WAL Hakkında Bilmek İstediğiniz (Hemen Hemen) Herşey

Devrim Gündüz

GITC Pro Limited, Gündüz Bilişim Danışmanlığı

Twitter : @DevrimGunduz (İngilizce, teknik)

Twitter: @DevrimGunduzTR (Türkçe, özel ve politik)

Hakkımda

- Bu arkadaş kim?
  - 1996’dan beri Red Hat (ve sonra Fedora) kullanıyorum.
  - 1998’den beri PostgreSQL kullanıyorumç
    - 20. yıl kutlaması :)
  - PostgreSQL’in resmi YUM (RHEL, CentOS, Fedora) ve Zypp (SLES) depolarının sorumlusuyum.
  - Fedora ve EPEL paketçisiyim.
  - 2011’den beri EnterpriseDB’de çalışıyorum. Türkiye’de Gündüz Bilişim Danışmanlığı, Londra’da GITC Pro Limited şirketlerinin sahibiyim.
  - Londra’da yaşıyorum.
  - PostgreSQL Dövmeli Adam!

PGDay.TR 2018
WAL hakkında bilmek istediğiniz herşey

Sosyal Medya

Lütfen tweet atın

#PGDayTR

Lütfen takip edin:

@PGDayTR

@PostgreSQL

GÜNDÜZ BİLİŞİM DANIŞMANLIĞI

Sosyal Medya

Tabii ki:

#BlameMagnus

@BlameMagnus

Sosyal Medya

(Tweetlediniz mi? Teşekkürler!)

GÜNDÜZ BİLİŞİM DANIŞMANLIĞI

Ajanda

- WAL nedir?
- İçinde ne var?
- Nasıl okunur?
- wal_level nedir?
- Replikasyon ve WAL
- Yedekleme ve WAL
- PITR ve WAL
- Full page write kavramı
- Diğer konular

GÜNDÜZ BİLİŞİM DANIŞMANLIĞI

Başlamadan önce:

WAL dosyalarını lütfen elle silmeyiniz.

Lütfen.

GÜNDÜZ BİLİŞİM DANIŞMANLIĞI

Başlamadan önce:

WAL dosyalarını lütfen elle silmeyiniz.

Lütfen.
Lütfen.

GÜNDÜZ BİLİŞİM DANIŞMANLIĞI

Başlamadan önce:

WAL dosyalarını lütfen elle silmeyiniz.

Lütfen.
Lütfen.

Lütfen.

GÜNDÜZ BİLİŞİM DANIŞMANLIĞI

WAL nedir?

- Veri kaybını önlemek için tasarlanmıştır.
  - İşletim sistemi, donanım ya da PostgreSQL'in çökmesi
  - Yazma transactionları WAL'a yazılır:
    - Transaction sonucu istemciye gönderilmeden önce
    - Veri dosyaları her transactionda değiştirilmez
    - Performans artışı
- Ayrı bir diskte tutulmasında yarar var
  - Initdb, ya da symlink

GÜNDÜZ BİLİŞİM DANIŞMANLIĞI

WAL nedir?

- Write Ahead Log:
  - Transactionların loglaması
  - Eskiden xlog olarak da bilinirdi (transaction log)
  - Neredeyse tüm kurulumlarda 16 MB (configure aşamasında –with-wal-segsize ile değiştirilebilir)
    - v11: initdb –wal-segsize parametresi içerir.
  - 8 kB page boyuyu (configure, --with-wal-blocksize)
  - pg_xlog (<= 9.6)→ pg_wal (10+)
    - İnsanlar “log” dizini altındakileri siliyorlardı!

GÜNDÜZ BİLİŞİM DANIŞMANLIĞI

WAL nedir?

- Gömülü bir özellik
- WAL'dan önce (Milattan önce kadar olmasa da...)
  - Tüm değişiklikler eninde sonunda diske gidiyor, ama:
    - Veriler shared buffers'a yüklenir.
    - Değişiklikler burada yapılırlar.
    - Dirty buffer!
  - Anlık olarak değil!
  - Çökme → Veri kaybı!

