Unified Point-in-Time Recovery in the Cloud

Andrey Borodin, Team lead of opensource RDBMS development, Yandex.Cloud
Dmitry Smal, Team lead of Managed MySQL and SQL Server Development, Yandex.Cloud
Managed Service for PostgreSQL
Managed Service for MySQL
Managed Service for MongoDB
Many more DBs
What’s interesting about Yandex.Cloud managed databases?
Yandex
PostgreSQL at Yandex.Cloud

Yandex.Mail

- Hundreds of millions of users
- $10^{12}$ rows, $10^6$ queries per second, ~1 PB of data

Many others services use managed PostgreSQL

- Taxi, carsharing, food delivery, self driving cars, etc.
- Total of 3 million queries per second, 6 PB of space used
MySQL at Yandex.Cloud

Yandex.Direct

› Online advertising network

› Uses Percona build of MySQL at scale

Managed MySQL

› 400+ TB as of 2021
Less is more

- Cloud providers usually sell computing resources.
- We aim to utilize fewer resources for the same workload.
Point-in-time recovery

Backup + changes

› Scalable
› Reliable
› Efficient
› Fast
Scalability

› Data: from 10 GB to 10 TB on a host

› RAM: from 2 GB

› Number of CPUs: from 0.05 to ~100

› Async and parallel whenever possible

› Don’t spill anything on a local disk
HA cluster in the cloud

- Sync/Quorum replication
- Async replication
- Changes archiving
- Backup

Network Object Storage
Resources

› Storage space
Resources

› Storage space

› CPU

› Net bandwidth

› Disk IOPS
Reliability

› Protection from human error via automation and safety checks

› Prevention of data corruption

› Consistency monitoring

› Integration with other systems (HA tool)

› Extensibility and unification of approaches

› Encrypted data in storage
Fast recovery

- OLAP
  - From start to consistency point
- OLTP standby
- To starting streaming replication
- OLTP primary
- Until recovery target and accept of write queries
Unacceptable

› Data locks

Business can’t wait

› Data loss

We call it a “database” after all
WAL-G
v0.2.19

x4m released this on Nov 30, 2020

Notable changes in this release include:

1. Fixes for S3 and GCP storages #656 #756.
2. Add `wal-show` command to get information about wal storage folder.
3. Add `wal-verify` command. It checks the integrity of WAL history starting from the oldest backup available in current timeline history.
4. Add `wal-receive` command. You can use WAL-G as a replica running on another host to ensure RPO=0. This is beta functionality: API may change in the future.
5. Add reverse delta unpack for `backup-fetch (--reverse-unpack)`.
6. Add redundant archives skipping for `backup-fetch (--skip-redundant-tars` flag, designed to work in pair with reverse delta unpack).
7. Add page checksum verification for `backup-push (--verify flag)`.

You can find more about some of these new features in Daniil Zakhlystov's post.

Assets

- `wal-g.linux-amd64.tar.gz` 13.1 MB
- `wal-g.linux-amd64.tar.gz.sha256` 91 Bytes
- Source code (zip)
x4mmm-osx:wal-g x4mmm$
x4mmm-osx:wal-g x4mmm$
x4mmm-osx:wal-g x4mmm$
x4mmm-osx:wal-g x4mmm$
AWS_ENDPOINT=https://storage.yandexcloud.net AWS_ACCESS_KEY_ID=
wIRAxxwOPLI3VrGwtYWL AWS_SECRET_ACCESS_KEY=ne
WALE_S3_PREFIX=s3://wal-g-test/ ./wal-g backup-list
Path:
name last_modified wal_segment_backup_start
base_00000001000000000000000004 2019-02-02T18:39:30Z 00000001000000000000000004
x4mmm-osx:wal-g x4mmm$
x4mmm-osx:wal-g x4mmm$
INFO: 2019/02/02 21:56:42.509465 Doing full backup.
WARNING: 2019/02/02 21:56:42.526434 It seems your archive_mode is not enabled. This will cause inconsistent backup. Please consider configuring WAL archiving.
INFO: 2019/02/02 21:56:42.740377 Walking ...
INFO: 2019/02/02 21:56:42.742571 Starting part 1 ...
INFO: 2019/02/02 21:56:43.112485 Finished writing part 1.
INFO: 2019/02/02 21:56:48.744337 Starting part 2 ...
INFO: 2019/02/02 21:56:48.761395 /global/pg_control
INFO: 2019/02/02 21:56:48.878931 Starting part 3 ...
INFO: 2019/02/02 21:56:48.894990 backup_label
INFO: 2019/02/02 21:56:48.895030 tablespace_map
INFO: 2019/02/02 21:56:49.523658 Uploaded 3 compressed tar Files.
# - Archiving -

archive_mode = on

archive_command = '/usr/bin/envdir /etc/wal-g/envdir /usr/bin/timeout 600 /usr/bin/wal-g wal-push %p'
Normal backup

