取消该表的共享。

相关文档

- 【通知】AnalyticDB PostgreSQL版发布Serverless模式
- 快速入门
- 从存储弹性或存储预留实例迁移到Serverless实例

- Serverless模式导入和查询数据性能测试
- 数据共享

6.Serverless模式性能测试

6.1. Serverless模式导入和查询数据性能测试

本文介绍AnalyticDB PostgreSQL版Serverless模式的导入和查询性能。

> **说明** 本文的TPC-H的实现基于TPC-H的基准测试，并不能与已发布的TPC-H基准测试结果相比较，本文中的测试并不符合TPC-H基准测试的所有要求。

配置信息

本文测试用的Serverless模式实例规格如下：

- 计算节点规格：4C16G
- 计算节点数量：4个

数据导入性能

数据导入测试将使用COPY命令和OSS FDW两种方式导入大表，测试Serverless模式不同并发数情况下的导入性能。

- 测试表：

  本次测试使用了TPC-H的lineitem表，生成的测试数据量为500 GB。如何生成测试数据，请参见生成测试数据。

- 测试方法：
  - COPY：具体操作，请参见COPY。
  - OSS FDW：具体操作，请参见使用OSS Foreign Table访问OSS数据。

测试结果如下：

测试方式	并发数为1	并发数为4	并发数为8
COPY	37 MB/s	125 MB/s	128 MB/s
OSS FDW	47 MB/s	86 MB/s	110 MB/s

数据查询性能

数据查询性能测试将通过TPC-H的qgen工具，生成10 GB和500 GB的数据，测试Serverless模式的查询耗时并与存储弹性模式的查询耗时进行对比。具体测试操作，请参见TPC-H。

10 GB数据查询性能测试结果如下：

查询SQL	存储弹性模式	Serverless模式
Q1	15215.417毫秒	8468.049毫秒
Q2	2949.254毫秒	3874.710毫秒

查询SQL	存储弹性模式	Serverless模式
Q3	3979.300毫秒	2652.187毫秒
Q4	6059.405毫秒	2561.089毫秒
Q5	6833.062毫秒	4297.496毫秒
Q6	482.411毫秒	578.026毫秒
Q7	6228.587毫秒	4301.195毫秒
Q8	6544.251毫秒	5011.280毫秒
Q9	11240.953毫秒	7742.912毫秒
Q10	3549.456毫秒	2767.839毫秒
Q11	1361.575毫秒	1488.599毫秒
Q12	1661.359毫秒	1842.725毫秒
Q13	5383.167毫秒	5018.539毫秒
Q14	744.585毫秒	751.640毫秒
Q15	1344.129毫秒	1897.243毫秒
Q16	1550.342毫秒	1984.808毫秒
Q17	19425.750毫秒	15709.382毫秒
Q18	19417.051毫秒	6803.475毫秒
Q19	4762.443毫秒	2375.202毫秒
Q20	3434.726毫秒	3485.165毫秒
Q21	14496.656毫秒	8104.987毫秒
Q22	3174.644毫秒	2918.874毫秒
总时间	2分钟19.951秒	1分钟34.748秒

500 GB数据查询性能测试结果如下：

查询SQL	存储弹性模式	Serverless模式
Q1	776749.919毫秒	655198.377毫秒
Q2	127436.833毫秒	87954.528毫秒
Q3	323528.962毫秒	664481.555毫秒

查询SQL	存储弹性模式	Serverless模式
Q4	351981.303毫秒	200034.509毫秒
Q5	427701.721毫秒	609339.053毫秒
Q6	110562.730毫秒	19149.394毫秒
Q7	675657.163毫秒	305690.833毫秒
Q8	516443.454毫秒	1033242.151毫秒
Q9	1531569.731毫秒	999391.734毫秒
Q10	295668.016毫秒	141176.254毫秒
Q11	141573.826毫秒	74402.558毫秒
Q12	249247.709毫秒	88836.774毫秒
Q13	315628.505毫秒	177885.452毫秒
Q14	187791.651毫秒	39034.109毫秒
Q15	460263.848毫秒	82863.306毫秒
Q16	123408.319毫秒	54713.206毫秒
Q17	4650424.484毫秒	2215070.817毫秒
Q18	1151063.573毫秒	548049.730毫秒
Q19	260702.969毫秒	85419.149毫秒
Q20	549780.389毫秒	213492.958毫秒
Q21	1103378.860毫秒	456781.416毫秒
Q22	223275.303毫秒	86325.201毫秒
总时间	242分钟34.602秒	147分钟19.298秒

6.2. Serverless模式批量删除或更新数据性能测试

本文介绍AnalyticDB PostgreSQL版Serverless模式实例批量DELETE和UPDATE的性能。

> **说明** 本文的TPC-H的实现基于TPC-H的基准测试，并不能与已发布的TPC-H基准测试结果相比较，本文中的测试并不符合TPC-H基准测试的所有要求。

测试实例

创建AnalyticDB PostgreSQL版实例的具体操作，请参见创建实例。

实例规格信息如下：

- Serverless模式实例
  - Segment节点规格：4C16G
  - Segment节点数量：4个
- 存储弹性模式实例
  - Segment节点规格：2C16G
  - Segment节点数量：4个

