云和恩墨
ENMOTECH

openGauss 最佳参数实践

高云龙@云和恩墨

参数查看方式

参数文件

cat $DATADIR/postgresql.conf

系统视图

select name,setting from pg_settings;

show 命令

show + 参数名

gsql -p $PORT postgres -c “show all” | grep -i 参数名

参数分类

参数类型	说明
INTERNAL	固定参数，无法修改
POSTMASTER	数据库服务端参数，重启生效
SIGHUP	数据库全局参数，重启或加载生效
BACKEND	会话连接参数，重启或加载生效
SUSET	管理员参数，重启或加载生效
USERSET	普通用户参数，重启或加载生效

编辑postgresql.conf文件

命令行修改

```
ALTER SYSTEM SET paraname TO value;
ALTER DATABASE dbname SET paraname TO value;
ALTER USER username SET paraname TO value;
SET paraname TO value;
```

gs_guc 命令修改

```
gs_guc set -D datadir -c "paraname=value”
gs_guc set -N all -I all -c "paraname=value"
```

加载生效

```
gs_ctl reload -D $DATADIR
select pg_reload_conf();
gs_guc reload ……
```

重启生效

```
gs_ctl restart -D $DATADIR
gs_om -t restart
```

gs_guc 介绍

gs_guc 命令既可以修改数据库参数，也可以修改pg_hba.conf 访问控制
gs_guc 命令不但可以修改单实例，还可以修改集群内所有节点

常用命令

set: 表示只修改配置文件中的参数
check: 表示只检查配置文件中的参数
reload: 表示修改配置文件中的参数，同时发送信号量给数据库进程，使其重新加载配置文件

-N: 需要设置的主机名称
-I : 需要设置的实例名称
-D: 不能与 -I 一起使用

-c parameter=value: 要设定的openGauss配置参数的名称和参数值
-c parameter: 当进行check操作时，表示需要检查的参数名称；
当进行set/reload操作时，表示需要恢复为数据库默认值的参数名称
-h host-auth-policy: 指定需要在“pg_hba.conf”增加的客户端认证策略

实践参数

参数名称	建议值	备注
max_process_memory	Physical memory * 80%	实例节点可用总内存
shared_buffers	Physical memory * 25%	影响数据读写效率
work_mem	64MB	影响order by， distinct和merge joins
maintenance_work_mem	2GB	影响VACUUM、CREATE INDEX， 可能消耗autovacuum_max_workers倍内存
max_connections	3000	每个空连接大约消耗5MB的内存， 应尽量减少idle连接
session_timeout	0	连接不进行任何操作的断开时间
statement_timeout	0	语句执行超时时间
password_encryption_type	1	0:md5 、1: md5 + sha256 、2: sha256
password_effect_time	0	默认90天，如需重用旧密码，需要结合参数 重用天数：password_reuse_time 重用次数：password_reuse_max

实践参数

参数名称	建议值	备注
wal_level	logical	minimal、archive、hot_standby、logical
wal_keep_segments	1024	pg_xlog保留的最小数量，具体保留数量应根据使用场景来设定
max_size_for_xlog_prune	104857600	enable_xlog_prune打开时，若备机断连且xlog日志大小大于此阈值，则回收日志
synchronous_commit	on	级别越高，数据安全性越高，对性能影响越大
archive_mode	on	将wal日志归档，可结合全量物理备份实现pitr
archive_dest	目录地址	替代archive_command参数，与数据目录分开存储
checkpoint_segments	1024	checkpoint_timeout周期内保留的WAL日志段文件数量
checkpoint_completion_target	0.9	检查点完成的目标，值越大对磁盘压力越平滑
max_replication_slots	32	物理流复制槽数+逻辑复制槽数
recovery_max_workers	4	重放并发数，并发数高，重放速度快，对磁盘io要求比较高

实践参数

参数名称	建议值	备注
sync_config_strategy	none_node	集群内所有节点的参数值是否保持一致
most_available_sync	on	同步备库故障后，同步级别降级
catchup2normal_wait_time	0	单同步备机情况下，控制备机数据追赶（catchup）阻塞主机的最长时间，默认值-1，一直等待
enable_wdr_snapshot	on	生成wdr报告的基础，关闭此参数需要人工操作快照相关的表
autovacuum	on	自动维护线程，根据维护参数因子，数据库自动清理，统计数据库对象信息
autovacuum_max_workers	5	并发越高，维护操作越快，对服务器资源需求越多
standby_shared_buffers_fraction	1	备库所在服务器使用shared_buffers内存缓冲区大小的比例
local_syscache_threshold	32MB	控制session动态内存大小
enable_slot_log	on	是否开启逻辑复制槽主备同步特性
instr_unique_sql_count	50000	控制系统中unique sql信息实时收集功能

云和恩墨 ENMOTECH

实践参数

参数名称	建议值	备注
logging_collector	on	数据库操作日志收集功能
log_filename	postgresql_%d.log	如果需要循环覆盖需要结合log_truncate_on_rotation参数
log_line_prefix	%m %u %d %r %p	日志信息前缀，时间戳、用户、数据库、客户端ip、端口号
log_truncate_on_rotation	on	日志循环前是否清理文件内容
log_hostname	off	是否解析ip为主机名
log_min_duration_statement	1000	记录慢SQL的阈值
log_statement	mod	控制记录SQL语句，none，ddl，mod，all
log_lock_waits	on	是否记录锁等超时相关信息
lockwait_timeout	60s	锁等待超时时间
update_lockwait_timeout	60s	并发更新同一行时单个锁的最长等待时间
deadlock_timeout	1s	死锁检测超时

