ALIBABA CLOUD

阿里云

Polardb PostgreSQL
Polardb Postgresql（合集）

文档版本：20210712

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.新版本更新说明

本文介绍PolarDB PostgreSQL的产品功能动态，分别为内核小版本、控制台、时空数据库和API的更新说明。

> **说明** 您可通过如下语句查看PolarDB PostgreSQL的内核小版本的版本号：
>
> ```
> show polar_version;
> ```

PolarDB PostgreSQL对应的原生PostgreSQL版本如下所示。

PolarDB PostgreSQL版本	原生PostgreSQL版本
v1.1.14	11.9
v1.1.13	11.9
V1.1.12	11.9
V1.1.11	11.9
V1.1.10	11.9
V1.1.9	11.9
V1.1.8	11.9
V1.1.7	11.9
V1.1.6	11.9
V1.1.5	11.8.15
V1.1.4	11.5.12
V1.1.3	11.5.12
V1.1.2	11.2
V1.1.1	11.2c
V1.1.0	11.2

2021年7月

内核小版本（V1.1.14）

类别	名称	描述	相关文档
新增	适配GCC9编译	GCC升级至GCC9.2.1，支持更多编译优化，提高性能。	无

时空数据库（V3.9）

类别	名称	描述	相关文档
新增	ST_RemoveDriftPoints函数	新增ST_RemoveDriftPoints函数，使用指定规则删除轨迹中的漂移点。	ST_removeDriftPoints
	ST_Split函数	新增ST_Split函数，使用指定的几何对象对轨迹进行切分。	ST_Split
	ST_ExportTo函数	新增ST_ExportTo函数，将轨迹导出到外部文件存储，数据库内仅保留元数据。	ST_ExportTo
	ST_IsExternal函数	新增ST_IsExternal函数，检查轨迹是否存储于外部文件。	ST_IsExternal
	ST_ImportFrom函数	新增ST_ImportFrom函数，将外部存储模式的轨迹重新转化为存储在数据库内的轨迹。	ST_importFrom
	ST_StorageLocation函数	新增ST_StorageLocation函数，返回轨迹存储在外部的位置。	ST_StorageLocation
	ST_AKID函数	新增ST_AKID函数，返回读取轨迹时，与OSS交互时所使用的AccessKeyID。	ST_AKID
	ST_SetAccessKey函数	新增ST_SetAccessKey 函数，设置读取轨迹时，与OSS交互时所使用的AccessKey（包括AccessKeyID和AccessKeySecret，类似用户名与密码）。	ST_SetAccessKey
	ST_SetAKID函数	新增ST_SetAKID函数，设置读取轨迹时，与OSS交互时所使用的AccessKeyID。	ST_SetAkId
	ST_SetAKSecret函数	新增ST_SetAKSecret函数，设置读取轨迹时，与OSS交互时所使用的AccessKeySecret。	ST_SetAkSecret
	ST_SetStorageLocation函数	新增ST_SetStorageLocation函数，设置读取轨迹时，外部文件的存储位置。	ST_SetStorageLocation
	ST_DeleteGtf函数	新增ST_DeleteGtf函数，删除指定文件夹下所有的轨迹导出的文件。	ST_DeleteGTF
缺陷修复	修复创建矢量金字塔使用字符串方式过滤失败问题。		
	修复QGIS连接时无法列出图层列表的问题。		

2021年6月

内核小版本（V1.1.13）

类别	名称	描述	相关文档

类别	名称	描述	相关文档
新增	polar_stat_sql/polar_stat_plan增加IO统计展示	• 新增可通过查询不同视图获取相关资源信息，进而方便定位性能问题。 • 新增PolarDB PostgreSQ性能诊断，包含实例问题分析和SQL性能分析。 • polar_stat_sql/polar_stat_plan视图增加IO统计展示相关参数。	PolarDB PostgreSQL视图、性能诊断
	跨机并行查询	新增polar_bt_write_page_buffer_size参数，指定了索引构建过程中的写IO策略。	跨机并行查询概述
		跨机并行查询功能可用于BRIN索引扫描，进一步提升查询性能。	使用跨机并行查询加速索引创建
优化	基于RTO的crash recover时间改进	支持配置polar_crash_recovery_rto参数，指定实例期望的RTO时间，从而保证在发生crash recovery时，数据库可以在期望的RTO时间内完成崩溃恢复重启，恢复可用状态。	无

时空数据库（V3.8）

类别	名称	描述	相关文档
新增	ST_AsDatasetFile函数	新增ST_AsDatasetFile函数，支持将指定范围的栅格对象以指定文件格式的二进制流进行返回。	ST_AsDatasetFile
	ST_RasterDrivers函数	新增ST_RasterDrivers函数，支持对所有栅格数据驱动的状态进行查询。	ST_RasterDrivers
缺陷修复	修复ST_Clip函数指定空间参考进行重投影操作失败的问题。		

2021年5月

内核小版本（V1.1.12）

类别	名称	描述	相关文档
新增	并行执行	支持跨机并行查询： • 支持多个只读节点分布式地执行SQL查询，加速PolarDB PostgreSQL的查询性能。 • 支持通过跨机并行查询加速构建索引，提升大表场景下索引创建效率。	跨机并行查询
优化	性能监控	新增max_slot_wal_keep_size参数，限制使用复制槽（replication slot）的文件大小，防止WAL文件堆积。	无
	polar_resource_group插件	polar_resource_group插件新增total_mem_limit_remain_size、idle_mem_limit_rate、enable_terminate_active、policy_mem_release、enable_log等参数，避免内存占用过多而影响数据库进程。	无

类别	名称	描述	相关文档
	提高只读节点启动速度	提高只读节点的启动速度，提升性能。	无
	主备切换	通过并行回放，加速恢复主备切换后的数据库服务，通常情况下主库异常可以15s内恢复服务。	无

时空数据库（V3.7）

类别	名称	描述	相关文档
新增	时空并行查询	支持空间索引并行查询，提升查询效率。	开启时空并行查询
优化	栅格图像处理	ST_SubRaster函数支持栅格像元类型转换以及像元值拉伸。支持多波段遥感影像转三波段图像的AI识别场景。	ST_SubRaster
缺陷修复	修复栅格解析无效的直方图信息可能会导致数据库崩溃的问题。		
	修复进行投影变化操作后，数据库退出时可能会崩溃的问题。		
	修复栅格数据导入时可能会出现崩溃的问题。		

2021年4月

内核小版本（V1.1.11）

类别	名称	描述	相关文档
新增	idle_session_timeout参数	新增idle_session_timeout参数，用于控制空闲会话的保留时间，超出保留时间的会话将会被释放。 • 取值范围：0~2147483647 0（默认值）表示关闭功能。 • 单位 ◦ 'ms'：毫秒（默认单位） ◦ 's'：秒 ◦ 'min'：分钟 ◦ 'h'：小时 ◦ 'd'：天	无
缺陷修复	修复DBMS_JOB内置包跨库使用的问题。		
	修复CONNECT BY与ROWNUM函数连用的问题。		
	修复CONNECT BY找不到上层对象的问题。		

时空数据库（V3.6）

类别	名称	描述	相关文档
新增	栅格数据类型	新增ST_RPCGeoreference函数，用于获取栅格数据RPC（Rational Polynomial Coefficients）信息。	ST_RPCGeoreference
		新增ST_SetRPCGeoreference函数，用于设置栅格数据RPC信息。	ST_SetRPCGeoreference
		新增ST_RPCRectify函数，用于根据栅格影像的RPC参数对栅格进行校正操作，返回校正后的栅格对象。	ST_RPCRectify
	GisT索引	新增使用并行方式创建GisT索引。	并行创建空间索引
缺陷修复	修复使用pg_dump时，自定义的空间参考无法导出的问题。		

2021年3月

内核小版本（V1.1.10）

类别	名称	描述	相关文档
优化	性能优化	通过在共享内存中缓存表数据库块大小信息，优化了数据文件I/O读性能。	无

时空数据库（V3.5）

类别	名称	描述	相关文档
新增	大对象存储优化	新增Simple存储策略，优化大对象存储。	使用Simple存储策略
优化	栅格对象	使用栅格对象更新一个具有分块数据的栅格对象进行时，系统会自动删除原有栅格对象的分块数据。	无
缺陷修复	修复Trajectory扩展无法升级的问题。		
	修复某些情况下，栅格对象采用Average重采样时出错的问题。		
	修复轨迹对象中如果多个时间戳的轨迹点相同，查询结果可能不正确的问题。		
	修复Geos对象转换失败后，直接退出的问题，同时对转换失败的原因进行提示。		