GÜNDÜZ BİLİŞİM DANIŞMANLIĞI

WAL nedir?

- WAL'dan sonra:
  - Tüm "değişiklikler" WAL dosyalarına "kaydedilirler" (WAL record)
    - Transaction iptal olsa bile (ROLLBACK/ABORT)
  - wal_buffers → WAL "segmentleri" (dosyaları)
  - Çökme sonrası veri kurtarabilme
  - Checkpoint!

GÜNDÜZ BİLİŞİM DANIŞMANLIĞI

Nerede kullanılır?

- Transaction logging!
- Replikasyon
- PITR
- REDO
  - Dosyalar sıralı olarak olmalıdırlar.
  - REDO ve UNDO
  - Temp tablolar ve unlogged tablolar için REDO yok.

GÜNDÜZ BİLİŞİM DANIŞMANLIĞI

Shared Buffers, Bgwriter and checkpointer

- PostgreSQL’de shared_buffers
  - Dirty buffers
  - Transactionlar burada gerçekleştirilirler
  - Yan etkisi: Storage’da veri bütünlüğü yoktur, ama bu sorun değildir.
- Bgwriter: Background writer
  - LRU
- Checkpointer
  - Tüm dirty bufferları storage’a yazar
  - Otomatik olarak ya da elle tetiklenir
- “Backendler” de aynı zamanda heap üzerine yazabilirler

GÜNDÜZ BİLİŞİM DANIŞMANLIĞI

WAL: LSN

- Log Sequence Number
  - Kaydın WAL dosyası içindeki yeri
  - Her WAL kaydının tekil olmasını sağlar
- 64-bit tamsayı (tarihsel nedenlerle 2x32-bit) (Bu bilgiye ileride gereksinmemiz olacak)
- Belgelerde: “Pointer to a location in WAL file”
- LSN: Block ID + Segment ID (Sonraki slaytlarda açıkl
- aması var)
- Kurtarma esnasında, page’deki LSN ve WAL’deki LSN karşılaştırılır.
  - Büyük olan kazanır.

GÜNDÜZ BİLİŞİM DANIŞMANLIĞI

WAL dosyası adlandırması

- 24 chars, hex.
  - 1st 8 chars: timelineID
    - 00000001 initdb tarafından yaratılan timelineID
  - 2nd 8 chars: Block ID
  - 3rd 8 chars: Segment ID
- 000000010000000000000001 → 000000010000000000000002
- … 00000001000000000000000FF → 000000010000000100000000
- ...and 00000001000000010000000FF → 000000010000000200000000

WAL dosyası adlandırması

- Öntanımlı WAL dosya boyutu: 16 MB
  - WAL dosyası içindeki bir kaydın yeri 24 bit ile tanımlanır (2^24 = 16 MB)
  - 64'ü alın, 32+32 olarak ikiye bölün. İkinci 32'den üstteki 24'ü çıkarın. 8 elde edeceksiniz. Bu 8 bit, WAL dosyası içinde saklanan low-order 32-bit tamsayıdır.
  - Hexadecimal olarak, her karakter 4 bit olarak tanımlanır, doyayısıyla 8 bit'i 4'e bölersek: 8 / 4= 2. Bu da önceki slayttaki FF'dir :)

GÜNDÜZ BİLİŞİM DANIŞMANLIĞI

WAL: Mevcut WAL dosyasını bulma

- ls -ltr listesindeki son dosya değildir muhtemelen!

- postgres=# SELECT * from pg_current_wal_lsn();

  pg_current_wal_location

  40E6/2C85AC10

- postgres=# SELECT pg_walfile_name('40E6/2C85AC10');
  pg_walfile_name

  000000030000040E60000002C

  Dolayısıyla:

- postgres=# SELECT pg_walfile_name(pg_current_wal_lsn());

  pg_walfile_name

  000000030000040E60000002C

Checkpoint, ve pg_control

- Checkpoint başlar başlamaz, REDO noktası shared buffers içinde saklanır.
- checkpoint'in başlama noktasını gösteren bir WAL kaydı da shared buffers içinde saklanır.
  - pg_control dosyası $PGDATA/global altındadır.
- bgwriter'ın aksine, checkpointer shared_buffers içindeki **tüm** veriyi diske yazar.
- PostgreSQL en son REDO noktasını, pg_control dosyasına bakarak anlar.

