

PGroonga運用技法

~PGroongaのWALを
放置していませんか？~

堀本泰弘

クリアコード

PostgreSQL Conference Japan 2020
2020-11-13





自己紹介



堀本 泰弘

全文検索エンジン

gr^{cloud}nga pgr^{cloud}nga
mr^{cloud}nga rr^{cloud}nga

の開発・サポート



今日のテーマ

PGroongaの
WALを
うまく使おう



PGroongaの紹介

の前に...



PGroongaの紹介

(知らない人向け)
PGroonga紹介



PGroongaの紹介

PGroongaとは？



PGroongaの紹介

PostgreSQL経由 で 高速な全文検索



PGroongaの特徴

1. **簡単**に使える
2. **速い**
3. **全言語**対応



PGroongaの使い方

簡単に使える



PGroongaの使い方



専用のクエリー
を覚えなくてOK!





PGroongaの使い方

実際に全文検索
してみましよう



実行例：テーブル定義

```
CREATE TABLE entries (  
  title text,  
  content text  
);
```



実行例： インデックス定義

```
-- 全文検索用インデックス  
CREATE INDEX entries_full_text_search  
ON entries  
-- 「USING pgroonga」=「PGroongaを使う」  
USING pgroonga (title, content);
```



実行例：データ挿入

-- 普通に挿入するだけでよい

```
INSERT INTO entries
```

```
VALUES ('PGroongaで高速全文検索！',  
       '高速に全文検索したいですね！');
```



実行例：全文検索

```
SELECT title FROM entries  
WHERE
```

```
-- &@~で全文検索
```

```
-- 「検索」と「高速」をAND検索
```

```
title &@~ '検索 高速' OR  
content &@~ '検索 高速';
```



実行例 : LIKE

```
SELECT title FROM entries  
WHERE
```

- *LIKE*でもインデックスが効く
- = アプリを書き換えずに高速化可能
- ただし~より性能が落ちる

```
title LIKE '%検索%' OR  
content LIKE '%検索%';
```



PGroongaの使い方

簡単ですね！



PGroongaの速度

速い



PGroongaの速度

追加されたデータ
は**即時**検索可能



PGroongaの速度

つまり



PGroongaの速度

検索可能になるの
が**速い**



PGroongaの速度

更新中も検索性能
が落ちない



PGroongaの速度

ヒット数が多くても
検索性能が
落ちない



PGroongaの速度

PGroongaは
安定して速い



PGroongaの速度

もちろん
検索も**速い**



PGroongaの速度

- ベンチマーク
 - <https://pgroonga.github.io/ja/reference/pgroonga-versus-pg-bigm.html>