内核小版本（V1.1.9）

类别	名称	描述	相关文档
新增	AWR监控	自动负载信息库（Automatic Workload Repository，简称AWR）是一种性能收集和分析工具。通过AWR工具，可以从数据库的动态视图等统计信息记录中生成一份该时段的统计分析报告，分析数据库在某个时间段的性能。	无

类别	名称	描述	相关文档
优化	WAL	采用流水线技术，优化WAL开销占比，提高整体系统的性能。	无
缺陷修复	修复无用SIGHUP信号。		
	修复TDE功能中没有声明导致函数返回值地址被截断的问题。		
	修复roaringbitmap插件对输入异常参数时访问非法内存的问题。		
	修复当只读节点日志同步延迟较大时，可能不会及时给主节点流复制反馈，从而导致流复制链接中断的问题。		

时空数据库（V3.4）

类别	名称	描述	相关文档
新增	大对象存储优化	为了加速时空数据索引构建、提高空间查询效率，优化时空对象的存储模式，支持使用时空大对象特征签名，新增polar_enable_storage_partial参数，支持时空对象行内与行外组合存储。	时空大对象特征签名
	栅格模块支持函数并行化执行	ST_ImportFrom函数支持栅格数据并行化导入。	ST_ImportFrom
		ST_BuildPyramid函数支持栅格数据并行创建金字塔。	ST_BuildPyramid
	Trajectory Empty	新增Trajectory Empty对象，支持某些场景下不符合要求而返回NULL对象。	无
缺陷修复	修复ST_AddZ函数在16BSI情况下可能结果不正确的问题。		
	修复Trajectory模块在某些情况下无法升级的问题。		
	修复Trajectory模块中部分函数无法利用索引的问题。		
	修复部分nd函数没有正确处理时间段不相交的场景，导致返回error而非false的问题。		

2021年1月

内核小版本（V1.1.8）

类别	名称	描述	相关文档
优化	PBP	默认打开主节点的持久化缓冲池（Persisted Buffer Pool）。	无
缺陷修复	修复DBMS_JOB的无主键表在逻辑复制中删除任务报错的问题。		
	修复资源组（Resource Group）进程在内存资源超限时出现内存泄漏的问题。		

时空数据库（V3.3）

类别	名称	描述	相关文档
新增	ST_JaccardSimilarity函数	新增ST_JaccardSimilarity函数，用于计算轨迹对象相似度。	ST_JaccardSimilarity
	ST_Transform函数	新增ST_Transform函数，用于将轨迹从一个空间参考系转换到另一个空间参考系。	ST_Transform
优化	矢量金字塔	创建矢量金字塔时，支持用户指定创建范围，解决在部分场景下无法自动计算空间范围的问题。	无
缺陷修复	修复某些环境下，Ganos数据目录设置不正确的问题。		

2020年12月

内核小版本（V1.1.7）

类别	名称	描述	相关文档
新增	兼容性	兼容PostgreSQL 11.9.20版本。	无

时空数据库（V3.2）

类别	名称	描述	相关文档
新增	ST_AsPNG	新增矢量金字塔返回图片格式（基于流形式）功能，用于矢量数据的快速图形化显示。	ST_AsPNG
	栅格数据	新增栅格数据类型JPEG2000压缩算法，支持16bit栅格数据压缩存储。	无
优化	st_dwithin	优化st_dwithin距离查询，提升查询性能。	无
缺陷修复	修复轨迹数据类型时间相交错误问题。		
	修复Ganos内存拓扑索引的缺陷。		

2020年11月

内核小版本（V1.1.6）

类别	名称	描述	相关文档
新增	兼容性	兼容PostgreSQL 11.9版本。	无
	主备切换	默认开启OnlinePromote功能，减少HA切换时间，提高集群可用性。	主备切换
缺陷修复	修复部分场景下节点重启可能卡顿的情况。		
	修复部分场景下内存溢出导致实例崩溃的问题。		
	修复部分场景下只读节点崩溃的问题。		

2020年10月

时空数据库（V3.1）

类别	名称	描述	相关文档
新增	Trajectory数据类型	新增Gist索引支持索引轨迹类型，并提供六种不同维度的算子族以支持不同维度的分析需求。	无
		新增时空外包框类型BoxND，可用于时空范围表示以及存储轨迹。	无
		新增对应不同维度的相交(&&)、包含(@>)、被包含(<@)算子。	无
		新增ST_ndIntersects、ST_ndDWithin、ST_ndContains、ST_ndWithin四类轨迹处理函数。	• ST_{Z|T|2D|2DT|3D|3DT}Intersects • ST_{2D|2DT|3D|3DT}DWithin • ST_{T|2D|2DT|3D|3DT}Contains • ST_{T|2D|2DT|3D|3DT}Within
		对轨迹类型提供统计信息收集功能，以及根据统计信息预估扫描代价功能。	无
		提供新的索引方式TrajGist，提供更好的索引选择。	无
	ganos_update函数	新增ganos_update函数，用 `select ganos_update();` 命令可以升级所有的Ganos插件到最新版本。	无
优化	时空范围查询	优化时空范围查询，GiST索引二阶段查询优化，提升查询性能。	无
	矢量金字塔功能	支持任意SRID坐标的源数据，支持3857和4326两种瓦片输出。	无
		新增pixelSize参数设置，对点数据进行聚合，减少瓦片的数量。	无
缺陷修复	修复某些情况下更新Ganos Raster失败问题。		
	修复Ganos二进制文件更新到新版本可能出现崩溃的问题。		
	修复用默认参数构建矢量金字塔点数据后，顶级瓦片数据量过大的问题。		

2020年9月

内核小版本（V1.1.4）

类别	名称	描述	相关文档
新增	ali_decoding插件	支持ali_decoding插件，用于数据库间增量同步DML语句。	无
	视图	新增进程状态信息视图。	无
		新增SQL信息视图耗时统计。	无
	审计日志	新增审计日志包含出错SQL。	无
缺陷修复	修复在特定场景下数据页预取可能会导致死锁的问题。		
	修复分区表修剪时JOIN查询不到正确分区的问题。		

时空数据库（V3.0）

类别	名称	描述	相关文档
	支持具有SubSet的NetCDF数据类型数据	新增支持具有SubSet的NetCDF数据类型数据，可按照指定的图层名称导入。	无
	栅格数据自定义元数据以及时序信息	新增ST_MetaItems函数，获取所有的自定义元数据项目名称。	ST_MetaItems
		新增ST_MetaData函数， 获取自定义元数据项以及返回以JSON方式表达的元数据项。	ST_MetaData
		新增ST_SetMetaData函数，用于设置元数据项。	ST_SetMetaData
		新增ST_BeginDateTime函数，用于获取栅格数据的起始时间。	ST_BeginDateTime
		新增ST_EndDateTime函数，用户获取栅格数据的终止时间。	ST_EndDateTime
		新增ST_SetBeginDateTime函数，用于设置栅格数据的开始时间。	ST_SetBeginDateTime
		新增ST_SetEndDateTime函数，用于设置栅格数据的结束时间。	ST_SetEndDateTime
		新增ST_SetDateTime函数，用于设置栅格数据的开始和结束时间以及波段获取时间。	ST_SetDateTime
	支持栅格数据返回基于流形式的图片格式	新增ST_AsImage函数，用于获取基于流形式的图片格式。	ST_AsImage
		新增ST_AsPng，用于获取基于流形式的PNG图片格式。	ST_AsPNG
		新增ST_AsJPEG，用于获取基于流形式的JPEG图片格式。	ST_AsJPEG

类别	名称	描述	相关文档
新增	支持几何网格数据类型以及操作运算	新增GeomGrid数据类型。	GeomGrid SQL参考
		新增ST_AsText函数，用于将网格数据类型转换为文本表示方式。	ST_AsText
		新增ST_AsGeometry函数，用于将网格数据类型转换为几何数据类型。	ST_AsGeometry
		新增ST_AsBinary函数，用于将网格数据类型转换为二进制数据类型。	ST_AsBinary
		新增ST_AsBox函数， 用于将网格数据量类型转换为BOX数据类型。	ST_AsBox
		新增ST_AsGrid函数， 用于计算几何数据类型所对应的几何网格数据。	ST_AsGrid
		新增ST_GridFromText函数， 用于将基于文本表示网格转换为几何网格数据类型。	ST_GridFromText
		新增ST_GridFromBinary函数，用于将基于二进制的表示的网格转换为几何网格数据类型。	ST_GridFromBinary
		新增ST_Intersects函数，用于判断栅格数据类型与几何数据类型是否相交。	ST_Intersects
		新增ST_Contains函数，用于判断栅格数据与栅格数据，栅格数据与几何数据是否是包含关系。	ST_Contains
		新增ST_Within函数，用于判断网格数据与网格数据，网格数据与几何数据是否是被包含关系。	ST_Within
	矢量数据类型快速显示存储	新增ST_BuildPyramid函数，用于创建快显支撑数据表。	ST_BuildPyramid
		新增ST_DeletePyramid函数，用于删除快显支撑数据表。	ST_DeletePyramid
		新增ST_Tile函数，用于获取基于MVT格式的矢量数据。	ST_Tile
缺陷修复	修复在某些情况下创建金字塔会出现Out Of Memory的问题。		
	修复移动对象无法创建2000-01-01时间点的问题。		
	修复某些场景下移动对象使用ST_Intersection返回子轨迹错误的问题。		
	修复Ganos升级时会出现奔溃问题。		
	PostGIS兼容性升级到2.5.4。		