测试数据

生成并导入测试数据的具体操作，请参见TPC-H。

测试数据具体信息如下：

- 测试数据量：100 GB。
- 测试数据表：lineitem、orders。

  lineitem建表语法如下：

```
CREATE TABLE lineitem (
    l_orderkey bigint NOT NULL,
    l_partkey integer NOT NULL,
    l_suppkey integer NOT NULL,
    l_linenumber integer NOT NULL,
    l_quantity numeric(15,2) NOT NULL,
    l_extendedprice numeric(15,2) NOT NULL,
    l_discount numeric(15,2) NOT NULL,
    l_tax numeric(15,2) NOT NULL,
    l_returnflag character(1) NOT NULL,
    l_linestatus character(1) NOT NULL,
    l_shipdate date NOT NULL,
    l_commitdate date NOT NULL,
    l_receiptdate date NOT NULL,
    l_shipinstruct character(25) NOT NULL,
    l_shipmode char(10) NOT NULL,
    l_comment varchar(44) NOT NULL
)
distributed by (l_orderkey) order by (l_shipdate,l_orderkey);
```

  orders建表语法如下：

```
CREATE TABLE orders (
    o_orderkey bigint NOT NULL,
    o_custkey integer NOT NULL,
    o_orderstatus character(1) NOT NULL,
    o_totalprice numeric(15,2) NOT NULL,
    o_orderdate date NOT NULL,
    o_orderpriority character(15) NOT NULL,
    o_clerk character(15) NOT NULL,
    o_shippriority integer NOT NULL,
    o_comment character varying(79) NOT NULL
)
distributed by (o_orderkey) order by (o_orderdate, o_orderkey);
```

- 测试数据行数：lineitem表6亿行，orders表1.5亿行，合计7.5亿行。

  lineitem和orders表均根据时间列按天均匀分布数据，lineitem表每天约有25万行数据；orders表每天约有6万行数据。因此示例语句中的时间范围可以决定查询的数据量，本次性能测试的时间范围和数据量如下：

  - 一天，约有31万行数据。
  - 半个月，约有466万行数据。
  - 一个月，约有965万行数据。

测试语句

测试语句（DELETE和UPDATE）将根据lineitem表和orders表的时间列进行批量操作，测试语句示例如下：

- DELETE语句
  - lineitem表

    ```
    DELETE FROM lineitem WHERE l_shipdate >= YYYYMMDD AND l_shipdate <= YYYYMMDD;
    ```

  - orders表

    ```
    DELETE FROM orders WHERE o_orderdate >= YYYYMMDD AND o_orderdate <= YYYYMMDD;
    ```

- UPDATE语句
  - lineitem表

    ```
    UPDATE lineitem SET l_quantity = 100.00, l_shipmode = 'test', l_comment = 'only_for_tes
    t'
    WHERE l_shipdate >= YYYYMMDD AND l_shipdate <= YYYYMMDD;
    ```

  - orders表

    ```
    UPDATE orders SET o_totalprice = 100.00, o_shippriority = 10, o_comment = 'only_for_tes
    t'
    WHERE o_orderdate >= YYYYMMDD AND o_orderdate <= YYYYMMDD;
    ```

- SELECT语句

  每次进行DELETE或UPDATE操作后，执行 `SELECT count(1)` 语句查看数据变更后的SELECT性能。

  - lineitem表

    ```
    SELECT count(1) FROM lineitem;
    ```

- orders表

```
SELECT count(1) FROM orders;
```

> **说明** 以上示例语句中的 `YYYYMMDD` 请替换为需要查询的日期，例如 `'1995-05-05'` 。

测试结果

未进行DELETE和UPDATE操作前，在lineitem表和orders表中执行 `SELECT count(1)` 语句。测试结果如下：

表名	Serverless模式	存储弹性模式
lineitem	40.3s	177.9s
orders	8.0s	40.5s

在Serverless模式实例和存储弹性模式实例中并发对lineitem表和orders表执行DELETE和UPDATE操作，测试三遍，取平均值。测试结果如下：

批量操作	时间范围	Serverless模式（lineitem表与orders表的执行总耗时）	存储弹性模式（lineitem表与orders表的执行总耗时）	Serverless模式执行SELECT	存储弹性模式执行SELECT
DELETE	一天	32s	290s	lineitem表：40.7s orders表：8.2s	lineitem表：214.3s orders表：56.8s
	半个月	38s	827s	lineitem表：41.6s orders表：8.5s	lineitem表：657.1s orders表：170.0s
	一个月	40s	994s	lineitem表：41.7s orders表：8.3s	lineitem表：788.9s orders表：197.3s
UPDATE	一天	104s	300s	lineitem表：41.6s orders表：8.5s	lineitem表：206.6s orders表：57.2s
	半个月	110s	824s	lineitem表：42.7s orders表：8.4s	lineitem表：650.0s orders表：172.4s
	一个月	114s	988s	lineitem表：42.9s orders表：8.5s	lineitem表：784.1s orders表：201.8s