PGroongaの特徴

全言語対応



PGroongaの特徴

さらに



PGroongaの特徴

継続してメンテナ
ンスされている



PGroongaの特徴

PostgreSQL13
にも対応！



PGroongaの紹介

以上、PGroonga
の紹介でした



今日のテーマ

本題に入ります



今日のテーマ

PGroongaの
WALを
うまく使おう

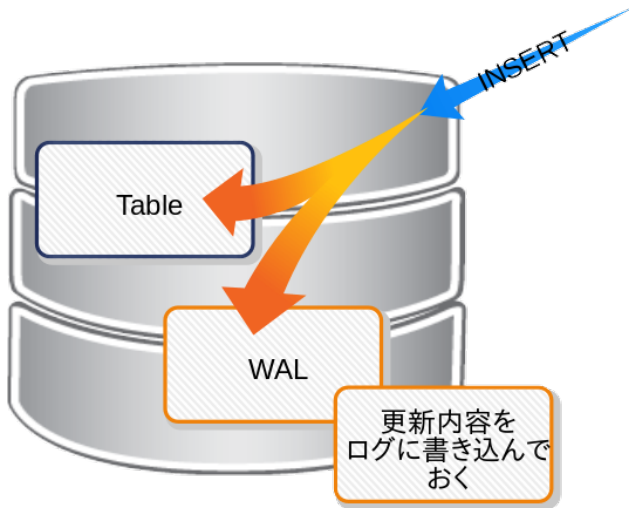


WALとは？

Write Ahead Log



WALとは？





WALとは？



aa	bb	cc
A	B	C

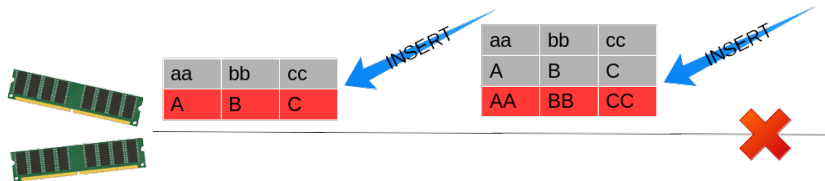
同期書き込み

aa	bb	cc
A	B	C





WALとは？





WALの使い方

どのように
使うのか？

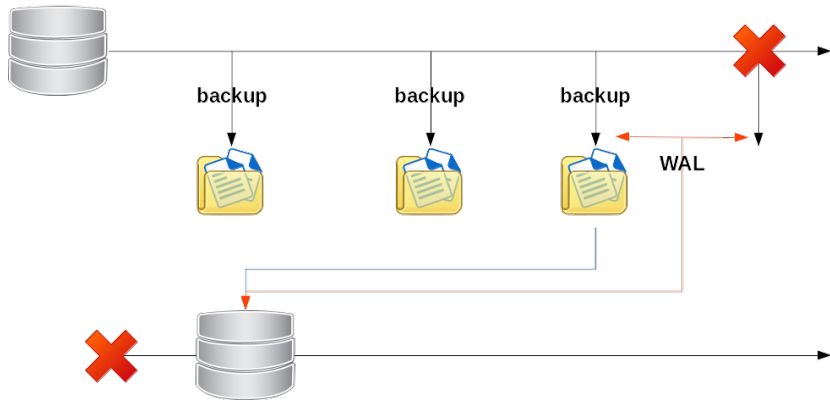


WALの使い方

ポイントイン
タイムリカバリー



WALの使い方



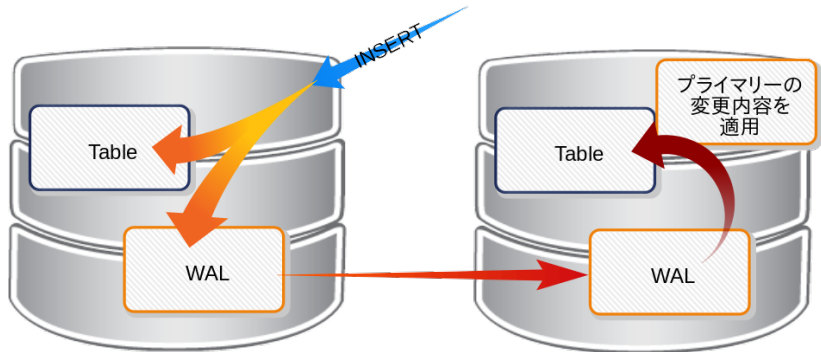


WALの使い方

ストリーミング レプリケーション



WALの使い方





PGroongaのWAL

PGroongaも
WALが使える



PGroongaのWAL

WALから
インデックスの
復元ができる



PGroongaのWAL

メリット



インデックスの 復旧時間を短縮



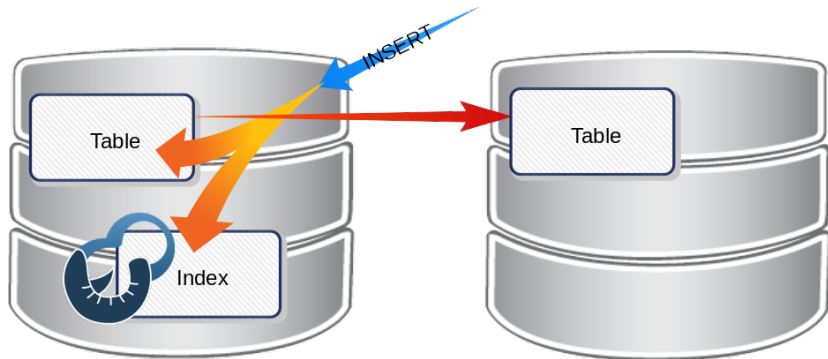
PGroongaのWAL



PGroongaのWALが無いと
どうなるの？



PGroongaのWAL





PGroongaのWAL

REINDEX



PGroongaのWAL

REINDEXは
時間がかかる



PGroongaのWAL

どうやって
使うの？



PGroongaのWAL

簡単です
ただ使うだけ
なら



PGroongaのWAL

```
pgroonga.enable_wal = on
```



PGroongaのWAL

- PostgreSQLのWALと同じように復旧に使える
 - ポイントインタイムリカバリー
 - ストリーミングレプリケーション



PGroongaのWAL

しかし



PGroongaのWAL

ただ使っている
だけだと、
困ることがある



PGroongaのWALの注意点

1. WALが適用できないケースがある
2. 定期的なメンテナンスが必要



PGroongaのWALの注意点

WALが適用
できないケース