2020年8月

内核小版本（V1.1.3）

类别	名称	描述	相关文档
新增	视图	新增SQL执行统计信息视图。	无
		新增QPS统计信息视图。	无
		新增CGroup状态信息视图。	无
缺陷修复	修复SLRU内存大小计算错误问题。		
	修复回放延迟导致RO重建问题。		

时空数据库（V2.9）

类别	名称	描述	相关文档
新增	COG文件格式支持	新增COG(Cloud Optimize Geotiff) 文件格式支持。支持读取COG文件格式中存储的金字塔信息。	无
	ST_AddZ函数	新增ST_AddZ函数，支持通过栅格数据的像素值为几何对象添加z值。	ST_AddZ
	栅格对象空间范围信息获取增强	新增ST_Extent函数，用于获得栅格对象的空间范围，以BOX形式返回。	ST_Extent
		新增ST_Envelope函数，用于获得栅格对象的空间范围，以几何对象形式返回。	ST_Envelope
		新增ST_ConvexHull函数，用于获得栅格对象的空间范围，以几何对象形式返回。	ST_ConvexHull
		新增ST_Height函数，用于获得栅格对象的像素高度。	ST_Height
		新增ST_Width函数，用于获得栅格对象的像素宽度。	ST_Width
缺陷修复	修复使用外部栅格数据会使用1*n分块导致性能局限性问题，允许用户通过存储选项自定义分块的大小。		
	修复ST_Values函数在查询某些方向的线对象时结果与坐标排序不一致的问题。		
	修复ST_BestPyramidLevel函数在某些情况下会返回负数的问题。		
	修复ST_BuildPyramid函数在某些情况下会重复创建金字塔的问题。		
	修复Truncate栅格表时未能清理对应的块表的问题。		
	修复ST_ExportTo函数对于CreateOption在某些情况下无效的问题。		
	修复ST_ClearChunks函数对于表名存在大小写时会出现错误的问题。		
	修复外部金字塔在某些情况下无法创建overview的问题。		

类别	名称	描述	相关文档
	修复具有外部金字塔的栅格对象无法创建内部金字塔的问题。		
	修复具有NaN数值的栅格数据在计算统计信息时会导致结果不正确的问题。		

2020年7月

内核小版本（V1.1.2）

类别	名称	描述	相关文档
新增	临时表	支持临时表文件写本地存储，大幅提升临时表操作的性能。	无
	触发事件	支持polar_superuser进行触发事件（Event Trigger）的逻辑订阅。	无
	自定义资源隔离	支持自定义资源隔离组发起请求（Request）。	无
优化	崩溃恢复流程	优化崩溃恢复流程，减少极端情况下的崩溃恢复耗时。	无
缺陷修复	修复polar_superuser无法使用索引建议器（index advisor）的问题。		

时空数据库（V2.8）

类别	名称	描述	相关文档
新增	栅格数据元数据访问接口增强	新增ST_XMin函数，用于获取栅格数据X方向最小值。	ST_XMin
		新增ST_YMin函数，用于获取栅格数据Y方向最小值。	ST_YMin
		新增ST_XMax函数，用于获取栅格数据X方向最大值。	ST_XMax
		新增ST_YMax函数，用于获取栅格数据Y方向最大值。	ST_YMax
		新增ST_ChunkHeight函数，用于获取栅格数据分块高度。	ST_ChunkHeight
		新增ST_ChunkWidth函数，用于获取栅格数据分块宽度。	ST_ChunkWidth
		新增ST_ChunkBands函数，用于获取栅格数据分块波段数量。	ST_ChunkBands
	ST_SrFromEsriWkt函数	新增ST_SrFromEsriWkt函数，用于支持Esri格式空间参考字符串转换为OGC格式空间参考字符串。	ST_SrFromEsriWkt
	栅格数据类型	新增栅格数据类型支持Zstd和Snappy压缩方式。	无
	点云数据类型	新增点云数据类型支持二进制拷贝功能。	无
	环境变量	新增支持PROJ_LIB和GDAL_DATA环境变量设置，同时部署相关数据。	无

类别	名称	描述	相关文档
缺陷修复	修复OSS路径非法导致数据库崩溃问题。		
	修复部分栅格数据导入SRID与定义不一致的问题。		

2020年6月

内核小版本（V1.1.1）

类别	名称	描述	相关文档
新增	polar_proxy_utils插件	新增polar_proxy_utils插件，用于管理与proxy相关的功能，主要支持只读UDF和只读表的配置，允许通过集群地址将只读UDF以及只读表的访问路由到只读节点。	无
	polar_resource_group插件	新增polar_resource_group插件，支持自定义资源隔离组，基于user、database、session粒度，通过CPU、Memory维度进行资源隔离。	无
优化	可靠性和可用性	数据库计算节点和文件系统解耦，文件系统可独立运维，大幅提高数据库的可靠性和可用性。	无
	事务处理	使用单调递增版本号替代原有的活跃事务列表快照，大幅提升数据库事务处理性能。	无
	执行计划	执行计划优化，避免使用过旧的统计信息。	无
缺陷修复	修复了插件timescaledb在申请内存时出错时进程的异常问题。		
	修复了I/O监控功能中进程退出后没有汇总统计信息。		
	修复了lock_debug开启后，可能发生空指针异常问题。		
	修复了特定情况下导致pg_cron插件不可用的问题。		
	修复了社区已知的DSM死锁问题。		
	修复了用户连接数超限的问题。		

2020年5月

内核小版本（V1.1.0）

类别	名称	描述	相关文档
新增	polar_concurrency_control插件	新增polar_concurrency_control插件，可以对事务执行、SQL查询、存储过程、DML等操作进行并发限制，您可以自定义大查询，并对大查询进行并发限制，优化高并发下的执行性能。	无

类别	名称	描述	相关文档
	oss_fdw插件	新增oss_fdw插件，用于Aliyun OSS外部表支持，您可以通过OSS外部表将数据库数据写入到OSS，也可以通过OSS外部表将OSS数据加载到数据库中，OSS外部表支持并行和压缩，极大提高了导入和导出数据的性能，同时也可以使用这个功能来实现多类型存储介质的冷热数据存储。	使用oss_fdw读写外部数据文本文件
优化	polar_stat_activity视图	优化polar_stat_activity视图，新增wait_info列和wait_time列，分别用于监控进程等待对象（pid或fd）的等待时长。	无
	索引	提供插入索引时的索引页预扩展功能，用于提升将数据插入带有索引的表的执行性能。	无

时空数据库（V2.7）

类别	名称	描述	相关文档
新增	MD5操作函数	新增ST_MD5Sum函数，用于获取栅格对象的MD5码值。	ST_MD5Sum
		新增ST_SetMD5Sum函数，用于设置栅格对象的MD5码值。	ST_SetMD5Sum
	空间栅格对象OSS认证方式	新增ST_AKId函数，用于获取以OSS方式存储的栅格对象的AcessKey ID	ST_AKId
		新增ST_SetAccessKey函数，用于设置以OSS方式存储的栅格对象的AcessKey ID和AccessKey Secret。	ST_SetAccessKey
		新增ST_SetAKId函数，用于设置以OSS方式存储的栅格对象的AcessKey ID。	ST_SetAKId
		新增ST_SetAKSecret函数，用于设置以OSS方式存储的栅格对象的AcessKey Secret。	ST_SetAKSecret
	空间栅格元数据操作函数	新增ST_ScaleX函数，用于获取栅格对象在空间参考系下X方向像素宽度。	ST_ScaleX
		新增ST_ScaleY函数，用于获取栅格对象在空间参考系下Y方向像素宽度。	ST_ScaleY
		新增ST_SetScale函数，用于设置栅格对象在空间参考系下像素宽度。	ST_SetScale
		新增ST_SkewX函数，用于获取栅格对象在空间参考系下X方向旋转。	ST_SkewX
		新增ST_SkewY函数，用于获取栅格对象在空间参考系下Y方向旋转。	ST_SkewY
		新增ST_SetSkew函数，用于设置栅格对象在空间参考系下旋转。	ST_SetSkew