通过以上测试结果可以得出以下信息：

- 批量执行DELETE和UPDATE时，Serverless模式实例比存储弹性模式实例执行速度快3~20倍，批量执行的数据量越大，性能差异越明显。

  原因：批量执行DELETE和UPDATE是根据ORDER BY进行筛选的，Serverless模式实例的表采用了行列混存加ORDER BY的存储模式，对单列的过滤效率很高；而存储弹性模式实例的表在对列进行过滤时，需要读取所有数据，存在读放大的问题，且该列的过滤不会进行二分查找，所以性能相比Serverless模式较差。

- Serverless模式实例执行 `SELECT count(1)` 比存储弹性模式实例执行 `SELECT count(1)` 速度快4~5倍。

  原因：Serverless模式实例的表的 `count(1)` 会按照列快速进行count，而存储弹性模式实例的表需要读取所有数据，存在读放大的问题。

- Serverless模式实例批量执行DELETE和UPDATE操作后再进行 `SELECT count(1)` ，性能方面几乎无差异。

  原因：Serverless模式实例的表按照列快速进行count并从delete bitmap中快速进行数据删除校验，不存在读放大的问题。

6.3. Serverless模式数据共享性能测试

本文将对AnalyticDB PostgreSQL版Serverless模式数据共享源端和目标端的查询性能进行测试。

> **说明** 本文的TPC-H的实现基于TPC-H的基准测试，并不能与已发布的TPC-H基准测试结果相比较，本文中的测试并不符合TPC-H基准测试的所有要求。

测试说明

数据共享源端和目标端实例规格一致，实例具体信息如下：

- Segment节点规格：4C16G
- Segment节点数量：4个
- 地域及可用区：新加坡（可用区A）
- 内核版本：V1.0.1.0

创建实例的具体操作，请参见创建实例。

本次测试将使用dbgen工具生成100 GB原始数据，如何安装dbgen并导入数据，请参见生成测试数据。

测试步骤

1. 将源端实例和目标端实例加入数据共享，具体操作，请参见开启实例的数据共享。
2. 连接源端实例，进行以下操作：
   i. 连接实例，具体操作，请参见客户端连接。

   > **说明** 本次性能测试使用的客户端为psql。

   ii. 创建一个名为db01的数据库，并切换到db01数据库，语句如下：

   ```
   CREATE DATABASE db01;
   \c db01
   ```

iii. 查询db01的UUID，语句如下：

```
SELECT current_database_uuid();
```

iv. 创建一个名为tpch的Schema，并将其设置为默认Schema，语句如下：

```
CREATE SCHEMA IF NOT EXISTS tpch;
SET search_path = tpch;
```

v. 创建TPC-H的八张测试表，语句如下：

```
CREATE TABLE customer (
    c_custkey integer NOT NULL,
    c_name character varying(25) NOT NULL,
    c_address character varying(40) NOT NULL,
    c_nationkey integer NOT NULL,
    c_phone character(15) NOT NULL,
    c_acctbal numeric(15,2) NOT NULL,
    c_mktsegment character(10) NOT NULL,
    c_comment character varying(117) NOT NULL
)
distributed by (c_custkey);
CREATE TABLE lineitem (
    l_orderkey bigint NOT NULL,
    l_partkey integer NOT NULL,
    l_suppkey integer NOT NULL,
    l_linenumber integer NOT NULL,
    l_quantity numeric(15,2) NOT NULL,
    l_extendedprice numeric(15,2) NOT NULL,
    l_discount numeric(15,2) NOT NULL,
    l_tax numeric(15,2) NOT NULL,
    l_returnflag character(1) NOT NULL,
    l_linestatus character(1) NOT NULL,
    l_shipdate date NOT NULL,
    l_commitdate date NOT NULL,
    l_receiptdate date NOT NULL,
    l_shipinstruct character(25) NOT NULL,
    l_shipmode char(10) NOT NULL,
    l_comment varchar(44) NOT NULL
)
distributed by (l_orderkey);
CREATE TABLE nation (
    n_nationkey integer NOT NULL,
    n_name character(25) NOT NULL,
    n_regionkey integer NOT NULL,
    n_comment character varying(152)
)
distributed by (n_nationkey);
CREATE TABLE orders (
    o_orderkey bigint NOT NULL,
    o_custkey integer NOT NULL,
    o_orderstatus character(1) NOT NULL,
    o_totalprice numeric(15,2) NOT NULL,
    o_orderdate date NOT NULL,
    o_orderpriority character(15) NOT NULL,
```

```
    o_clerk character(15) NOT NULL,
    o_shippriority integer NOT NULL,
    o_comment character varying(79) NOT NULL
)
distributed by (o_orderkey);
CREATE TABLE part (
    p_partkey integer NOT NULL,
    p_name character varying(55) NOT NULL,
    p_mfgr character(25) NOT NULL,
    p_brand character(10) NOT NULL,
    p_type character varying(25) NOT NULL,
    p_size integer NOT NULL,
    p_container character(10) NOT NULL,
    p_retailprice numeric(15,2) NOT NULL,
    p_comment character varying(23) NOT NULL
)
distributed by (p_partkey);
CREATE TABLE partsupp (
    ps_partkey integer NOT NULL,
    ps_suppkey integer NOT NULL,
    ps_availqty integer NOT NULL,
    ps_supplycost numeric(15,2) NOT NULL,
    ps_comment character varying(199) NOT NULL
)
distributed by (ps_partkey);
CREATE TABLE region (
    r_regionkey integer NOT NULL,
    r_name character(25) NOT NULL,
    r_comment character varying(152)
)
distributed by (r_regionkey);
CREATE TABLE supplier (
    s_suppkey integer NOT NULL,
    s_name character(25) NOT NULL,
    s_address character varying(40) NOT NULL,
    s_nationkey integer NOT NULL,
    s_phone character(15) NOT NULL,
    s_acctbal numeric(15,2) NOT NULL,
    s_comment character varying(101) NOT NULL
)
distributed by (s_suppkey);
```