初始化参数

```
gs_guc set -N all -I all -c 'max_connections = 4096';
gs_guc set -N all -I all -c 'allow_concurrent_tuple_update = true';
gs_guc set -N all -I all -c 'audit_enabled = off';
gs_guc set -N all -I all -c 'checkpoint_segments = 1024';
gs_guc set -N all -I all -c 'enable_alarm = off';
gs_guc set -N all -I all -c 'enable_codegen = false';
gs_guc set -N all -I all -c 'full_page_writes  = off';
gs_guc set -N all -I all -c 'max_files_per_process = 100000';
gs_guc set -N all -I all -c 'max_prepared_transactions = 2048';
gs_guc set -N all -I all -c 'max_process_memory = 200GB';
gs_guc set -N all -I all -c 'shared_buffers = 100GB';
gs_guc set -N all -I all -c 'use_workload_manager = off';
gs_guc set -N all -I all -c 'wal_buffers = 1GB';
gs_guc set -N all -I all -c 'synchronous_commit = on';
gs_guc set -N all -I all -c 'maintenance_work_mem = 2GB';
gs_guc set -N all -I all -c 'autovacuum = on';
gs_guc set -N all -I all -c 'checkpoint_timeout = 15min';
gs_guc set -N all -I all -c 'enable_thread_pool = off';
gs_guc set -N all -I all -c 'enable_double_write = on';
gs_guc set -N all -I all -c 'enable_incremental_checkpoint = on';
gs_guc set -N all -I all -c 'enable_opfusion = on';
gs_guc set -N all -I all -c 'advance_xlog_file_num = 10';
gs_guc set -N all -I all -c 'plog_merge_age = 0';
gs_guc set -N all -I all -c 'session_timeout = 0';
```

参数验证

```
with cte_settings_base as(
   SELECT name,setting
FROM (VALUES
('max_connections','3000'),
('wal_level','logical'),
('full_page_writes','off'),
('wal_log_hints','off'),
('synchronous_commit','on'),
('wal_keep_segments','1024'),
('archive_mode','on'),
('archive_dest','/ogarchive'),
('log_checkpoints','on'),
('max_wal_senders','16'),
('recovery_max_workers','4'),
('most_available_sync','on'),
('checkpoint_segments','1024'),
('checkpoint_completion_target','0.9'),
('password_encryption_type','1'),
('session_timeout','0'),
('enable_alarm','off'),
('enable_codegen','off'),
('enable_wdr_snapshot','on'),
('sync_config_strategy','none_node')
) AS t (name,setting)
)select t1.name,t1.setting as base,t2.setting
   from cte_settings_base t1
   join pg_settings t2
on t1.name=t2.name and t1.setting!=t2.setting;
```

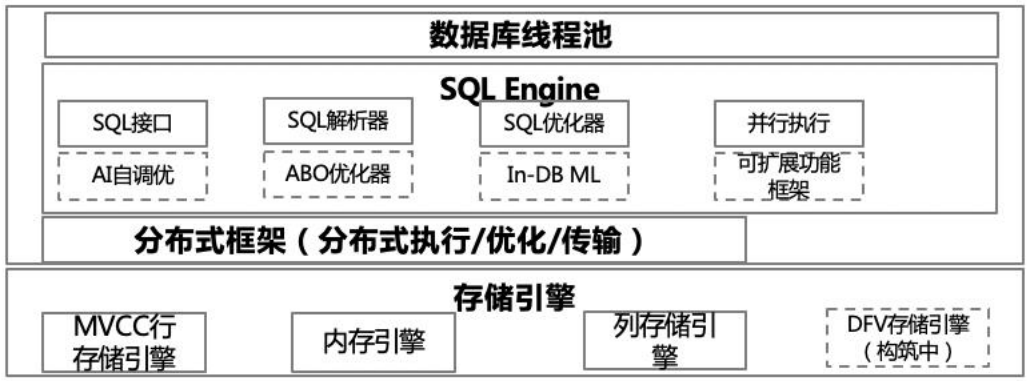
```
            name             |    base    |   setting
-----------------------------+------------+-------------
 archive_dest                | /ogarchive |
 archive_mode                | on         | off
 checkpoint_completion_target | 0.9        | 0.5
 checkpoint_segments         | 1024       | 64
 enable_alarm                | off        | on
 enable_codegen              | off        | on
 enable_wdr_snapshot         | on         | off
 full_page_writes            | off        | on
 log_checkpoints             | on         | off
 max_connections             | 3000       | 5000
 most_available_sync         | on         | off
 password_encryption_type    | 1          | 2
 session_timeout             | 0          | 10
 sync_config_strategy        | none_node  | all_node
 synchronous_commit          | on         | off
 wal_keep_segments           | 1024       | 16
 wal_level                   | logical    | hot_standby
 wal_log_hints               | off        | on
(18 rows)
```