类别	名称	描述	相关文档
		新增ST_UpperLeftX函数，用于获取栅格对象在空间参考系下左上角点的X坐标。	ST_UpperLeftX
		新增ST_UpperLeftY函数，用于获取栅格对象在空间参考系下左上角点的Y坐标。	ST_UpperLeftY
		新增ST_SetUpperLeft函数，用于获取栅格对象在空间参考系下左上角点坐标。	ST_SetUpperLeft
		新增ST_PixelWidth函数，用于获取栅格对象在空间参考系下像素宽度。	ST_PixelWidth
		新增ST_PixelHeight函数，用于获取栅格对象在空间参考系下像素高度。	ST_PixelHeight
缺陷修复	修复由于聚集函数导致扩展升级失败的问题。		

2.快速入门

快速入门旨在介绍如何创建PolarDB PostgreSQL集群、进行基本设置以及连接数据库集群，使您能够了解从购买PolarDB到开始使用的流程。

使用流程

通常，从购买PolarDB（创建新集群）到可以开始使用，您需要完成如下操作。

1. 创建PolarDB PostgreSQL数据库集群
2. 设置集群白名单
3. 创建数据库账号
4. 查看或申请连接地址
5. 连接数据库集群

3.性能白皮书

PostgreSQL自带一款轻量级的压力测试工具pgbench。pgbench是一种在PostgreSQL上运行基准测试的简单程序，它可以在并发的数据库会话中重复运行相同的SQL命令。本文介绍如何使用pgbench测试PolarDB PostgreSQL集群的最大性能。

测试环境

- 所有测试均在华北1（青岛）地域完成，PolarDB集群和ECS实例在同一可用区。
- ECS的实例规格：ecs.g5.16xlarge（64核 256 GiB）。
- ECS存储规格：SSD本地盘200 GiB。
- 网络类型：专有网络（VPC），PolarDB集群和ECS实例在同一VPC。
- 操作系统：CentOS 7.6 x64。

> **说明** CentOS 6不支持PostgreSQL 11。

测试指标

- 只读QPS

  数据库只读时每秒执行的SQL数（仅包含SELECT）。

- 读写QPS

  数据库读写时每秒执行的SQL数（包含INSERT、SELECT、UPDATE）。

准备工作

- 安装测试工具

  执行如下命令在ECS实例中安装PostgreSQL 11。

  ```
  yum install -y https://download.postgresql.org/pub/repos/yum/10/redhat/rhel-7.8-x86_64/pgdg-redhat-repo-42.0-11.noarch.rpm
  yum install -y postgresql11
  ```

- 修改集群参数

  PolarDB集群需要修改的配置参数如下：

  ```
  log_statement = 'none'
  enable_hashjoin=off
  enable_mergejoin=off
  enable_bitmapscan=off
  ```

  > **说明**
  > - 目前仅log_statement参数支持在控制台修改。设置集群参数的方式，请参见设置集群参数。
  > - 其他参数无法在控制台直接修改，您需要提交工单联系技术支持工程师申请修改。

  修改配置参数后，重启集群让配置生效。

测试方法

1. 通过以下命令配置环境变量：

```
export PGHOST=<PolarDB集群地址的私网地址>
export PGPORT=<PolarDB集群地址的私网端口>
export PGDATABASE=postgres
export PGUSER=<PolarDB数据库用户名>
export PGPASSWORD=<PolarDB对应用户的密码>
```

> **说明** 如何查看PolarDB PostgreSQL集群的连接地址，请参见查看集群地址。

2. 根据目标库大小初始化测试数据，具体命令如下：
   - 初始化数据10亿：

     ```
     /usr/pgsql-11/bin/pgbench -i -s 10000
     ```

   - 初始化数据5亿：

     ```
     /usr/pgsql-11/bin/pgbench -i -s 5000
     ```

   - 初始化数据1亿：

     ```
     /usr/pgsql-11/bin/pgbench -i -s 1000
     ```

3. 创建只读和读写的测试脚本。
   - 创建只读脚本*ro.sql*：
     a. 在命令行执行 `vim ro.sql` 命令。
     b. 按I键进入编辑页面。
     c. 在编辑页面输入如下内容：

     ```
     \set aid random_gaussian(1, :range, 10.0)
     SELECT abalance FROM pgbench_accounts WHERE aid = :aid;
     ```

     d. 按ESC键退出编辑，输入 `:wq` 进行保存并退出。
   - 创建读写脚本*rw.sql*：
     a. 在命令行执行 `vim rw.sql` 命令。
     b. 按I键进入编辑页面。
     c. 在编辑页面输入如下内容：

     ```
     \set aid random_gaussian(1, :range, 10.0)
     \set bid random(1, 1 * :scale)
     \set tid random(1, 10 * :scale)
     \set delta random(-5000, 5000)
     BEGIN;
     UPDATE pgbench_accounts SET abalance = abalance + :delta WHERE aid = :aid;
     SELECT abalance FROM pgbench_accounts WHERE aid = :aid;
     UPDATE pgbench_tellers SET tbalance = tbalance + :delta WHERE tid = :tid;
     UPDATE pgbench_branches SET bbalance = bbalance + :delta WHERE bid = :bid;
     INSERT INTO pgbench_history (tid, bid, aid, delta, mtime) VALUES (:tid, :bid, :aid, :delta, CURREN
     T_TIMESTAMP);
     END;
     ```

d. 按ESC键退出编辑，输入 `:wq` 进行保存并退出。

4. 在命令行执行如下命令测试：
   - 只读测试：

```
88C 710 GB(polar.pg.x8.12xlarge)
总数据量10亿，热数据1亿
/usr/pgsql-11/bin/pgbench -M prepared -v -r -P 1 -f ./ro.sql -c 256 -j 128 -T 120 -D scale=10000 -D range
=100000000
总数据量10亿，热数据5亿
/usr/pgsql-11/bin/pgbench -M prepared -n -r -P 1 -f ./ro.sql -c 256 -j 128 -T 120 -D scale=10000 -D range
=500000000
总数据量10亿，热数据10亿
/usr/pgsql-11/bin/pgbench -M prepared -n -r -P 1 -f ./ro.sql -c 256 -j 128 -T 120 -D scale=10000 -D range
=1000000000
64C 512 GB(polar.pg.x8.8xlarge)
总数据量10亿，热数据1亿
/usr/pgsql-11/bin/pgbench -M prepared -v -r -P 1 -f ./ro.sql -c 256 -j 256 -T 120 -D scale=10000 -D range
=100000000
总数据量10亿，热数据5亿
/usr/pgsql-11/bin/pgbench -M prepared -n -r -P 1 -f ./ro.sql -c 256 -j 128 -T 120 -D scale=10000 -D range
=500000000
总数据量10亿，热数据10亿
/usr/pgsql-11/bin/pgbench -M prepared -n -r -P 1 -f ./ro.sql -c 256 -j 128 -T 120 -D scale=10000 -D range
=1000000000
32C 256 GB(polar.pg.x8.4xlarge)
总数据量10亿，热数据1亿
/usr/pgsql-11/bin/pgbench -M prepared -v -r -P 1 -f ./ro.sql -c 128 -j 128 -T 120 -D scale=10000 -D range
=100000000
总数据量10亿，热数据5亿
/usr/pgsql-11/bin/pgbench -M prepared -n -r -P 1 -f ./ro.sql -c 128 -j 128 -T 120 -D scale=10000 -D range
=500000000
总数据量10亿，热数据10亿
/usr/pgsql-11/bin/pgbench -M prepared -n -r -P 1 -f ./ro.sql -c 128 -j 128 -T 120 -D scale=10000 -D range
=1000000000
16C 128 GB(polar.pg.x8.2xlarge)
总数据量10亿，热数据1亿
/usr/pgsql-11/bin/pgbench -M prepared -v -r -P 1 -f ./ro.sql -c 64 -j 64 -T 120 -D scale=10000 -D range=1
00000000
总数据量10亿，热数据5亿
/usr/pgsql-11/bin/pgbench -M prepared -n -r -P 1 -f ./ro.sql -c 64 -j 64 -T 120 -D scale=10000 -D range=5
00000000
总数据量10亿，热数据10亿
/usr/pgsql-11/bin/pgbench -M prepared -n -r -P 1 -f ./ro.sql -c 64 -j 64 -T 120 -D scale=10000 -D range=1
000000000
8C 64 GB(polar.pg.x8.xlarge)
总数据量10亿，热数据1亿
/usr/pgsql-11/bin/pgbench -M prepared -v -r -P 1 -f ./ro.sql -c 64 -j 32 -T 120 -D scale=10000 -D range=1
00000000
总数据量10亿，热数据5亿
/usr/pgsql-11/bin/pgbench -M prepared -n -r -P 1 -f ./ro.sql -c 64 -j 32 -T 120 -D scale=10000 -D range=5
00000000
总数据量10亿，热数据10亿
/usr/pgsql-11/bin/pgbench -M prepared -n -r -P 1 -f ./ro.sql -c 64 -j 32 -T 120 -D scale=10000 -D range=1
000000000
```

```
000000000
8C 32 GB(polar.pg.x4.xlarge)
总数据量10亿，热数据1亿
/usr/pgsql-11/bin/pgbench -M prepared -v -r -P 1 -f ./ro.sql -c 64 -j 32 -T 120 -D scale=10000 -D range=1
00000000
总数据量10亿，热数据5亿
/usr/pgsql-11/bin/pgbench -M prepared -n -r -P 1 -f ./ro.sql -c 64 -j 32 -T 120 -D scale=10000 -D range=5
00000000
总数据量10亿，热数据10亿
/usr/pgsql-11/bin/pgbench -M prepared -n -r -P 1 -f ./ro.sql -c 64 -j 32 -T 120 -D scale=10000 -D range=1
000000000
4C 16 GB(polar.pg.x4.large)
总数据量5亿，热数据5000万
/usr/pgsql-11/bin/pgbench -M prepared -v -r -P 1 -f ./ro.sql -c 16 -j 16 -T 120 -D scale=5000 -D range=50
000000
总数据量5亿，热数据1亿
/usr/pgsql-11/bin/pgbench -M prepared -n -r -P 1 -f ./ro.sql -c 16 -j 16 -T 120 -D scale=5000 -D range=10
0000000
2C 8 GB(polar.pg.x4.medium)
总数据量1亿，热数据5000万
/usr/pgsql-11/bin/pgbench -M prepared -v -r -P 1 -f ./ro.sql -c 16 -j 32 -T 120 -D scale=1000 -D range=50
000000
总数据量1亿，热数据1亿
/usr/pgsql-11/bin/pgbench -M prepared -n -r -P 1 -f ./ro.sql -c 16 -j 32 -T 120 -D scale=1000 -D range=10
0000000
```