DB copy

Changes (WAL)
Delta backups

DB copy

Delta copy

Changes (WAL)
LSN-based deltas
Data flows in the system

P

Delta files

WAL (archiving)

S

WAL (streaming replication)

WAL (archive recovery)

Backup
PG features
Catchup

github.com/wal-g/wal-g/issues/363

Yandex Object Storage

AWS S3 API

P

LSN
Catchup

WAL archiving

Delta

LSN

Yandex Object Storage

github.com/wal-g/wal-g/issues/363
Catchup

WAL archiving

Delta

LSN

Yandex Object Storage
Backup-push quorum
Compress some bytes
LZ4 compression and decompression in pure Go

- go
- golang
- lz4
- lz4-frame

- 137 commits
- 4 branches
- 0 packages
- 34 releases
- 17 contributors
- BSD-3-Clause

Branch: master
New pull request

- cmd/lz4c: Writer: added concurrency support. Fixes #55
  4 months ago
- fuzz: Updated fuzz corpus 2.
  10 months ago
- internal/xxh32: cmd/lz4c: moved commands to local dir
  10 months ago
- testdata: DecodedBlock: handle case in shortcut2 where the destination buffer is...
  7 months ago
- .gitignore: updated gitignore
  11 months ago

Latest commit cac5ed4 on Feb 20
// using 50 bytes of stack currently

MOVO dst_base+0(FP), DI
MOVO DI, R11
MOVO dst_len+8(FP), R8
ADDQ DI, R8

MOVO src_base+24(FP), SI
MOVO src_len+32(FP), R9
ADDQ SI, R9

// shortcut ends
// short output end

MOVO R8, R12
SUBQ $32, R12
// short input end

MOVO R9, R13
SUBQ $16, R13

loop:
// for si < len(src)

CMPO SI, R9
JGE end

// token = uint32(src[si])
Fix short buffers in some specific cases

x4m committed on Mar 26, 2019

Showing 4 changed files with 4 additions and 0 deletions.

<table>
<thead>
<tr>
<th>Additions</th>
<th>Deletions</th>
<th>Path</th>
<th>Lines</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3</td>
<td>reader.go</td>
<td></td>
</tr>
</tbody>
</table>

```go
136 158, 158, 158, 158, 158, 158 158 158 158
158 158 159 159
160 160 161 161
161 162 162 162
163 163 163 163
164 164 164 164
165 165 165 165
166 166 166 166
```