云和恩墨
ENMOTECH

openGauss 与 PostgreSQL

高云龙@云和恩墨

体系结构



openGauss Kernel架构

PostgreSQL架构

关键差异化因素		openGauss	PostgreSQL
宏观架构	执行模型	线程模型：动态分配执行线程，支持1万并发	进程模型：进程执行模型，一个链接一个进程，小于1000并发
	内存模型	进程内内存被多线程共享，内存安全性好	多进程共享内存，内存安全性弱；动态扩展难
事务处理	并发控制	事务支持CSN快照，procArray免锁高并发	事务ID回卷，长期运行性能因为ID回收周期大幅波动
	日志和检查点	增量checkpoint，性能波动<5%	全量checkpoint，性能短期波动>15%
	鲲鹏NUMA	NUMA多核优化，单机两路性能TPMC 大于150w	NUMA多核能力弱，单机两路性能TPMC <60w
数据组织	多引擎	行存、列存、内存引擎，在研DFV存储引擎	仅支持行存
SQL引擎	优化器	支持CBO，吸收工行等大型企业场景优化能力	支持CBO，复杂场景优化能力一般
	SQL解析	ANSI/ISO标准SQL92、SQL99和SQL2003和企业扩展包	ANSI/ISO标准SQL92、SQL99和SQL2003

运维工具

		PostgreSQL	MogDB/openGauss
备份恢复	逻辑	pg_dump、pg_dumpall、pg_restore	gs_dump、gs_dumpall、gs_restore
	物理	pg_basebackup、pg_probackup	gs_basebackup、gs_probackup
		pg_rman	BRM
		pgbackreset	
		barman	
监控工具		postgresql_exporter	opengauss_exporter
高可用工具	手动	Promote命令	支持switchover、failover命令
	自动	patroni repmgr pacemaker+corosync pgpool	MogHA工具

特性对比

	PostgreSQL	MogDB/openGauss
支持系统	市面上的系统基本都可以安装	openEuler on arm、centos on x86、 kylin on arm、红旗 on x86、 其他系统需要适配
安装方式	源码安装、rpm安装	源码安装、gs_om工具安装
内存限制	-	参数控制
兼容性	-	兼容oracle、postgresql、mysql
XID	2^32	2^64
CHECKPOINT	全量	全量/增量
磁盘保护	full page write	double write
表存储类型	行存	行存、列存、内存
线程池	-	支持
AI特性	-	支持

数据库对象对比

		PostgreSQL	MogDB/openGauss
扩展		支持扩展丰富	需要适配
分区表	范围分区	有default分区	间隔分区自动添加分区
	hash分区	分区数量无限制	最多64个分区
	list分区	分区数量，分区键无限制	最多64个分区，64个分区键
分区索引		普通索引	本地索引、全局索引
字符类型		n表示字符	兼容pg模式，n表示字符
字符字段长度		1T	10MB
json		json、jsonb	json

checkpoint

检查点是事务日志序列中的一个点，在该点时刻的所有数据文件的修改信息，所有脏数据页都已经被刷写到磁盘。在崩溃恢复过程中通过检查最新的检查点记录来决定从日志中哪个点开始REDO操作。

PG中检查点在每checkpoint_timeout秒开始，或快要超过checkpoint_segments/max_wal_size时开始，全量检查点刷写所有脏数据页到磁盘，对I/O负载能力要求比较高

增量检查点

enable_incremental_checkpoint = on
pagewriter_thread_num=2
pagewriter_sleep=2000ms
dirty_page_percent_max=0.9（1.0.1版本）
pagewriter_threshold=818（1.0.0版本）

全量checkpoint

执行checkpoint命令
数据库关机
做数据库备份

double write

在操作系统崩溃过程中可能磁盘页面只写入了一部分内容，从而导致在同一个页面中包含新旧数据的混合。在崩溃后进行恢复期间，由于在WAL日志中存储的数据变化信息不够完整，无法完全恢复该数据页，数据库不可用或数据丢失，PostgreSQL打开full_page_writes参数，在chenckpoint后，会在第一次修改页面时，把完整的页面影像保存下来，保证在恢复期间页面可以被正确还原，代价是增加了写入WAL日志的数据量。