- 读写测试：

```
88C 710 GB(polar.pg.x8.12xlarge)
总数据量10亿，热数据1亿
/usr/pgsql-11/bin/pgbench -M prepared -v -r -P 1 -f ./rw.sql -c 128 -j 128 -T 120 -D scale=10000 -D range
=100000000
总数据量10亿，热数据5亿
/usr/pgsql-11/bin/pgbench -M prepared -n -r -P 1 -f ./rw.sql -c 128 -j 128 -T 120 -D scale=10000 -D range
=500000000
总数据量10亿，热数据10亿
/usr/pgsql-11/bin/pgbench -M prepared -n -r -P 1 -f ./rw.sql -c 128 -j 128 -T 120 -D scale=10000 -D range
=1000000000
64C 512 GB(polar.pg.x8.8xlarge)
总数据量10亿，热数据1亿
/usr/pgsql-11/bin/pgbench -M prepared -v -r -P 1 -f ./rw.sql -c 128 -j 128 -T 120 -D scale=10000 -D range
=100000000
总数据量10亿，热数据5亿
/usr/pgsql-11/bin/pgbench -M prepared -n -r -P 1 -f ./rw.sql -c 128 -j 128 -T 120 -D scale=10000 -D range
=500000000
总数据量10亿，热数据10亿
/usr/pgsql-11/bin/pgbench -M prepared -n -r -P 1 -f ./rw.sql -c 128 -j 128 -T 120 -D scale=10000 -D range
=1000000000
32C 256 GB(polar.pg.x8.4xlarge)
总数据量10亿，热数据1亿
/usr/pgsql-11/bin/pgbench -M prepared -v -r -P 1 -f ./rw.sql -c 128 -j 128 -T 120 -D scale=10000 -D range
=100000000
总数据量10亿，热数据5亿
/usr/pgsql-11/bin/pgbench -M prepared -n -r -P 1 -f ./rw.sql -c 128 -j 128 -T 120 -D scale=10000 -D range
=500000000
```

总数据量10亿，热数据10亿
/usr/pgsql-11/bin/pgbench -M prepared -n -r -P 1 -f ./rw.sql -c 128 -j 128 -T 120 -D scale=10000 -D range=1000000000
16C 128 GB(polar.pg.x8.2xlarge)
总数据量10亿，热数据1亿
/usr/pgsql-11/bin/pgbench -M prepared -v -r -P 1 -f ./rw.sql -c 64 -j 64 -T 120 -D scale=10000 -D range=100000000
总数据量10亿，热数据5亿
/usr/pgsql-11/bin/pgbench -M prepared -n -r -P 1 -f ./rw.sql -c 64 -j 64 -T 120 -D scale=10000 -D range=500000000
总数据量10亿，热数据10亿
/usr/pgsql-11/bin/pgbench -M prepared -n -r -P 1 -f ./rw.sql -c 64 -j 64 -T 120 -D scale=10000 -D range=1000000000
8C 64 GB(polar.pg.x8.xlarge)
总数据量10亿，热数据1亿
/usr/pgsql-11/bin/pgbench -M prepared -v -r -P 1 -f ./rw.sql -c 64 -j 64 -T 120 -D scale=10000 -D range=100000000
总数据量10亿，热数据5亿
/usr/pgsql-11/bin/pgbench -M prepared -n -r -P 1 -f ./rw.sql -c 64 -j 64 -T 120 -D scale=10000 -D range=500000000
总数据量10亿，热数据10亿
/usr/pgsql-11/bin/pgbench -M prepared -n -r -P 1 -f ./rw.sql -c 64 -j 64 -T 120 -D scale=10000 -D range=1000000000
8C 32 GB(polar.pg.x4.xlarge)
总数据量10亿，热数据1亿
/usr/pgsql-11/bin/pgbench -M prepared -v -r -P 1 -f ./rw.sql -c 32 -j 32 -T 120 -D scale=10000 -D range=100000000
总数据量10亿，热数据5亿
/usr/pgsql-11/bin/pgbench -M prepared -n -r -P 1 -f ./rw.sql -c 32 -j 32 -T 120 -D scale=10000 -D range=500000000
总数据量10亿，热数据10亿
/usr/pgsql-11/bin/pgbench -M prepared -n -r -P 1 -f ./rw.sql -c 32 -j 32 -T 120 -D scale=10000 -D range=1000000000
4C 16 GB(polar.pg.x4.large)
总数据量5亿，热数据5000万
/usr/pgsql-11/bin/pgbench -M prepared -v -r -P 1 -f ./rw.sql -c 16 -j 16 -T 120 -D scale=5000 -D range=50000000
总数据量5亿，热数据1亿
/usr/pgsql-11/bin/pgbench -M prepared -n -r -P 1 -f ./rw.sql -c 16 -j 16 -T 120 -D scale=5000 -D range=100000000
2C 4 GB(polar.pg.x4.medium)
总数据量1亿，热数据5000万
/usr/pgsql-11/bin/pgbench -M prepared -v -r -P 1 -f ./rw.sql -c 8 -j 8 -T 120 -D scale=1000 -D range=50000000
总数据量1亿，热数据1亿
/usr/pgsql-11/bin/pgbench -M prepared -n -r -P 1 -f ./rw.sql -c 8 -j 8 -T 120 -D scale=1000 -D range=10000000

> 说明
> - scale：该值乘以10万表示测试数据量。
> - range：表示活跃数据量。
> - -c：表示测试连接数，测试连接数不代表该规格的最大连接数，最大连接数请参考计算节点规格。