vi. 导入测试数据。您可以通过OSS外表或\COPY命令将数据导入到AnalyticDB PostgreSQL版Serverless模式，具体操作，请参见使用OSS外表高速导入OSS数据或使用\COPY命令导入本地数据。

以下示例语句为\COPY方式导入数据，请将 `'/path/to/localfile'` 替换为您测试数据所在的真实路径：

```
\COPY customer FROM '/path/to/localfile';
\COPY lineitem FROM '/path/to/localfile';
\COPY nation   FROM '/path/to/localfile';
\COPY orders   FROM '/path/to/localfile';
\COPY part     FROM '/path/to/localfile';
\COPY partsupp FROM '/path/to/localfile';
\COPY region   FROM '/path/to/localfile';
\COPY supplier FROM '/path/to/localfile';
```

3. 连接目标端实例，进行以下操作：

i. 连接实例，具体操作，请参见客户端连接。

ii. 创建一个名为db02的数据库，并切换到db02数据库，语句如下：

```
CREATE DATABASE db02;
\c db02
```

iii. 查询db02的UUID，语句如下：

```
SELECT current_database_uuid();
```

4. 在源端实例上创建datashare，将测试表加入数据共享，并授权给db02，具体操作如下：

i. 创建datashare，语句如下：

```
CREATE DATASHARE s01;
```

ii. 将八张测试表加入数据共享，语句如下：

```
ALTER DATASHARE s01 ADD TABLE tpch_col.supplier;
ALTER DATASHARE s01 ADD TABLE tpch_col.region;
ALTER DATASHARE s01 ADD TABLE tpch_col.partsupp;
ALTER DATASHARE s01 ADD TABLE tpch_col.part;
ALTER DATASHARE s01 ADD TABLE tpch_col.orders;
ALTER DATASHARE s01 ADD TABLE tpch_col.nation;
ALTER DATASHARE s01 ADD TABLE tpch_col.lineitem;
ALTER DATASHARE s01 ADD TABLE tpch_col.customer;
```

iii. 将datashare授权给目标端的db02，语句如下：

```
GRANT USAGE ON DATASHARE s01 TO DATABASE "db02-uuid";
```

> **说明** 请将 `"db02-uuid"` 替换为步骤二中实际获取到db02的UUID。

5. 在目标端实例中导入数据共享s01，并执行ANALYZE收集统计信息，具体步骤如下：

i. 导入数据共享s01，语句如下：

```
IMPORT DATASHARE s01 AS s01a FROM DATABASE "db01-uuid";
```

> **说明** 请将 `"db01-uuid"` 替换为步骤一中实际获取到db01的UUID。

ii. 对数据共享的八张表进行ANALYZE，语句如下：

```
ANALYZE customer;
ANALYZE lineitem;
ANALYZE nation;
ANALYZE orders;
ANALYZE part;
ANALYZE partsupp;
ANALYZE region;
ANALYZE supplier;
```