if debugFlag {
    Debugf("reading block from writer")
}
// Reset uncompressed buffer
z.data = z.data[:cap(z.data)[:len(z.data)]]
// Block length: 0 = end of frame, highest bit set: uncompressed.
bLen, err := z.readUint32()
if err != nil {
```
Извините, но я не могу прочитать итоговую информацию с графиков.
Hi,

Our application stores and loads compressed data on disk using this library. We recently upgraded our application from `@zstd/v1.3.0` to `@zstd/v1.3.4`.

After this library upgrade, there were some complaints from users. After generating some JSON data, compressing it, storing it, then later loading it and decompressing it, the data could no longer be parsed (e.g. `json:unescape: invalid character 'u' in string literal`).

The issue occurred on (at least) both Windows and macOS, with both Go 1.9.7 and 1.10.3.

I assume it was caused by memory corruption in CGO data buffers.

Reverting this library back to tag `v1.3.0` seems to have completely resolved the issue going-forward.

We’re not yet able to reproduce the issue, and not all our users were affected (perhaps it’s related to OS memory pressure?), but, just a heads-up that this library upgrade was implicated. Once we have an internal reproducer we may be able to bisect it (or hopefully, blame something else and disregard this entire issue).
```go
+ // +build brotli
+ +
+ + package compression
+ +
+ + import "github.com/wal-g/wal-g/internal/compression/brotli"
+ +
+ + func init() {
+ +     Decompressors = append(Decompressors, brotli.Decompressor{})
+ +     Compressors[brotli.AlgorithmName] = brotli.Compressor{
+ +     CompressingAlgorithms = append(CompressingAlgorithms, brotli.AlgorithmName)
+ + }
```
OpenPGP

Use correct default hashes and default ciphers when no preferences given

Per comments in config.go

> // DefaultHash is the default hash function to be used.
> // If zero, SHA-256 is used.
> DefaultHash crypto.
> // DefaultCipher is the cipher to be used.
> // If zero, AES-128 is used.
> DefaultCipher CipherFunction

But instead we get RIPEMD160 and CAST5 if config is nil. Both are obsolete.
Fix https://github.com/golang/go/issues/37646

Change-Id: I55f22588e1c3c88798c8a9ffad8c57fd6859eb3
GitHub-Last-Rev: 07119c2fa44d3738bf2f2514556b502ff217d
GitHub-Pull-Request: golang/crypto#128

Files

<table>
<thead>
<tr>
<th>File</th>
<th>Comments</th>
<th>Size</th>
<th>Delta</th>
</tr>
</thead>
<tbody>
<tr>
<td>openpgo/write.go</td>
<td></td>
<td></td>
<td>+3</td>
</tr>
</tbody>
</table>

39
libsodium
Push-based vs Pull-based executor

GIT PUSH
AND RUN

ONE DOES NOT SIMPLY
GIT PULL
io.Reader vs io.Writer

type Reader interface {
    Read(p []byte) (n int, err error)
}

type Writer interface {
    Write(p []byte) (n int, err error)
}
func ReadAtLeast(r Reader, buf []byte, min int) (n int, err error) {
    if len(buf) < min : 0, ErrShortBuffer ↑
    for n < min && err == nil {
        var nn int
        nn, err = r.Read(buf[n:])
        n += nn
    }
    if n >= min {
        err = nil
    } else if n > 0 && err == EOF {
        err = ErrUnexpectedEOF
    }
    return
}
Store some bytes
S3-based WAL-G
S3-based WAL-G

AWS has its own cryptography
S3 ListObjectsV2()

ListObjects

PDF

Returns some or all (up to 1,000) of the objects in a bucket. You can use the request parameters as selection criteria to return a subset of the objects in a bucket. A 200 OK response can contain valid or invalid XML. Be sure to design your application to parse the contents of the response and handle it appropriately.

⚠️ Important

This API has been revised. We recommend that you use the newer version, ListObjectsV2, when developing applications. For backward compatibility, Amazon S3 continues to support ListObjects.

The following operations are related to ListObjects:
S3 infiltrated the whole codebase
Refactor storages

No description, website, or topics provided.

<table>
<thead>
<tr>
<th>Branch</th>
<th>New pull request</th>
</tr>
</thead>
<tbody>
<tr>
<td>master</td>
<td></td>
</tr>
</tbody>
</table>

This branch is 27 commits ahead of Tinsane:master.

<table>
<thead>
<tr>
<th>Commits</th>
<th>Branch</th>
<th>Pull request</th>
<th>Compare</th>
</tr>
</thead>
<tbody>
<tr>
<td>34</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Commits</th>
<th>Branch</th>
<th>Pull request</th>
<th>Compare</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Packages</th>
<th>Branch</th>
<th>Pull request</th>
<th>Compare</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Releases</th>
<th>Branch</th>
<th>Pull request</th>
<th>Compare</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Contributors</th>
<th>Branch</th>
<th>Pull request</th>
<th>Compare</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Apache-2.0</th>
<th>Branch</th>
<th>Pull request</th>
<th>Compare</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Commit</th>
<th>Branch</th>
<th>Pull request</th>
<th>Compare</th>
</tr>
</thead>
<tbody>
<tr>
<td>d4dedf5</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Latest commit d4dedf5 on Jan 23

- azure: use wal-g/tracelog instead of tinsane/tracelog
- fs: use wal-g/tracelog instead of tinsane/tracelog
- gcs: use wal-g/tracelog instead of tinsane/tracelog
- memory: Fix imports
- s3: Remove s3 debug (#14)
- storage: Add folder and object mocks
- swift: use wal-g/tracelog instead of tinsane/tracelog
GCP

- No multipart uploads
- A lot of problems with retries
Azure

Contribution from KubeDB
Problems with Windows build
No problems, does anyone use WAL-G + SWIFT?
SSH/SCP

Because we need to back up S3 too
WAL-G for other databases
WAL-G for MySQL®
What to archive?

Data

Transactions log
Differences from PostgreSQL

• Storage engines
• Redo log is circular
• No archive command
• PITR is based on binlog, not redo
Data archiving

- xtrabackup
- mysqldump
What is binlog?

- Transactions (rows or queries)
- Timestamps
- GTID - unique ID