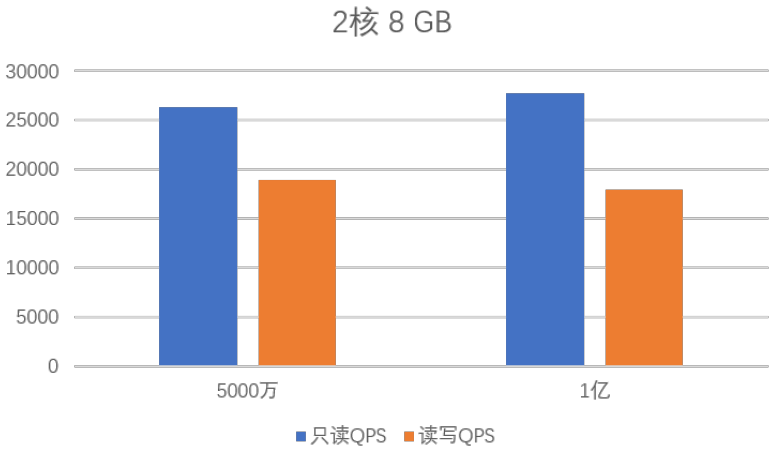
测试结果

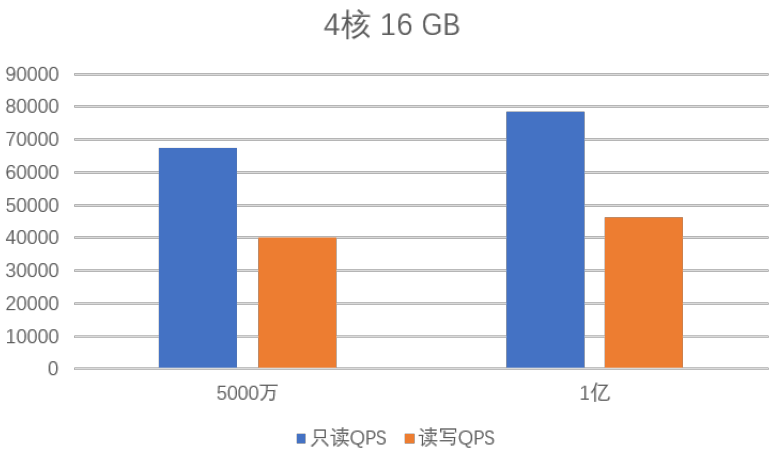
规格	测试数据量	热（活跃）数据量	只读QPS	读写QPS
polar.pg.x8.12xlarge 88核 710 GB	10亿	1亿	628863.87	259862.37
		5亿	612612.86	243871.65
		10亿	607162.83	217592.26
polar.pg.x8.8xlarge 64核 512 GB	10亿	1亿	587612.46	227352.23
		5亿	539371.54	209782.29
		10亿	498728.28	199821.76
polar.pg.x8.4xlarge 32核 256 GB	10亿	1亿	487138.21	237280.67
		5亿	442339.25	215081.79
		10亿	420348.28	198341.34
polar.pg.x8.2xlarge 16核 128 GB	10亿	1亿	269781.83	168612.27
		5亿	249271.32	131725.26
		10亿	233219.96	109826.82
polar.pg.x8.xlarge 8核 64 GB	10亿	1亿	148621.38	71787.83
		5亿	130862.86	59298.44
		10亿	123151.90	52324.72
polar.pg.x4.xlarge 8核 32 GB	10亿	1亿	137366.92	59738.33
		5亿	114932.64	52873.87
		10亿	109248.29	48993.82
polar.pg.x4.large 4核 16 GB	5亿	5000万	67289.76	40221.21
		1亿	78393.56	46281.85

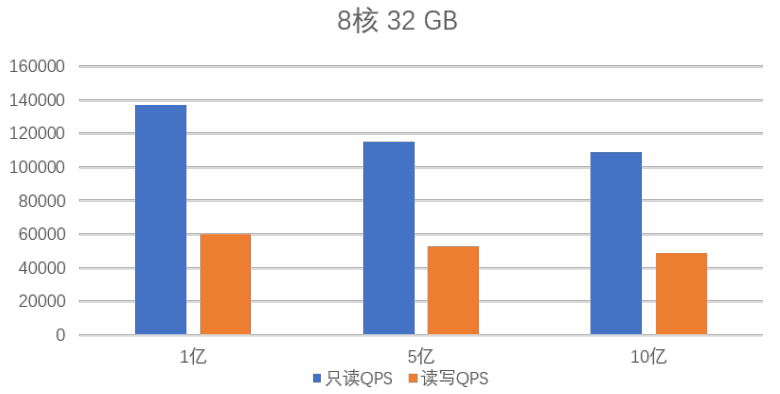
规格	测试数据量	热（活跃）数据量	只读QPS	读写QPS
polar.pg.x4.medium 2核 8 GB	1亿	5000万	26383.91	18983.55
		1亿	27821.49	17986.46

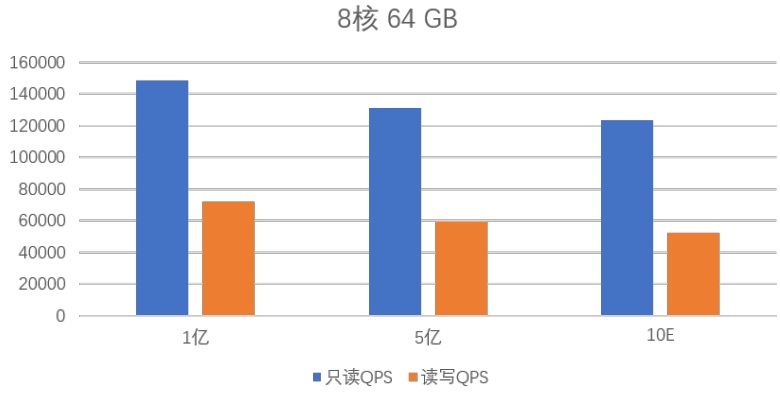
说明

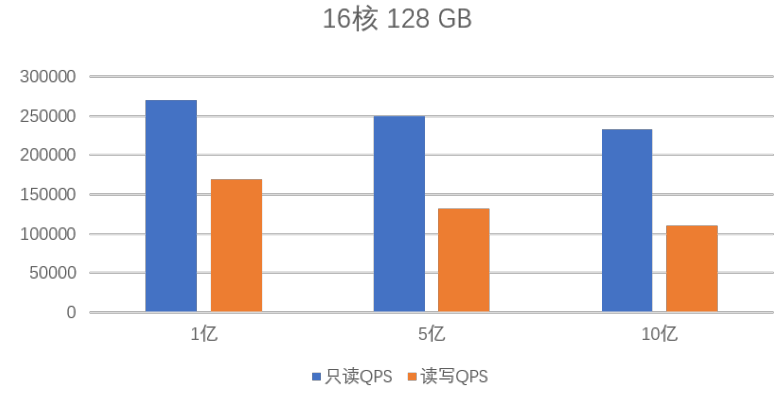
- 规格：PolarDB PostgreSQL的规格代码。
- 测试数据量：本轮测试数据的记录条数。
- 热（活跃）数据量：本轮测试的查询、更新SQL的记录条数。
- 只读QPS：只读测试的结果，表示每秒请求数。
- 读写QPS：读写测试的结果，表示每秒请求数。

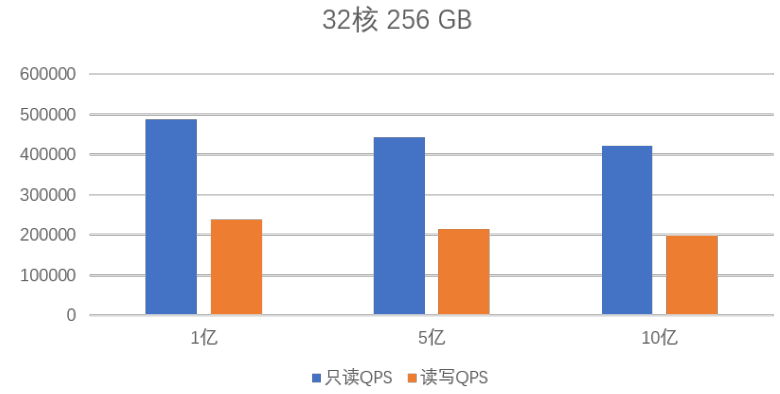


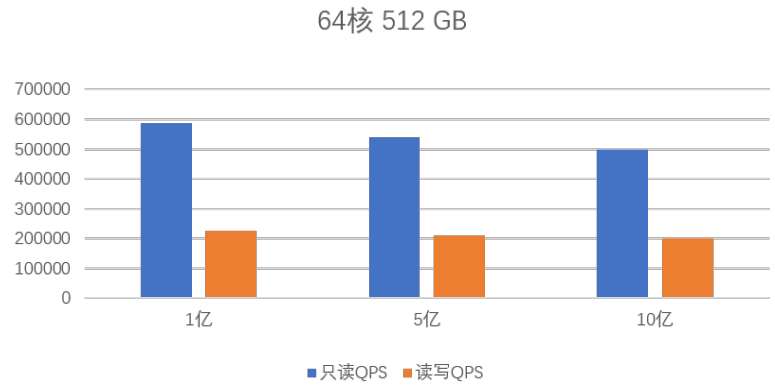


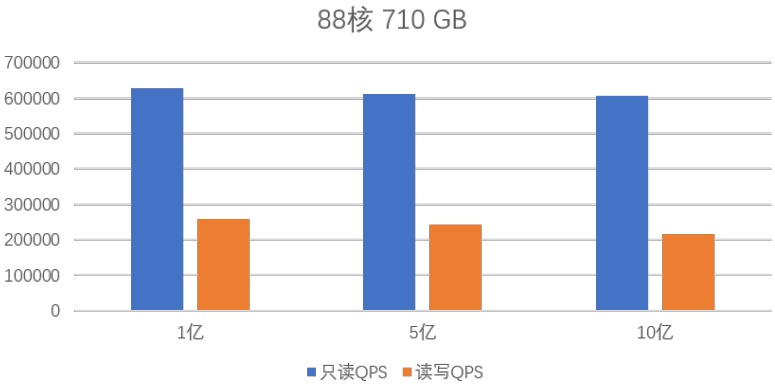

8核 32 GB
160000
140000
120000
100000
80000
60000
40000
20000
0
1亿
5亿
10亿
只读QPS
读写QPS