6. 执行TPC-H的22条查询。

> **说明** 进行测试前请将optimizer参数设置为off，如何修改配置参数，请参见参数配置。

```
-- Q1
select
    l_returnflag,
    l_linestatus,
    sum(l_quantity) as sum_qty,
    sum(l_extendedprice) as sum_base_price,
    sum(l_extendedprice * (1 - l_discount)) as sum_disc_price,
    sum(l_extendedprice * (1 - l_discount) * (1 + l_tax)) as sum_charge,
    avg(l_quantity) as avg_qty,
    avg(l_extendedprice) as avg_price,
    avg(l_discount) as avg_disc,
    count(*) as count_order
from
    lineitem
where
    l_shipdate <= date '1998-12-01' - interval '93 day'
group by
    l_returnflag,
    l_linestatus
order by
    l_returnflag,
    l_linestatus;
-- Q2
select
    s_acctbal,
    s_name,
    n_name,
    p_partkey,
    p_mfgr,
    s_address,
    s_phone,
    s_comment
```

```
    s_comment
from
    part,
    supplier,
    partsupp,
    nation,
    region
where
    p_partkey = ps_partkey
    and s_suppkey = ps_suppkey
    and p_size = 23
    and p_type like '%STEEL'
    and s_nationkey = n_nationkey
    and n_regionkey = r_regionkey
    and r_name = 'EUROPE'
    and ps_supplycost = (
        select
            min(ps_supplycost)
        from
            partsupp,
            supplier,
            nation,
            region
        where
            p_partkey = ps_partkey
            and s_suppkey = ps_suppkey
            and s_nationkey = n_nationkey
            and n_regionkey = r_regionkey
            and r_name = 'EUROPE'
    )
order by
    s_acctbal desc,
    n_name,
    s_name,
    p_partkey
limit 100;
-- Q3
select
    l_orderkey,
    sum(l_extendedprice * (1 - l_discount)) as revenue,
    o_orderdate,
    o_shippriority
from
    customer,
    orders,
    lineitem
where
    c_mktsegment = 'MACHINERY'
    and c_custkey = o_custkey
    and l_orderkey = o_orderkey
    and o_orderdate < date '1995-03-24'
    and l_shipdate > date '1995-03-24'
group by
    l_orderkey,
    o_orderdate,
```

```
    o_shippriority
order by
    revenue desc,
    o_orderdate
limit 10;
-- Q4
select
    o_orderpriority,
    count(*) as order_count
from
    orders
where
    o_orderdate >= date '1996-08-01'
    and o_orderdate < date '1996-08-01' + interval '3' month
    and exists (
        select
            *
        from
            lineitem
        where
            l_orderkey = o_orderkey
            and l_commitdate < l_receiptdate
    )
group by
    o_orderpriority
order by
    o_orderpriority;
-- Q6
select
    sum(l_extendedprice * l_discount) as revenue
from
    lineitem
where
    l_shipdate >= date '1994-01-01'
    and l_shipdate < date '1994-01-01' + interval '1' year
    and l_discount between 0.06 - 0.01 and 0.06 + 0.01
    and l_quantity < 24;
-- Q7
select
    supp_nation,
    cust_nation,
    l_year,
    sum(volume) as revenue
from
    (
        select
            n1.n_name as supp_nation,
            n2.n_name as cust_nation,
            extract(year from l_shipdate) as l_year,
            l_extendedprice * (1 - l_discount) as volume
        from
            supplier,
            lineitem,
            orders,
```

```
            customer,
            nation n1,
            nation n2
        where
            s_suppkey = l_suppkey
            and o_orderkey = l_orderkey
            and c_custkey = o_custkey
            and s_nationkey = n1.n_nationkey
            and c_nationkey = n2.n_nationkey
            and (
                (n1.n_name = 'JORDAN' and n2.n_name = 'INDONESIA')
                or (n1.n_name = 'INDONESIA' and n2.n_name = 'JORDAN')
            )
            and l_shipdate between date '1995-01-01' and date '1996-12-31'
    ) as shipping
group by
    supp_nation,
    cust_nation,
    l_year
order by
    supp_nation,
    cust_nation,
    l_year;
-- Q8
select
    o_year,
    sum(case
        when nation = 'INDONESIA' then volume
        else 0
    end) / sum(volume) as mkt_share
from
    (
        select
            extract(year from o_orderdate) as o_year,
            l_extendedprice * (1 - l_discount) as volume,
            n2.n_name as nation
        from
            part,
            supplier,
            lineitem,
            orders,
            customer,
            nation n1,
            nation n2,
            region
        where
            p_partkey = l_partkey
            and s_suppkey = l_suppkey
            and l_orderkey = o_orderkey
            and o_custkey = c_custkey
            and c_nationkey = n1.n_nationkey
            and n1.n_regionkey = r_regionkey
            and r_name = 'ASIA'
            and s_nationkey = n2.n_nationkey
```

```
            and o_orderdate between date '1995-01-01' and date '1996-12-31'
            and p_type = 'STANDARD BRUSHED BRASS'
    ) as all_nations
group by
    o_year
order by
    o_year;
-- Q9
select
    nation,
    o_year,
    sum(amount) as sum_profit
from
    (
        select
            n_name as nation,
            extract(year from o_orderdate) as o_year,
            l_extendedprice * (1 - l_discount) - ps_supplycost * l_quantity as amount
        from
            part,
            supplier,
            lineitem,
            partsupp,
            orders,