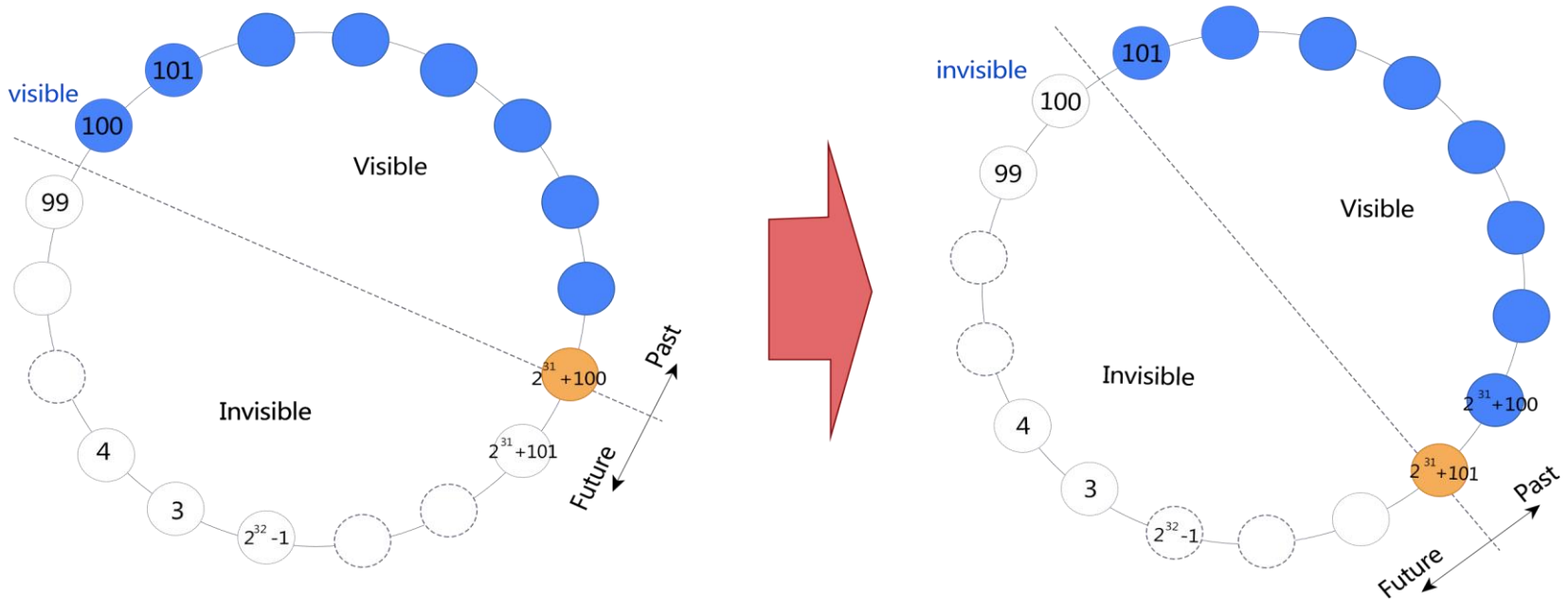
openGauss在保留PG的full_page_writes特性的基础上，新增类似mysql的双写的功能来替换full_page_writes参数所带来的影响，在写数据块的同时将脏页也写到一个共享的双写空间里，如果这时发生故障，需要恢复时会从双写空间里找到完整的数据页进行恢复。

要开启enable_double_write参数需要和增量检查点一起配合使用：
enable_incremental_checkpoint=on
enable_double_write=on

修改这两个参数，需要重启数据库集群。

事务ID

PostgreSQL中xid（事务id）的数量大约42.9亿个，是一个事物环的形态，这个事物环分成两半，一半是未来事物号，一半是过去事物号，事物号通过freeze来回卷循环使用。



事务ID

当数据库执行vacuum freeze失败，会有xid用尽的风险，当数据库中最旧 XID 和回卷点之间剩余不足1千万个事务号时会出现需要vacuum的的警告

```
WARNING:  database "mydb" must be vacuumed within 177009986 transactions
HINT:  To avoid a database shutdown, execute a database-wide VACUUM in "mydb".
```

当距离回卷点只剩下1百万个事务时，数据库系统将会关闭并且拒绝开始任何新的事务

```
ERROR:  database is not accepting commands to avoid wraparound data loss in database "mydb"
HINT:  Stop the postmaster and vacuum that database in single-user mode.
```

这时只能关闭数据库，然后以单用户模式进入相应的数据库执行vacuum操作

```
/bin/postgres --single -D /data/pgdata mydb
backend> vacuum;
```

事务ID

openGauss数据库改进的一个特性是将postgresql的xid限制由32位提升到了64位，这也就意味着opengauss的xid永远用不完，也就不会涉及到事务回卷的情况发生

```
[omm@ecs-be61-0002 ~]$ gsql -Uomm -p9832 postgres -r
gsql ((openGauss 1.0.0 build 197f217c) compiled at 2020-09-08 08:50:40 commit 0 last mr  )
Non-SSL connection (SSL connection is recommended when requiring high-security)
Type "help" for help.

postgres=# select version();
                                                version
----------------------------------------------------------------------------------------------------------------------------------------
 (openGauss 1.0.0 build 197f217c) compiled at 2020-09-08 08:50:40 commit 0 last mr   on x86_64-unknown-linux-gnu, compiled by g++ (GCC) 8.2.0, 64-bit
(1 row)

postgres=# select txid_current();
 txid_current
--------------
   4580988517
(1 row)

postgres=# select xmin,age(xmin),* from vacuum_test_1;
xmin | age | id
-----+-----+----
   2 |  -1 | 1
(1 row)

postgres=#
```