8核 64 GB
160000
140000
120000
100000
80000
60000
40000
20000
0
1亿
5亿
10E
只读QPS
读写QPS


16核 128 GB
300000
250000
200000
150000
100000
50000
0
1亿
5亿
10亿
只读QPS
读写QPS


32核 256 GB
600000
500000
400000
300000
200000
100000
0
1亿
5亿
10亿
只读QPS
读写QPS


64核 512 GB
700000
600000
500000
400000
300000
200000
100000
0
1亿
5亿
10亿
只读QPS
读写QPS


88核 710 GB
700000
600000
500000
400000
300000
200000
100000
0
1亿
5亿
10亿
只读QPS
读写QPS

4.SDK下载

PolarDB支持Java、Node.js、Go、PHP、.NET和Python开发。

下表列举了各语言SDK的下载地址和开发指南，更多SDK的信息，请访问阿里云开放平台。

Alibaba Cloud SDK	PolarDB SDK	说明文档
Alibaba Cloud SDK for Java	Alibaba Cloud PolarDB SDK for Java	快速开始
Alibaba Cloud SDK for Node.js	Alibaba Cloud PolarDB SDK for Node.js	快速开始
Alibaba Cloud SDK for Go	Alibaba Cloud PolarDB SDK for Go	快速开始
Alibaba Cloud SDK for PHP	Alibaba Cloud PolarDB SDK for PHP	快速开始
Alibaba Cloud SDK for .NET	Alibaba Cloud PolarDB SDK for .NET	快速开始
Alibaba Cloud SDK for Python	Alibaba Cloud PolarDB SDK for Python	快速开始

5.常见问题FAQ

本文介绍PolarDB PostgreSQL的常见问题和解答。

基本问题

- Q：什么是PolarDB?

  A：PolarDB是一个关系型数据库云服务，目前已在全球十多个地域（Region）的数据中心部署，向用户提供开箱即用的在线数据库服务。PolarDB目前支持3种独立的引擎，分别可以100%兼容MySQL、100%兼容PostgreSQL、高度兼容Oracle语法，存储容量最高可达100 TB，用户可以按需购买、按量计费，最低每小时只需要付费1.3元即可体验完整的产品功能。详情请参见什么是PolarDB。

- Q：为什么云原生关系型数据库PolarDB优于传统数据库?

  A：相较于传统数据库，云原生关系型数据库PolarDB支持上百TB级别海量数据存储，提供高可用和高可靠保障、快速弹性升降级、无锁备份等功能，详情请参见产品优势。

- Q：PolarDB是什么时候发布？什么时候开始商用?

  A：2017年9月发布公测，2018年3月开始商用。

- Q：集群和节点分别指的是什么?

  A：PolarDB集群版采用多节点集群的架构，集群中有一个主节点和多个只读节点。单个PolarDB集群支持跨可用区，但不能跨地域，面向集群进行管理和计费。详情请参见PolarDB术语表。

- Q：支持哪些编程语言?

  A：PolarDB支持Java、Python、PHP、Golang、C、C++、.NET、Node.js等编程语言。

- Q：PolarDB是分布式数据库吗?

  A：是的，PolarDB是基于Parallel Raft一致性协议的分布式存储集群，计算引擎是由1~16个分布在不同服务器上的计算节点构成，最大支持100 TB，最高支持88核710 GB内存，可在线动态扩容存储和计算资源，扩容时不会影响业务的正常运行。

- Q：购买PolarDB后，如果需要分库分表是否还需要购买PolarDB-X数据库中间件?

  A：是的。

- Q：PolarDB是否支持表的分区?

  A：支持。

- Q：PolarDB是否已经自动包含了分区机制?

  A：PolarDB在存储层做了分区，对用户透明，无感知。

- Q：对比原生MySQL，PolarDB单表最多支持存储多少数据量?

  A：PolarDB不限制单表大小，但单表大小受磁盘空间大小限制，详情请参见使用限制。

费用

- Q：PolarDB的费用都包含哪些?

  A：包含存储空间、计算节点、备份（附赠免费额度）、SQL洞察（可选），详情请参见规格与定价。

- Q：收费的存储空间都包含哪些内容?

  A：包含数据库表文件、索引文件、undo日志文件、Redo日志文件、slowlog文件及少量的系统文件，详情请参见规格与定价。

- Q：PolarDB的存储包怎么用？

  A：购买的包年包月或按量付费的集群，均可使用存储包抵扣存储费用。例如您有3个存储容量均为40 GB的集群（即总容量为120 GB），这3个集群可以共享一个100 GB的存储包，多出的20 GB则按量计费，详情请参见购买存储包。

集群访问（读写分离）

- Q：如何实现PolarDB的读写分离？

  A：只需在应用程序中使用集群地址，即可根据配置的读负载节点实现读写分离，详情请参见创建自定义集群地址。

- Q：一个PolarDB集群内最多可以支持多少个只读节点？

  A：PolarDB采用分布式集群架构，一个集群包含一个主节点和最多15个只读节点（至少一个，用于保障高可用）。

- Q：多个只读节点间负载不均衡的原因是什么？

  A：只读节点间负载不均衡的原因有只读节点连接数较少、自定义集群地址分配时未包括某个只读节点等。

- Q：造成主节点负载高或低的原因是什么？

  A：造成主节点（主库）负载高的原因有直连主地址、主库接受读请求、存在大量的事务请求、主从复制延迟高导致请求被路由到主库、只读节点异常导致读请求被路由到主库等。

  而主节点负载较低的原因可能是主库开启了不接受读选项。

- Q：怎么降低主节点的负载？

  A： 您可以使用如下几种方式降低主节点负载：

  - 使用集群地址来连接PolarDB集群，详情请参见修改集群地址。
  - 如果由于事务较多导致主节点压力大，您可以通过控制台打开事务拆分功能，把事务中的部分查询路由到只读节点，详情请参见高级选项-事务拆分。
  - 如果由于复制延迟导致请求被路由到主库，您可以考虑降低一致性等级（如使用最终一致性），详情请参见高级选项-一致性级别。
  - 主库接受读请求，可能也会导致主库负载高，您可以通过控制台关闭主库接受读功能，减少读请求被路由到主库。

- Q：为什么读不到刚插入的数据？

  A：该问题可能是由于一致性级别的配置导致的，PolarDB的集群地址支持如下几种一致性级别：

  - 最终一致性：不论是同一会话（连接）或不同会话，最终一致性都不保证读能够马上读到刚插入的数据。
  - 会话一致性：一定能够读到同一会话插入之后的读数据。

  > **说明** 一致性等级越高，性能越差，对主库的压力越大，请谨慎选择。对于大多数应用场景会话一致性能够保证业务正常工作，对于少数有强一致性的需求的语句，可以通过Hint `/* FORCE_MASTER */` 来实现，详情请参见PolarDB PostgreSQL一致性级别。

- Q：如何强制SQL到主节点执行？

  使用集群地址时，在SQL语句前加上 `/* FORCE_MASTER */` 或 `/* FORCE_SLAVE */` ，即可强制指定这条SQL的路由方向，详情请参见自定义路由-Hint。

- `/* FORCE_MASTER */` 强制请求被路由到主库。该用法可以用于解决少数一致性要求较高的读请求的场景。
- `/* FORCE_SLAVE */` 强制请求被路由到从库。该用法可以用于解决少数PolarDB代理由于保证正确性，要求特殊语法被路由到主库的场景（比如存储过程的调用，multistatement的使用等语句默认是会被路由到主库）。

> 说明
> - 若您需要通过MySQL官方命令行执行上述Hint语句，请在命令行中加上-c参数，否则该Hint会被MySQL官方命令行过滤导致Hint失效，具体请参见MySQL官方命令行。
> - Hint的路由优先级最高，不受一致性级别和事务拆分的约束，使用前请进行评估。
> - Hint语句里不要有改变环境变量的语句，例如 `/*FORCE_SLAVE*/ set names utf8;` 等，这类语句可能导致查询结果非预期。