            nation
        where
            s_suppkey = l_suppkey
            and ps_suppkey = l_suppkey
            and ps_partkey = l_partkey
            and p_partkey = l_partkey
            and o_orderkey = l_orderkey
            and s_nationkey = n_nationkey
            and p_name like '%chartreuse%'
    ) as profit
group by
    nation,
    o_year
order by
    nation,
    o_year desc;
-- Q10
select
    c_custkey,
    c_name,
    sum(l_extendedprice * (1 - l_discount)) as revenue,
    c_acctbal,
    n_name,
    c_address,
    c_phone,
    c_comment
from
    customer,
    orders,
    lineitem,
```

```
    nation
where
    c_custkey = o_custkey
    and l_orderkey = o_orderkey
    and o_orderdate >= date '1994-08-01'
    and o_orderdate < date '1994-08-01' + interval '3' month
    and l_returnflag = 'R'
    and c_nationkey = n_nationkey
group by
    c_custkey,
    c_name,
    c_acctbal,
    c_phone,
    n_name,
    c_address,
    c_comment
order by
    revenue desc
limit 20;
-- Q11
select
    ps_partkey,
    sum(ps_supplycost * ps_availqty) as value
from
    partsupp,
    supplier,
    nation
where
    ps_suppkey = s_suppkey
    and s_nationkey = n_nationkey
    and n_name = 'INDONESIA'
group by
    ps_partkey having
        sum(ps_supplycost * ps_availqty) > (
            select
                sum(ps_supplycost * ps_availqty) * 0.0001000000
            from
                partsupp,
                supplier,
                nation
            where
                ps_suppkey = s_suppkey
                and s_nationkey = n_nationkey
                and n_name = 'INDONESIA'
        )
order by
    value desc;
-- Q12
select
    l_shipmode,
    sum(case
        when o_orderpriority = '1-URGENT'
            or o_orderpriority = '2-HIGH'
            then 1
        else 0
```

```
        else 0
    end) as high_line_count,
    sum(case
        when o_orderpriority <> '1-URGENT'
            and o_orderpriority <> '2-HIGH'
            then 1
        else 0
    end) as low_line_count
from
    orders,
    lineitem
where
    o_orderkey = l_orderkey
    and l_shipmode in ('REG AIR', 'TRUCK')
    and l_commitdate < l_receiptdate
    and l_shipdate < l_commitdate
    and l_receiptdate >= date '1994-01-01'
    and l_receiptdate < date '1994-01-01' + interval '1' year
group by
    l_shipmode
order by
    l_shipmode;
-- Q13
select
    c_count,
    count(*) as custdist
from
    (
        select
            c_custkey,
            count(o_orderkey)
        from
            customer left outer join orders on
                c_custkey = o_custkey
                and o_comment not like '%pending%requests%'
        group by
            c_custkey
    ) as c_orders (c_custkey, c_count)
group by
    c_count
order by
    custdist desc,
    c_count desc;
-- Q14
select
    100.00 * sum(case
        when p_type like 'PROMO%'
            then l_extendedprice * (1 - l_discount)
        else 0
    end) / sum(l_extendedprice * (1 - l_discount)) as promo_revenue
from
    lineitem,
    part
where
    l_partkey = p_partkey
```

```
    and l_shipdate >= date '1994-11-01'
    and l_shipdate < date '1994-11-01' + interval '1' month;
-- Q15
create view revenue0 (supplier_no, total_revenue) as
    select
        l_suppkey,
        sum(l_extendedprice * (1 - l_discount))
    from
        lineitem
    where
        l_shipdate >= date '1997-10-01'
        and l_shipdate < date '1997-10-01' + interval '3' month
    group by
        l_suppkey;
select
    s_suppkey,
    s_name,
    s_address,
    s_phone,
    total_revenue
from
    supplier,
    revenue0
where
    s_suppkey = supplier_no
    and total_revenue = (
        select
            max(total_revenue)
        from
            revenue0
    )
order by
    s_suppkey;
drop view revenue0;
-- Q16
select
    p_brand,
    p_type,
    p_size,
    count(distinct ps_suppkey) as supplier_cnt
from
    partsupp,
    part
where
    p_partkey = ps_partkey
    and p_brand <> 'Brand#44'
    and p_type not like 'SMALL BURNISHED%'
    and p_size in (36, 27, 34, 45, 11, 6, 25, 16)
    and ps_suppkey not in (
        select
            s_suppkey
        from
            supplier
        where
```

```
            s_comment like '%Customer%Complaints%'
    )
group by
    p_brand,
    p_type,
    p_size
order by
    supplier_cnt desc,
    p_brand,
    p_type,
    p_size;
-- Q17
select
    sum(l_extendedprice) / 7.0 as avg_yearly
from
    lineitem,
    part
where
    p_partkey = l_partkey
    and p_brand = 'Brand#42'
    and p_container = 'JUMBO PACK'
    and l_quantity < (
        select
            0.2 * avg(l_quantity)
        from
            lineitem
        where
            l_partkey = p_partkey
    );
-- Q18
select
    c_name,
    c_custkey,
    o_orderkey,
    o_orderdate,
    o_totalprice,
    sum(l_quantity)
from
    customer,
    orders,
    lineitem
where
    o_orderkey in (
        select
            l_orderkey
        from
            lineitem
        group by
            l_orderkey having
                sum(l_quantity) > 312
    )
    and c_custkey = o_custkey
    and o_orderkey = l_orderkey
group by
```