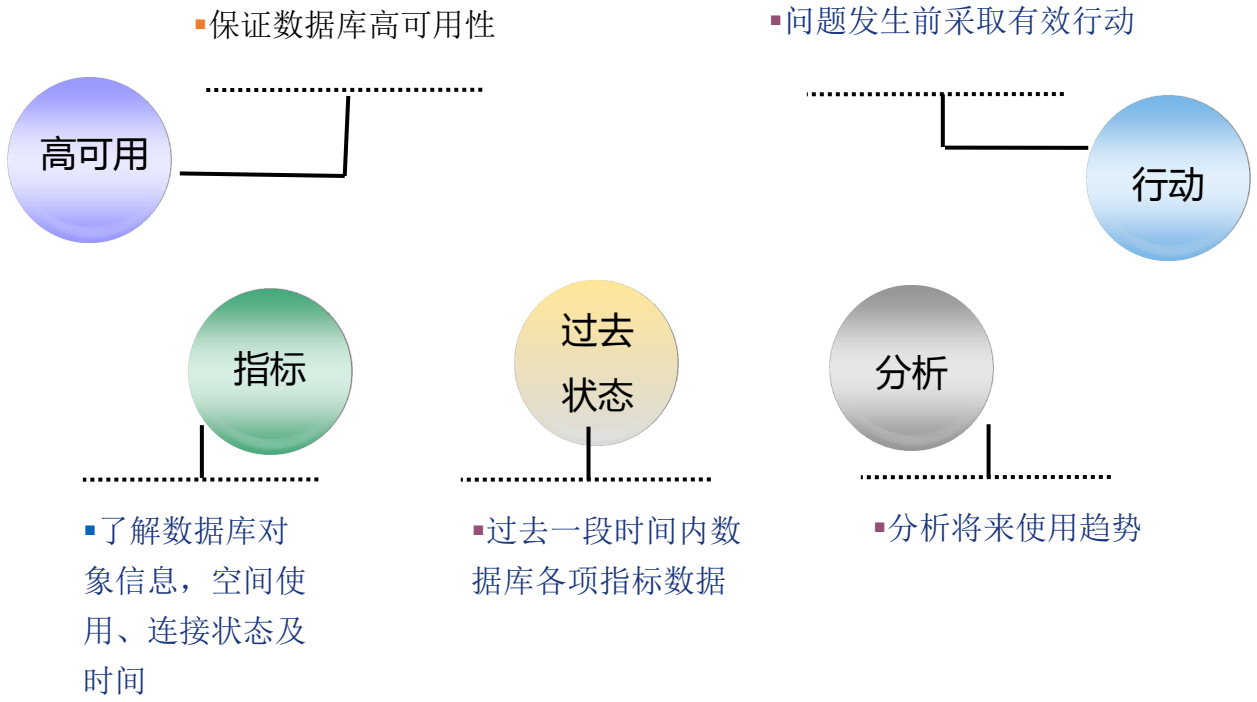
测试方法参考： https://www.modb.pro/db/31736

云和恩墨
ENMOTECH

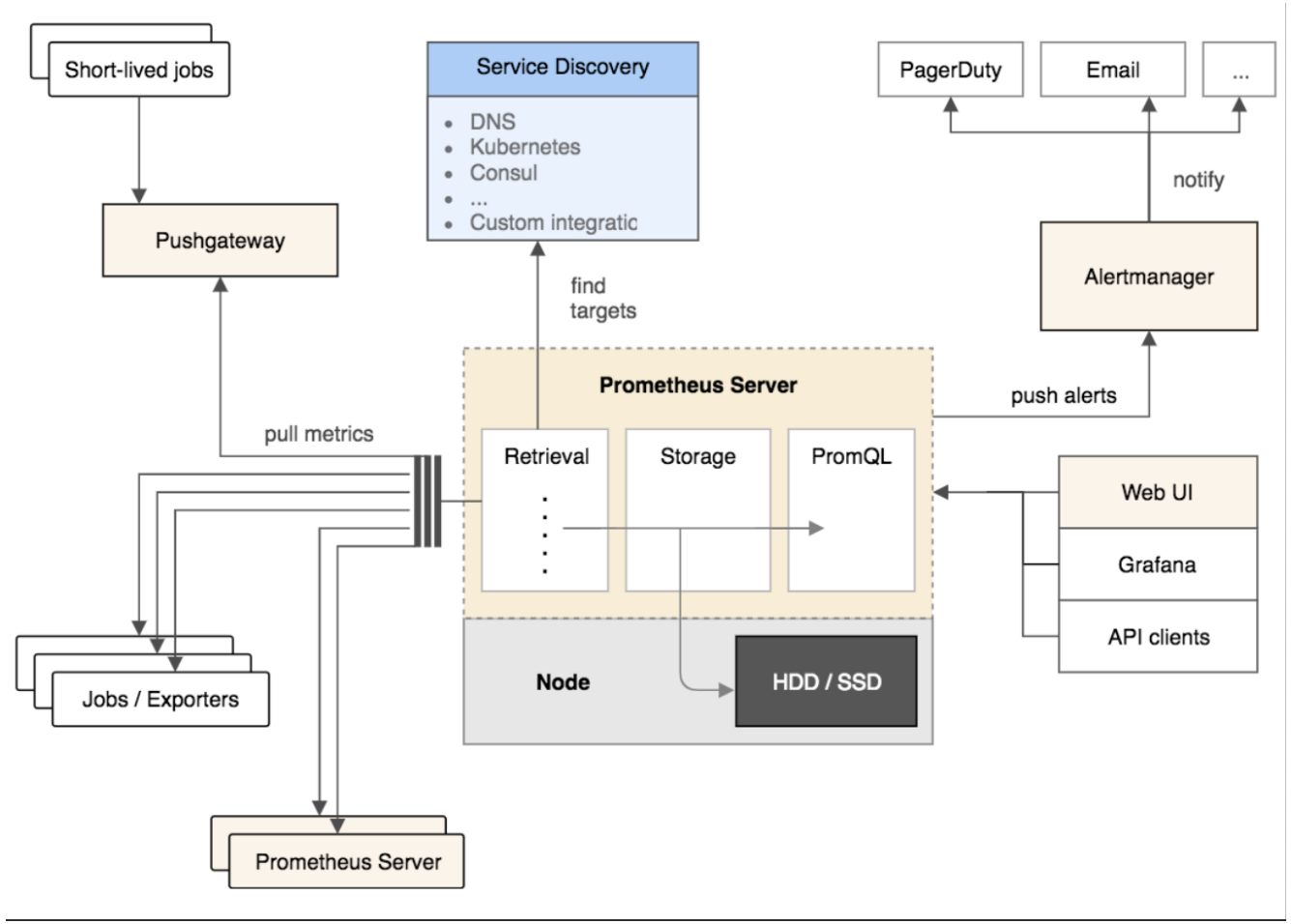
openGauss运维监控工具

高云龙@云和恩墨

监控


保证数据库高可用性
高可用
问题发生前采取有效行动
行动
指标
了解数据库对象信息，空间使用、连接状态及时间
过去状态
过去一段时间内数据库各项指标数据
分析
分析将来使用趋势