GÜNDÜZ BİLİŞİM DANIŞMANLIĞI

Checkpoint, ve pg_control

- **pg_controldata (v11 öncesi)**:

  Latest checkpoint location: 40E7/E43B16B8

  Prior checkpoint location: 40E7/D8689090

- **pg_controldata (v11+)**:

  Latest checkpoint location: 40E7/E43B16B8

  Bunlar LSN'dir.

- Checkpoint tamamlandığında, **pg_control** dosyası checkpoint noktası ile güncellenir.
- Checkpoint bitince, eski WAL dosyaları ya "recycle", ya da "remove" edilir.
- "recycle" için önceki checkpoint döngülerine bakarak bir "tahmin" yapılır.
- 9.5+: En az, **min_wal_size** kadar WAL dosyası her zaman "recycle" edilir.

GÜNDÜZ BİLİŞİM DANIŞMANLIĞI

pg_control ve REDO

- postmaster başlatılırken pg_control dosy

  /usr/pgsql-11/bin/pg_controldata -D /var/lib/pgsql/11/data | grep state

  - “Database cluster state”:
    - starting up
    - shut down
    - shut down in recovery
    - shutting down
    - in crash recovery
    - in archive recovery
    - in production
- Eğer pg_control dosyası “in production” diyorsa, ancak veritabanı sunucusu çalışmıyorsa, bu PostgreSQL sunucusu otomatik olarak önce “recovery” yapacaktır.

GÜNDÜZ BİLİŞİM DANIŞMANLIĞI

pg_control ve REDO

- pg_control kritik bir parçadır:
  - Zarar görmemesi gerekiyor
  - Belgelerde: “...theoretically a weak spot”
- REDO: Tam bir veri kurtarma için, tüm WAL dosyaları **sıralı** olarak bulunmalıdır
- UNDO: PostgreSQL 11’de yok, ancak:
    - https://github.com/EnterpriseDB/zheap/
    - https://wiki.postgresql.org/wiki/Zheap

GÜNDÜZ BİLİŞİM DANIŞMANLIĞI

Yeni WAL dosyasına geçme koşulları

- WAL segmenti dolmuş olabilir.
- PostgreSQL archiver süreci, archive_timeout değerine eriştiğinde yeni WAL'a geçiş yapacaktır.
- DBA **pg_switch_wal()** fonksiyonunu çağırmıştır.

GÜNDÜZ BİLİŞİM DANIŞMANLIĞI

WAL: Arşivleme

- Replikasyon, yedekleme, PITR
- archive_mode
- archive_command
- archive_timeout

GÜNDÜZ BİLİŞİM DANIŞMANLIĞI

WAL yönetimi

- PostgreSQL'in dahili araçlarını kullanın:
  - pg_archivecleanup
  - pg_resetwal
  - pg_waldump
  - ....