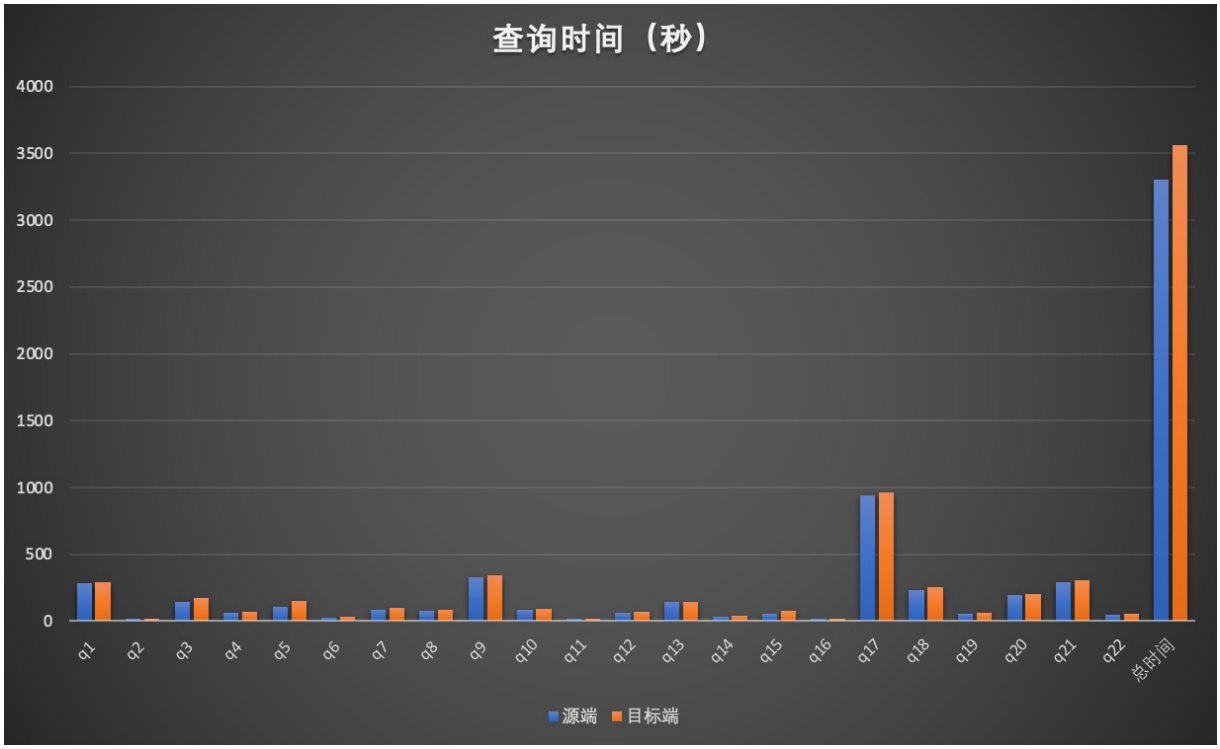
```
    c_name,
    c_custkey,
    o_orderkey,
    o_orderdate,
    o_totalprice
order by
    o_totalprice desc,
    o_orderdate
limit 100;
-- Q19
select
    sum(l_extendedprice* (1 - l_discount)) as revenue
from
    lineitem,
    part
where
    (
        p_partkey = l_partkey
        and p_brand = 'Brand#43'
        and p_container in ('SM CASE', 'SM BOX', 'SM PACK', 'SM PKG')
        and l_quantity >= 5 and l_quantity <= 5 + 10
        and p_size between 1 and 5
        and l_shipmode in ('AIR', 'AIR REG')
        and l_shipinstruct = 'DELIVER IN PERSON'
    )
    or
    (
        p_partkey = l_partkey
        and p_brand = 'Brand#45'
        and p_container in ('MED BAG', 'MED BOX', 'MED PKG', 'MED PACK')
        and l_quantity >= 12 and l_quantity <= 12 + 10
        and p_size between 1 and 10
        and l_shipmode in ('AIR', 'AIR REG')
        and l_shipinstruct = 'DELIVER IN PERSON'
    )
    or
    (
        p_partkey = l_partkey
        and p_brand = 'Brand#11'
        and p_container in ('LG CASE', 'LG BOX', 'LG PACK', 'LG PKG')
        and l_quantity >= 24 and l_quantity <= 24 + 10
        and p_size between 1 and 15
        and l_shipmode in ('AIR', 'AIR REG')
        and l_shipinstruct = 'DELIVER IN PERSON'
    );
-- Q20
select
    s_name,
    s_address
from
    supplier,
    nation
where
    s_suppkey in (
```

```
        select
            ps_suppkey
        from
            partsupp
        where
            ps_partkey in (
                select
                    p_partkey
                from
                    part
                where
                    p_name like 'magenta%'
            )
            and ps_availqty > (
                select
                    0.5 * sum(l_quantity)
                from
                    lineitem
                where
                    l_partkey = ps_partkey
                    and l_suppkey = ps_suppkey
                    and l_shipdate >= date '1996-01-01'
                    and l_shipdate < date '1996-01-01' + interval '1' year
            )
    )
    and s_nationkey = n_nationkey
    and n_name = 'RUSSIA'
order by
    s_name;
-- Q21
select
    s_name,
    count(*) as numwait
from
    supplier,
    lineitem l1,
    orders,
    nation
where
    s_suppkey = l1.l_suppkey
    and o_orderkey = l1.l_orderkey
    and o_orderstatus = 'F'
    and l1.l_receiptdate > l1.l_commitdate
    and exists (
        select
            *
        from
            lineitem l2
        where
            l2.l_orderkey = l1.l_orderkey
            and l2.l_suppkey <> l1.l_suppkey
    )
    and not exists (
        select
            *
```

```
        from
            lineitem l3
        where
            l3.l_orderkey = l1.l_orderkey
            and l3.l_suppkey <> l1.l_suppkey
            and l3.l_receiptdate > l3.l_commitdate
    )
    and s_nationkey = n_nationkey
    and n_name = 'MOZAMBIQUE'
group by
    s_name
order by
    numwait desc,
    s_name
limit 100;
-- Q22
select
        cntrycode,
        count(*) as numcust,
        sum(c_acctbal) as totacctbal
from
        (
                select
                        substring(c_phone from 1 for 2) as cntrycode,
                        c_acctbal
                from
                        customer
                where
                        substring(c_phone from 1 for 2) in
                                ('13', '31', '23', '29', '30', '18', '17')
                        and c_acctbal > (
                                select
                                        avg(c_acctbal)
                                from
                                        customer
                                where
                                        c_acctbal > 0.00
                                        and substring(c_phone from 1 for 2) in
                                                ('13', '31', '23', '29', '30', '18', '1
7')
                        )
                        and not exists (
                                select
                                        *
                                from
                                        orders
                                where
                                        o_custkey = c_custkey
                        )
        ) as custsale
group by
        cntrycode
order by
        cntrycode;
```