Prometheus + Grafana



云和恩墨
ENMOTECH

opengauss_exporter

opengauss_exporter 是由云和恩墨开发，为MogDB/openGauss数据库量身打造的数据采集工具，配合监控报警框架prometheus + grafana实时展示数据库信息，为MogDB/openGauss数据库的平稳运行保驾护航。

源码地址: https://gitee.com/enmotech/opengauss_exporter

特点:

- 支持所有版本MogDB/openGauss数据库
- 支持监控数据库集群
- 支持集群内主备角色判断
- 支持自动发现数据库
- 支持自定义查询query
- 支持在线加载配置文件
- 支持配置线程并发数
- 支持数据采集信息缓存

数据库配置

修改参数

opengauss默认加密方式是sha256,需要改成md5的密码加密方式
password_encryption_type=1

创建用户

CREATE USER opengauss_exporter WITH PASSWORD 'opengauss_exporter123' MONADMIN;

访问控制

修改pg_hba.conf文件，不需要加载就可以生效
host dbname opengauss_exporter x.x.x.x/32 md5

云和恩墨 ENMOTECH

监控指标文件

```yaml
pg_setting:
  name: pg_setting
  desc: Important postgres setting entries that must kept same on entire cluster
  query:
    - name: pg_setting
      sql: |-
        SELECT current_setting('max_connections')           AS max_connections,
               current_setting('max_prepared_transactions') AS max_prepared_transactions,
               current_setting('max_replication_slots')     AS max_replication_slots,
               current_setting('max_wal_senders')           AS max_wal_senders,
               current_setting('max_locks_per_transaction') AS max_locks_per_transaction,
               current_setting('block_size')                AS block_size,
               CASE current_setting('wal_log_hints') WHEN 'on' THEN 1 ELSE 0 END AS wal_log_hints;
      version: '>=0.0.0'
      timeout: 1
      ttl: 60
      status: enable
      dbRole: ""
  metrics:
    - name: max_connections
      description: number of concurrent connections to the database server
      usage: GAUGE
    - name: max_prepared_transactions
      description: maximum number of transactions that can be in the prepared state simultaneously
      usage: GAUGE
    - name: max_replication_slots
      description: maximum number of replication slots
      usage: GAUGE
    - name: max_wal_senders
      description: maximum number of concurrent connections from standby servers
      usage: GAUGE
    - name: max_locks_per_transaction
      description: no more than this many distinct objects can be locked at any one time
      usage: GAUGE
    - name: block_size
      description: pg page block size, 8192 by default
      usage: GAUGE
    - name: wal_log_hints
      description: whether wal_log_hints is enabled, 1 enabled 0 disabled
      usage: GAUGE
  status: enable
  ttl: 5
  timeout: 1
  public: true
```

启动exporter

配置环境变量

将以下配置添加到～/.bashrc 文件，也可以在每次执行命令前执行
export DATA_SOURCE_NAME="host=x.x.x.x user=opengauss_exporter
password=opengauss_exporter123 port=9832 dbname=og_pg sslmode=disable”

export DATA_SOURCE_NAME=“postgresql://login:password@hostname1:port1/dbname,
postgresql://login:password@hostname2:port2/dbname"

启动exporter

将编译好的二进制文件opengauss_exporter 放到目录/opt/opengauss_exporter/下，以nohup的方式启动：

nohup /opt/opengauss_exporter/opengauss_exporter --config=“/opt/opengauss_exporter/default_queries.yaml”
--auto-discover-databases --exclude-databases=“template0,template1” --parallel=5 --log.level=debug 2>&1 &