GÜNDÜZ BİLİŞİM DANIŞMANLIĞI

pg_waldump

- Hepimiz insanız :-)
- WAL dosyası içeriğini görmek için pg_waldump kullanın:
- rmgr --help to get list of all resource names, -f for follow, -n for limit. -z for stats.
- pg_waldump -n 20 -f 000000010000000700000033
- rmgr: Heap len (rec/tot): 3/ 59, tx: 389744, lsn: 7/33B66228, prev 7/33B661F0, desc: INSERT+INIT off 1, blkref #0: rel 1663/13326/190344 blk 0
- rmgr: Heap len (rec/tot): 3/ 59, tx: 389744, lsn: 7/33B66268, prev 7/33B66228, desc: INSERT off 2, blkref #0: rel 1663/13326/190344 blk 0
- rmgr: Transaction len (rec/tot): 8/ 34, tx: 389744, lsn: 7/33B662A8, prev 7/33B66268, desc: COMMIT 2017-02-03 03:03:49.482223 +03
- rmgr: Heap len (rec/tot): 14/ 69, tx: 389745, lsn: 7/33B662D0, prev 7/33B662A8, desc: HOT_UPDATE off 1 xmax 389745 ; new off 3 xmax 0, blkref #0: rel 1663/13326/190344 blk 0
- rmgr: Transaction len (rec/tot): 8/ 34, tx: 389745, lsn: 7/33B66318, prev 7/33B662D0, desc: COMMIT 2017-02-03 03:03:54.091645 +03
- rmgr: WAL len (rec/tot): 80/ 106, tx: 0, lsn: 7/33B66340, prev 7/33B66318, desc: CHECKPOINT_ONLINE redo 7/33B66340; tli 1; prev tli 1; fpw true; xid 0/389746; oid 198532; multi 1; offset 0; oldest xid 1866 in DB 129795; oldest multi 1 in DB 90123; oldest/newest commit timestamp xid: 388437/389745; oldest running xid 0; online
- rmgr: WAL len (rec/tot): 0/ 24, tx: 0, lsn: 7/33B663B0, prev 7/33B66340, desc: SWITCH

GÜNDÜZ BİLİŞİM DANIŞMANLIĞI

pg_waldump

- rmgr: XLOG len (rec/tot): 30/ 30, tx: 0, lsn: 0/0CE268C8, prev 0/0CE26890, desc: NEXTOID 26914

  rmgr: Storage len (rec/tot): 42/ 42, tx: 0, lsn: 0/0CE268E8, prev 0/0CE268C8, desc: CREATE base/14012/18722

  rmgr: Heap len (rec/tot): 54/ 1338, tx: 1829, lsn: 0/0CE26918, prev 0/0CE268E8, desc: INSERT off 7, blkref #0: rel 1663/14012/1247 blk 15 **FPW**

  rmgr: Btree len (rec/tot): 53/ 6393, tx: 1829, lsn: 0/0CE26E58, prev 0/0CE26918, desc: INSERT_LEAF off 315, blkref #0: rel 1663/14012/2703 blk 2 **FPW**

  ---

  rmgr: **Standby** len (rec/tot): 42/ 42, tx: 1833, lsn: 0/0CE57300, prev 0/0CE572C8, desc: LOCK xid 1833 db 14012 rel 18731

  rmgr: Heap len (rec/tot): 54/ 54, tx: 1833, lsn: 0/0CE57330, prev 0/0CE57300, desc: DELETE off 14 KEYS_UPDATED , blkref #0: rel 1663/14012/1247 blk 15

  rmgr: Heap len (rec/tot): 54/ 54, tx: 1833, lsn: 0/0CE57368, prev 0/0CE57330, desc: DELETE off 26 KEYS_UPDATED , blkref #0: rel 1663/14012/2608 blk 62

  rmgr: Standby len (rec/tot): 42/ 42, tx: 0, lsn: 0/0CE573A0, prev 0/0CE57368, desc: LOCK xid 1833 db 14012 rel 18731

GÜNDÜZ BİLİŞİM DANIŞMANLIĞI

pg_waldump

- pg_waldump -r list
  - src/include/access/rmgrlist.h
- pg_waldump -r sequence…

- Parametre değişiklikleri
- rmgr: XLOG len (rec/tot): 50/ 50, tx: 0, lsn: 2/9410C4A8, prev 2/9410C438, desc: PARAMETER_CHANGE max_connections=100 max_worker_processes=8 max_prepared_xacts=0 max_locks_per_xact=64 wal_level=replica wal_log_hints=off track_commit_timestamp=off

GÜNDÜZ BİLİŞİM DANIŞMANLIĞI

WAL: Point-In-Time Recovery (PITR)

- base backup (pg_basebackup!) ve WAL dosyaları gerekir
- WAL dosyaları, sıralı olarak tam olmalıdır, aksi taktirde PITR bitmez.
- “Roll-forward recovery”