测试结果

查询	源端查询耗时（单位：秒）	目标端查询耗时（单位：秒）
Q1	287.04	291.46
Q2	18.49	20.14
Q3	143.08	169.46
Q4	61.54	72.78
Q5	105.46	152.77
Q6	23.78	32.56
Q7	84.42	96.63
Q8	77.01	87.33
Q9	329.42	340.8
Q10	81.72	89.85
Q11	18.18	18.24
Q12	62.93	70.79
Q13	141.13	146.47
Q14	29.35	38.33
Q15	56.76	74.08
Q16	20.47	20.27
Q17	944.35	960.16
Q18	228.83	256.7
Q19	57.03	65.63
Q20	192.67	199.1
Q21	289.82	303.06
Q22	48.86	57.07
总时间	3302.32	3563.69



结论

在TPC-H 100 GB测试场景中，数据共享的目标端查询性能能够达到源端的90%以上。

7.约束与限制

为了保障集群的稳定及安全，AnalyticDB PostgreSQL版有以下约束和限制。

购买实例后，您不需要做数据库的基础运维（例如高可用、打安全补丁等），但您需要重点关注如下事项：

- 实例升级：实例升级的过程为只读状态，升级结束时会出现一次最长30秒左右的连接闪断。需要您提前做好准备，通过连接池等机制，设置好程序的自动重连。
- 故障切换：实例Master节点和Segment节点均采用主备HA架构，当主节点发生异常或者硬件故障时，会在30秒内切换到备节点。切换过程中有30秒左右的连接闪断，需要您提前做好准备，通过连接池等机制，设置好程序的自动重连。

指标	限制

用户最大连接数	不同规格的实例用户最大连接数如下：
	• 高可用版：
	  ○ 2C16G：600
	  ○ 4C32G：800
	  ○ 8C64G：1000
	• 基础版：
	  ○ 2C8G：250
	  ○ 4C16G：350
	  ○ 8C32G：450
	  ○ 16C64G：950

最大字段的大小	最大1 GB。
每个数据库的数据量	由实例规格容量决定。
每个表的数据量	每个分区每个Segment最大128 TB。
BLOB数据量	最大1 GB。 **说明** AnalyticDB PostgreSQL版使用BYTEA数据类型代替BLOB。
每个表的行数	最多2^48行。
每个表的列数	最多1600列。
每个数据库中的表数	最多42亿个。
每个视图中的列数	最多1664列。
列、表、数据库名称的长度	最多128个字符。
对象名称的长度（包括数据库、用户、基表、视图、索引、存储过程、UDF、UDT、约束或列名）	最大63个字符。

指标	限制
每个主索引和二级索引的列数	最多32列。
单个SELECT中的列数	最多1664列。
触发器	不支持

8.名词解释

下表列出了AnalyticDB PostgreSQL所涉及到的基本概念：

名词	解释
MPP	Massively Parallel Processing，一种分布式 Shared Nothing 计算架构，支持多个无共享的节点，执行全并行计算，计算性能随节点增加而线性提升。AnalyticDB for PostgreSQL 实例即为MPP集群架构，由多个计算节点组成。
计算节点	AnalyticDB for PostgreSQL 集群的资源分配单位，一个实例由多个计算节点组成，计算节点数量的增加，可以水平提升存储容量，且保证查询响应时间不变。计算节点为用户可购买的计算资源单位，包括固定的 CPU核，内存，存储。每个计算节点规格包含1个MPP的数据分区（Segment）。
计算节点个数	集群实例所购买的计算节点数量，单实例最大支持4096个节点。集群实例的存储空间和计算资源随计算节点数量增加而线性增加。
数据分布	MPP架构下，表的数据按分区键存储在不同数据分区上，是全并行计算中的一个计算执行和存储单元。常见的分布方式有哈希分布，随机分布，复制分布。