示例

确保防火墙关闭，如果防火墙打开，则需要开通9187端口 在浏览器输入服务器ip及exporter 端口号，如：http://127.0.0.1:9187/metrics 展示效果如下：

```
# HELP go_gc_duration_seconds A summary of the pause duration of garbage collection cycles.
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 6.0186e-05
go_gc_duration_seconds{quantile="0.25"} 8.2341e-05
go_gc_duration_seconds{quantile="0.5"} 9.2636e-05
go_gc_duration_seconds{quantile="0.75"} 0.000109303
go_gc_duration_seconds{quantile="1"} 0.000297362
go_gc_duration_seconds_sum 0.063502735
go_gc_duration_seconds_count 621
# HELP go_goroutines Number of goroutines that currently exist.
# TYPE go_goroutines gauge
go_goroutines 14
# HELP go_info Information about the Go environment.
# TYPE go_info gauge
go_info{version="go1.15.6"} 1
# HELP go_memstats_alloc_bytes Number of bytes allocated and still in use.
# TYPE go_memstats_alloc_bytes gauge
go_memstats_alloc_bytes 5.91084e+06
# HELP go_memstats_alloc_bytes_total Total number of bytes allocated, even if freed.
# TYPE go_memstats_alloc_bytes_total counter
go_memstats_alloc_bytes_total 1.55915744e+09
# HELP go_memstats_buck_hash_sys_bytes Number of bytes used by the profiling bucket hash table.
# TYPE go_memstats_buck_hash_sys_bytes gauge
go_memstats_buck_hash_sys_bytes 1.524875e+06
# HELP go_memstats_frees_total Total number of frees.
# TYPE go_memstats_frees_total counter
go_memstats_frees_total 2.0212574e+07
# HELP go_memstats_gc_cpu_fraction The fraction of this program's available CPU time used by the GC since the program started.
# TYPE go_memstats_gc_cpu_fraction gauge
go_memstats_gc_cpu_fraction 1.1222569890481846e-05
# HELP go_memstats_gc_sys_bytes Number of bytes used for garbage collection system metadata.
# TYPE go_memstats_gc_sys_bytes gauge
go_memstats_gc_sys_bytes 5.225712e+06
```

采集指标

数据库信息

```
SELECT datname,
  pg_database_size(pg_database.datname) as size_bytes,
  age(datfrozenxid64)          AS age,
  datistemplate              AS is_template,
  datallowconn               AS allow_conn,
  datconnlimit               AS conn_limit,
  datfrozenxid::TEXT::BIGINT as frozen_xid
FROM pg_database
where datname NOT IN ('template0','template1');
```

```yaml
pg_database:
  name: pg_database
  desc: OpenGauss Database size
  query:
    - name: pg_database
      sql: |-
        SELECT datname,
          pg_database_size(pg_database.datname) as size_bytes,
          age(datfrozenxid64)          AS age,
          datistemplate              AS is_template,
          datallowconn               AS allow_conn,
          datconnlimit               AS conn_limit,
          datfrozenxid::TEXT::BIGINT as frozen_xid
        FROM pg_database
        where datname NOT IN ('template0','template1');
      version: '>=0.0.0'
      timeout: 1
      ttl: 60
      status: enable
  metrics:
    - name: datname
      description: Name of this database
      usage: LABEL
    - name: size_bytes
      description: Disk space used by the database
      usage: GAUGE
    - name: age
      description: database age calculated by age(datfrozenxid64)
      usage: GAUGE
    - name: is_template
      description: 1 for template db and 0 for normal db
      usage: GAUGE
    - name: allow_conn
      description: 1 allow connection and 0 does not allow
      usage: GAUGE
    - name: conn_limit
      description: connection limit, -1 for no limit
      usage: GAUGE
    - name: frozen_xid
      description: tuple with xmin below this will always be visable (until wrap around)
      usage: GAUGE
  status: enable
  ttl: 60
  timeout: 1
```

采集指标

当前连接状态统计

```
SELECT datname,state,coalesce(count, 0) AS count,coalesce(max_duration, 0)  AS max_duration,coalesce(max_tx_duration, 0)   AS
max_tx_duration,coalesce(max_conn_duration, 0) AS max_conn_duration
FROM (SELECT d.oid AS database, d.datname, a.state FROM pg_database d,
        unnest(ARRAY ['active','idle','idle in transaction','idle in transaction (aborted)','fastpath function call','disabled']) a(state)
    WHERE d.datname NOT IN ('template0','template1')) base
    LEFT JOIN (
    SELECT datname, state,count(*) AS count,max(extract(epoch from now() - state_change)) AS max_duration,
        max(extract(epoch from now() - xact_start))   AS max_tx_duration,max(extract(epoch from now() - backend_start)) AS
max_conn_duration
    FROM pg_stat_activity WHERE pid <> pg_backend_pid() GROUP BY datname, state
) a USING (datname, state);
```

datname	state	count	max_duration	max_tx_duration	max_conn_duration
oracle	active	0	0	0	0
oracle	idle	0	0	0	0
oracle	idle in transaction	0	0	0	0
oracle	idle in transaction (aborted)	0	0	0	0
oracle	fastpath function call	0	0	0	0
oracle	disabled	0	0	0	0
ogexporter	active	0	0	0	0
ogexporter	idle	0	0	0	0
ogexporter	idle in transaction	0	0	0	0
ogexporter	idle in transaction (aborted)	0	0	0	0
ogexporter	fastpath function call	0	0	0	0
ogexporter	disabled	0	0	0	0
postgres	active	3	2.034902	0	16492.266555
postgres	idle	2	16492.260642	0	16492.260672
postgres	idle in transaction	0	0	0	0
postgres	idle in transaction (aborted)	0	0	0	0
postgres	fastpath function call	0	0	0	0
postgres	disabled	0	0	0	0