GÜNDÜZ BİLİŞİM DANIŞMANLIĞI

WAL: Point-In-Time Recovery (PITR)

- PITR: **recovery target** ‘a kadar WAL dosyalarını base backupların üzerine oynatır.
  - **recovery_target_{time, xid, name, lsn}**
  - Belirtilmezse, arşivlenmiş tüm WAL dosyaları replay edilir.
- **recovery.conf** ve **backup_label:** Enters recovery mode.
  - restore_command, recovery_target_XXX,recovery_target_inclusive
- backup_label: checkpoint yerini de içerir kurtarmanın başlama noktası)
- Hemen hemen normal recovery süreci gibi (WAL replay)
- recovery_target_XXX ‘e kadar replay edilir.

GÜNDÜZ BİLİŞİM DANIŞMANLIĞI

WAL: Point-In-Time Recovery (PITR)

- Recovery sürecinden sonra, timelineID ve fiziksel WAL dosya adı da 1 arttırılır)
- .history file dosyası yaratılır.
- $ cat 00000003.history

  1 403F/58000098 no recovery target specified

  2 4048/43000098 before 2018-08-28 11:13:21.124512+03

  “WAL dosyaları bu ana kadar replay edildi, ve replay location da 4048/43000098.

GÜNDÜZ BİLİŞİM DANIŞMANLIĞI

Full page writes (FPW)

- Bir WAL kaydı, bgwriter ve / ya da checkpointer sırasında zarar görmüş bir page üzerinde replay edilemez. Bu zararın nedeni donanım, işletim sistemi, kernel, vs çökmesi olabilir.
- “Full page writes”.
- Öntanımlı olarak açık.
  - Eğer PostgreSQL destek şirketlerine para aktarmak istiyorsanız, bu parametreyi kapatın ;)

GÜNDÜZ BİLİŞİM DANIŞMANLIĞI

Full page writes

- PostgreSQL , bir page **her** checkpoint sonrasındada sayfa değiştiğinde, header verisi + tüm veri page'ini WAL kaydı olarak yazar.
  - checkpoint_timeout süresini arttırmak işe yarar.
  - Full-page image, backup block.
- PostgreSQL bu sayede yazma hatalarından bile kendisini kurtarabilir.

GÜNDÜZ BİLİŞİM DANIŞMANLIĞI

WAL parametreleri

- wal_level: Minimal, replica ya da logical
  - Archiver processinin çalışması için > minimal olmalı.
  - Öntanımlı: replica
- fsync : Her zaman on, lütfen.
- synchronous_commit: Bazı transactionlar kaybolabilir:
  - Sunucu diske veriyi flush etmeden önce **bir miktar** bekler.
  - fsync'i kapatmaktan daha az riskli.
- wal_sync_method : fdatasync genelde daha iyidir. Denemek için pg_test_fsync'i kullanabilirsiniz.

GÜNDÜZ BİLİŞİM DANIŞMANLIĞI

WAL parametreleri

- wal_log_hints: When this value is set to on , the server writes the entire content of each disk page to WAL after a checkpoint and during the first modification of that page, even for non-critical modifications of so-called hint bits.
- wal_compression: off by default. Less WAL files, more CPU overhead.
- wal_buffers: -1: Automatic tuning of wal buffers: 1/32 of shared_buffers (not less than 64kB or no more than 16 MB (1 WAL file)
- wal_writer_delay : Rounds between WAL writer flushes WAL.
- wal_writer_flush_after: New in 9.6

GÜNDÜZ BİLİŞİM DANIŞMANLIĞI

Sorular, yorumlar?

GÜNDÜZ BİLİŞİM DANIŞMANLIĞI

WAL Hakkında Bilmek İstediğiniz (Hemen Hemen) Herşey

Devrim Gündüz

GITC Pro Limited, Gündüz Bilişim Danışmanlığı

Twitter : @DevrimGunduz (İngilizce, teknik)

Twitter: @DevrimGunduzTR (Türkçe, özel ve politik)