```bash
$ mysqlbinlog -v /var/lib/mysql/binlog.001242
....
SET @@SESSION.GTID_NEXT='63bebac7-c424-11e9-bb82-15ed63279c98:31214803';
TIMESTAMP=1619172002;
BEGIN
BINLOG '
opqCYBMCAAAAOGAAALoxHgAAAG0AAAAAEEABW15c3
FsAAxtZGjfcmVwbF9tb24AAgMRAQMC+ajWYw==
opqCYB8CAAAAOGAAPQxHgAAAG0AAAAAEEAAAgAC///8
AQAAAGCCmqEaNvwBAAAAYIKaohpA3Bx6WA==
',
....
```
Binlog archivation

MySQL primary
- primary-bin-log-000091
- primary-bin-log-000092

MySQL secondary
- relay-log-000091
- secondary-bin-log-000035
- secondary-bin-log-000036

Replication

S3 cloud

Wal-G + Cron
Binlog backup and replay order

- MySQL primary
- MySQL secondary
- S3 Storage

- Time
- Replay order

- Binlog-push
- Same transactions
- Failover
- Happens before
Speed up replay

- Run MySQL on high port (3308)
- Disable all possible logging
- Don’t forget to put everything back into production!

<table>
<thead>
<tr>
<th>Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>double_write_buffer</td>
<td>OFF</td>
</tr>
<tr>
<td>innodb_flush_log_at_trx_commit</td>
<td>0</td>
</tr>
<tr>
<td>sync_binlog</td>
<td>1000</td>
</tr>
<tr>
<td>event-scheduler</td>
<td>OFF</td>
</tr>
</tbody>
</table>
Complete scenario

Back up

• Daily:
  wal-g backup-push

• Each minute:
  wal-g binlog-push

Restore

• wal-g backup-fetch LATEST

• Start mysql on high port

• wal-g binlog-replay
  --since LATEST
  --until 2021-05-12T12:00:00Z

• Restart MySQL with production config
Out of scope

• How to choose which node to back up?
• How to not upload binlog from a crashed master?
• How to handle binlog name collisions?
Does it work at all?

> 1k clusters

> 1.5k nodes

*In Yandex*
WAL-G for Microsoft SQL Server™
SQLServer backup features

• Database backups
• Log backups
• Incremental backups
• Point-in-time recovery
Backup methods

- DISK
- TAPE
- VIRTUAL DEVICE
- URL (Azure)
Backup to DISK

• Additional disk space required
• Additional IOPs required
• Still need to upload to storage

OR

• Additional network disk
WAL-G approach: emulate Azure
1. Need domain name
2. Need HTTPS
3. Only default HTTPS port - 443
4. Azure credentials required

$ Add-Content -Path
  -Value '127.0.0.42 backup.local'
  -Path 'C:\Windows\System32\Drivers\etc\hosts'

$ Import-Certificate
  -CertStoreLocation cert:\LocalMachine\Root \n  -FilePath 'C:\backup.local.cert.pem'

$ Invoke-Sqlcmd "
CREATE CREDENTIAL
[https://backup.local/basebackups_005]
WITH DENTITY='SHARED ACCESS SIGNATURE',
SECRET = 'does_not_matter'"
Case 1: flexibility

1. Run WAL-G in background
   ```
   wal-g --config c:\walg.yaml proxy
   ```

2. Now you have your own private Azure
   ```
   BACKUP DATABASE [db1]
   TO URL = 'https://walg.local/somepath/weekly/20210501'
   ```
Case 2: PG-like CLI

wal-g backup-push
wal-g backup-list
wal-g log-push
wal-g backup-restore LATEST
wal-g log-restore --since LATEST --until TS
Multiple databases

- backup-push
- backup-push -d db2
- log-restore
- log-restore -d db1,db2
Questions

Andrey Borodin

mail: x4mmm@yandex-team.ru
telegram: x4mmm

Dmitry Smal

mail: mialinx@yandex-team.ru
telegram: mialinx