- Q：是否可以给不同的业务分配不同的地址？不同地址间是否可以达到隔离的效果？

  A：您可以创建多个自定义地址给不同的业务使用，若底层节点不同则自定义地址间可同时具备隔离的效果，不会互相影响。关于如何创建自定义地址，详情请参见修改集群地址。

- Q：如果有多个只读节点，如何为其中某个只读节点单独创建单节点地址？

  A：仅当集群地址读写模式为只读且集群内拥有三个及以上节点时，才支持创建单节点地址，详细操作步骤请参见修改集群地址。

> 警告 创建单节点地址后，当此节点故障时，该地址可能会出现最多1小时不可用的情况，请勿用于生产环境。

- Q：一个集群内最多允许创建多少个单节点地址？

  A：如果您的集群内有3个节点，则只允许为其中1个只读节点创建单节点地址；若集群内有4个节点，则允许为其中2个只读节点创建各自的单节点地址，以此类推。

- Q：只用了主地址，但是发现只读节点也有负载，是否主地址也支持读写分离？

  A：主地址不支持读写分离，始终只连接到主节点。只读节点有少量QPS是正常现象，与主地址无关。

管理与维护

- Q：主节点（主）与只读节点（备）是否存在复制延迟？

  A：是，它们之间存在毫秒级延迟。

- Q：什么情况下会导致复制延迟增大？

  A：出现如下情况时会致复制延迟增大：

  - 主节点写入负载高，产生了过多的Redo日志，导致只读节点来不及应用。
  - 只读节点负载过高，抢占了过多原本属于应用Redo日志的资源。
  - I/O出现瓶颈，导致读写Redo日志过慢。

- Q：存在复制延迟的情况下，如何保证查询的一致性？

  A：您可以使用集群地址并为其选择合适的一致性级别。目前一致性从高到低分别为会话一致性和最终一致性，详情请参见修改集群地址。

- Q：单节点故障的情况下是否可以保证RPO为0？

  A：可以。

- Q：升级规格配置（比如从2核8 GB升级到4核16 GB）后端是怎么实现的？对业务有什么影响？

  A：PolarDB的代理（Proxy）和数据库节点（Node）均需要升级到最新的配置，采用多个节点滚动升级的方式尽量减少对业务的影响。目前每次升级大概需要10~15分钟，对业务的影响时间不超过30秒，期间可能会产生1~3次连接闪断，详情请参见变更配置。

- Q：添加节点要多久？是否会影响业务？

  A：每增加一个节点需要5分钟，对业务无影响。关于如何添加节点，详情请参见增加只读节点。

  > **说明** 新增只读节点之后新建的读写分离连接会转发请求到该只读节点。新增只读节点之前建立的读写分离连接不会转发请求到新增的只读节点，需要断开该连接并重新建立连接，例如，重启应用。

- Q：升级到最新修订版本需要多久？是否会影响业务？

  A：PolarDB采用多节点滚动升级的方式尽量减少对业务的影响，目前每次升级大概需要10~15分钟，对业务的影响时间不超过30秒，期间可能会产生1~3次连接闪断，详情请参见小版本升级。

- Q：如何进行故障自动切换？

  A：PolarDB采用双活（Active-Active）的高可用集群架构，可读写的主节点和只读节点之间自动进行故障切换（Failover），系统自动选举新的主节点。PolarDB每个节点都有一个故障切换（Failover）优先级，决定了故障切换时被选举为主节点的概率高低。当多个节点的优先级相同时，则有相同的概率被选举为主节点，详情请参见主备切换。

备份与恢复

- Q：PolarDB采用什么备份方式？

  A：PolarDB采用快照（Snapshot）的方式进行备份，详情请参见备份数据。

- Q：数据库恢复的速度如何？

  A：目前，基于备份集（快照）进行恢复（克隆）的速度是40分钟/TB。如果是恢复到任意时间点，则需要包含应用Redo日志的时间，这部分的恢复速度大概是20~70秒/GB，整个恢复时间是这两部分之和。

性能和容量

- Q：表个数上限是多少？表个数到多少时有可能会引起性能下降？

  A：表个数的上限受文件数量限制，详情请参见使用限制。

- Q：表分区能够提高PolarDB的查询性能吗？

  A：通常来说，如果查询SQL能够落在某个分区内，是可以提升性能的。

- Q：PolarDB是否支持创建1万个数据库？数据库个数上限是多少？

  A：PolarDB支持创建1万个数据库。数据库个数上限受文件数量限制，详情请参见使用限制。

- Q：IOPS是怎么限制和隔离的？是否会出现多个PolarDB集群节点的I/O争抢？

  A：PolarDB集群的每个节点根据规格大小设置IOPS，每个节点之间IOPS独立隔离，互不影响。

- Q：只读节点的性能变慢是否会影响主节点？

  A：只读节点的负载过高、复制延迟增高时，可能会少量增加主节点的内存消耗。

- Q：打开SQL洞察（全量SQL日志审计），对性能有什么影响？

  A：无影响。

- Q：PolarDB使用了什么高速网络协议？

  A：PolarDB的数据库计算节点和存储节点之间，以及存储数据多副本之间，都使用了双25Gbps RDMA技术，提供低延迟、高吞吐的强劲I/O性能。

- Q：PolarDB外网连接的带宽上限是多少？

  A：PolarDB外网连接的带宽上限为10Gbit/s。

- Q：重启节点需要的时间很长怎么办？

  A：当您的集群中存在的文件数量越多，节点重启需要的时间会越长。此时您可以通过修改innodb_fast_startup参数值为*ON*来加速重启，关于如何修改参数，请参见设置集群参数。

  > **说明** 仅 8.0集群支持配置该参数。

6.服务等级协议

版本生效日期：2018年4月1日

本服务等级协议（Service Level Agreement，以下简称 “SLA”）规定了阿里云向客户提供的云数据库PolarDB（简称“PolarDB”）的服务可用性等级指标及赔偿方案。

1. 定义
   i. 服务周期：一个服务周期为一个自然月。
   ii. 服务周期总分钟数：服务周期内的总天数╳24（小时）╳60（分钟）计算。
   iii. 服务不可用分钟数：当某一分钟内，客户所有试图与指定的PolarDB集群建立连接的连续尝试均失败，则视为该分钟内该PolarDB集群服务不可用。 在一个服务周期内PolarDB集群不可用分钟数之和即服务不可用分钟数。
   iv. 月度服务费用：客户在一个自然月中就单个PolarDB集群所支付的服务费用总额，如果客户一次性支付了多个月份的服务费用，则将按照所购买的月数分摊计算月度服务费用。
2. 服务可用性
   i. 服务可用性计算公式服务可用性以单个集群为维度，按照如下方式计算：服务可用性 =（（服务周期总分钟数 - 服务不可用分钟数）/ 服务周期总分钟数）× 100%
   ii. 服务可用性承诺PolarDB标准版服务可用性不低于99.99%，单节点普惠版服务可用性不低于99.95%，如PolarDB未达到前述可用性承诺，客户可以根据本协议第3条约定获得赔偿。
   iii. 除外情形因下述原因导致的服务不可用的时长不计入服务不可用时间：
      a. 阿里云预先通知客户后进行系统维护所引起的，包括割接、维修、升级和模拟故障演练。
      b. 任何阿里云所属设备以外的网络、设备故障或配置调整引起的。
      c. 客户的应用程序或数据信息受到黑客攻击而引起的。
      d. 客户维护不当或保密不当致使数据、口令、密码等丢失或泄漏所引起的。
      e. 客户的疏忽或由客户授权的操作所引起的。
      f. 客户未遵循阿里云产品使用文档或使用建议引起的。
      g. 不可抗力引起的。
3. 赔偿方案
   i. 赔偿标准每个PolarDB集群按单集群月度服务可用性，按照下表中的标准计算赔偿金额，赔偿方式仅限于用于购买PolarDB产品的代金券，且赔偿总额不超过未达到服务可用性承诺当月客户支付的月度服务费用的25%（不含用代金券抵扣的费用）。

服务可用性	赔偿代金券金额
■ 标准版：99.90% ≤ 服务可用性 < 99.99% ■ 单节点普惠版：99.50% ≤ 服务可用性 < 99.95%	月度服务费用的10%
■ 标准版：服务可用性 < 99.90% ■ 单节点普惠版：服务可用性 < 99.50%	月度服务费用的25%

   ii. 赔偿申请时限客户可在每个自然月第五（5）个工作日后对上个月没有达到服务可用性承诺的集群提出赔偿申请。赔偿申请最迟不应晚于PolarDB未达到服务可用性承诺的相关月份结束后两（2）个月内提出。

4. 其他阿里云有权对本SLA条款作出修改。如本SLA条款有任何修改，阿里云将提前30天以网站公示或发送邮件的方式通知您。如您不同意阿里云对SLA所做的修改，您有权停止使用PolarDB，如您继续使用PolarDB，则视为您接受修改后的SLA。