采集指标

主备延迟

```
pid              | 139704987416320
client_addr      | 192.168.122.101
application_name | WalSender to Standby
state            | Streaming
sync_state       | Async
lsn              | 126011568
sent_diff        | 0
write_diff       | 0
flush_diff       | 0
replay_diff      | 0
sent_location    | 126011568
write_location   | 126011568
flush_location   | 126011568
replay_location  | 126011568
backend_uptime   | 16732.647734
sync_priority    | 0
```

```
SELECT pid,client_addr,application_name,state,sync_state,lsn,
    lsn - sent_location   as sent_diff,
    lsn - write_location  as write_diff,
    lsn - flush_location  as flush_diff,
    lsn - replay_location as replay_diff,
    sent_location,write_location,flush_location,replay_location,
    backend_uptime,sync_priority
    FROM (
    SELECT pid,
      client_addr,
      application_name,
      state,
      sync_state,
      pg_xlog_location_diff(CASE WHEN pg_is_in_recovery() THEN pg_last_xlog_receive_location() ELSE pg_current_xlog_location() END, '0/0') AS lsn,
      pg_xlog_location_diff(sender_sent_location,'0/0')                    AS sent_location,
      pg_xlog_location_diff(receiver_write_location,'0/0')                  AS write_location,
      pg_xlog_location_diff(receiver_flush_location,'0/0')                  AS flush_location,
      pg_xlog_location_diff(receiver_replay_location,'0/0')                  AS replay_location,
      pg_xlog_location_diff(receiver_replay_location,pg_current_xlog_location())   AS replay_lag,
      extract(EPOCH FROM now() - backend_start) AS backend_uptime,
      sync_priority
  FROM pg_stat_replication) t;
```

采集指标

复制槽

```
select slot_name,
    database                as datname,
    coalesce(plugin,'_') as plugin,
    slot_type,datoid,coalesce(database,'_') as database,
    active,
    coalesce(xmin,'_') as xmin,
    coalesce(catalog_xmin,'_') as catalog_xmin,
    restart_lsn,
    pg_xlog_location_diff(pg_current_xlog_location(),restart_lsn) as delay_lsn,
    dummy_standby,
    pg_xlog_location_diff(restart_lsn ,'0/0'::text)       AS restart_lsn,
    pg_xlog_location_diff(CASE WHEN pg_is_in_recovery() THEN pg_last_xlog_receive_location()
    ELSE pg_current_xlog_location() END , restart_lsn)  AS retained_bytes
from pg_replication_slots;
```

```
slot_name      | dn_6002
datname        |
plugin         | _
slot_type      | physical
datoid         | 0
database       | _
active         | t
xmin           | 0
catalog_xmin   | 0
restart_lsn    | 0/783E208
delay_lsn      | 0
dummy_standby  | f
restart_lsn    | 126083592
retained_bytes | 0
```

锁信息统计

```
SELECT datname, mode, coalesce(count, 0) AS count
      FROM (
        SELECT d.oid AS database, d.datname, l.mode
        FROM pg_database d,unnest(ARRAY
['AccessShareLock','RowShareLock','RowExclusiveLock','ShareUpdateExclusiveLock','ShareLock','ShareRowExclusiveLock','ExclusiveLock','
AccessExclusiveLock']) l(mode)
        WHERE d.datname NOT IN ('template0','template1')) base
      LEFT JOIN (SELECT database, mode, count(1) AS count
            FROM pg_locks
            WHERE database IS NOT NULL GROUP BY database, mode) cnt
      USING (database, mode);
```

采集指标

锁阻塞信息

```
select distinct locker.pid as locker_pid,
       locked.pid as locked_pid,
       coalesce(locker_act.client_addr,'127.0.0.1')::inet as locker_addr,
       coalesce(locked_act.client_addr,'127.0.0.1')::inet as locked_addr,
       locker_act.usename as locker_username,
       locked_act.usename as locked_username,
       locker.mode as locker_mode,
       locked.mode as locked_mode,
       locker.locktype as locker_locktype,
       locked.locktype as locked_locktype,
       locker_act.usename as locker_user,
       locked_act.usename as locker_user,
       (locker_act.xact_start)::text as locker_xact_start,
       (locked_act.xact_start)::text as locked_xact_start,
       (locker_act.query_start)::text as locker_query_start,
       (locked_act.query_start)::text as locked_query_start,
       extract(epoch from now() - locked_act.query_start) as locked_times,
       locker_act.query as locker_query,
       locked_act.query as locked_query
  from pg_locks locked,
       pg_locks locker,
       pg_stat_activity locked_act,
       pg_stat_activity locker_act
 where locker.granted=true
   and locked.granted=false
   and locked.pid=locked_act.pid
   and locker.pid=locker_act.pid
   and locker_act.query not like '%select distinct locker.pid %'
   and locker.pid <> locked.pid
   and locker.mode not like 'AccessShareLock' and locker.mode not like 'ExclusiveLock'
 order by 13 asc limit 10;
```

```
locker_pid          | 139704896648960
locked_pid          | 139704821020416
locker_addr         | 127.0.0.1
locked_addr         | 127.0.0.1
locker_username     | omm
locked_username     | omm
locker_mode         | RowExclusiveLock
locked_mode         | ShareLock
locker_locktype     | partition
locked_locktype     | transactionid
locker_user         | omm
locker_user         | omm
locker_xact_start   | 2021-03-25 18:16:00.353784+08
locked_xact_start   | 2021-03-25 18:16:24.059294+08
locker_query_start  | 2021-03-25 18:16:20.094659+08
locked_query_start  | 2021-03-25 18:16:34.859462+08
locked_times        | 2.642785
locker_query        | update plt set col1='test1' where id=1;
locked_query        | update plt set col1='test2' where id=1;
```

思考

你关注的数据库监控指标有哪些？

希望opengauss_exporter 具有哪些功能点？

云和恩墨
ENMOTECH

数据驱动 成就未来
Make Your Data Dance