PGroongaのWALの注意点

- クラッシュセーフではない
 - インデックス更新中のクラッシュでインデックスが破損する(ことがある)



PGroongaのWALの注意点

- インデックス破損の状態では、WALを適用できない(ことがある)
 - この場合はREINDEXを使って復元



PGroongaのWALの注意点

定期的な メンテナンス



PGroongaのWALの注意点

定期的な
VACUUM実行



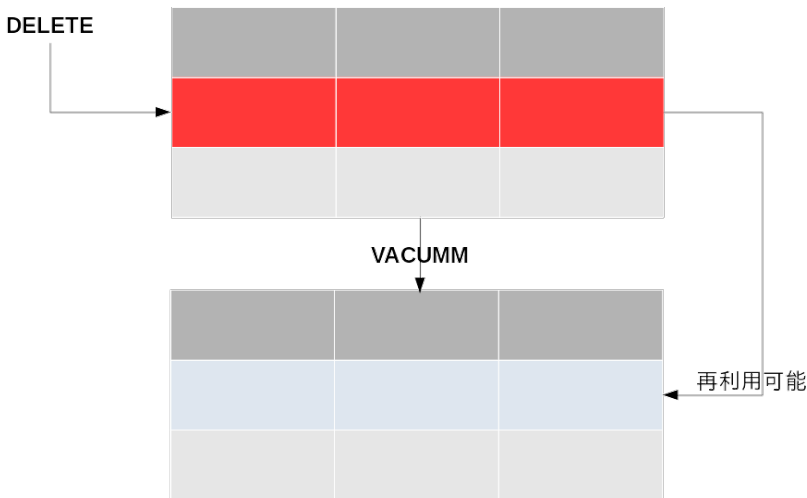
インデックスファイルの
サイズが増え続けている。。。。



定期的にVACUUMしているから、ファイルサイズは
一定の範囲に収まるんじゃないの？



PGroongaのWALの注意点





PGroongaのWALの注意点

なぜ？



ファイルサイズ増加の原因

原因はPGroonga
のWAL



ファイルサイズ増加の原因

PGroongaの
WALは削除しな
いと
増え続ける



ファイルサイズ増加の原因

つまり



ファイルサイズ増加の原因

1. PGroongaのWALが有効
2. 何もせずずっと運用している



ファイルサイズ増加の原因

ストレージが食いつぶされていく...



ファイルサイズの増加がとまらない





PGroongaのWALの運用方法

なので



定期的な メンテナンス が必要



ファイルサイズの
増加を防ぐには？



PGroongaのWALの運用方法

1. REINDEXによるインデックス再構築
2. WALを定期的に削除



REINDEX

- REINDEXすると、インデックスを0から再生成
- 古いインデックスは、VACUUM時に削除
 - このときにWALも削除される



REINDEX

- 😞 時間がかかる
- 😞 REINDEX中は検索、更新がブロックされる
- 😊 手間は少ない



WALの定期的な削除

- PGroongaのWALはSQLで削除できる



WALの定期的な削除

```
SELECT pgroonga_wal_truncate('インデックス名');
```



WALの定期的な削除

- 🙄 手間がかかる
 - テーブルのロックが必要
 - データの更新ができない
 - 明示的なWALの適用が必要
- 😄 REINDEXより短時間



WALの定期的な削除

- 明示的なWALの適用が必要
 - 単純にWAL削除すると、未適用のWALまで削除してしまう可能性がある



PGroongaのWALの運用方法

なので



明示的にWALを 適用する



PGroongaのWALの運用方法

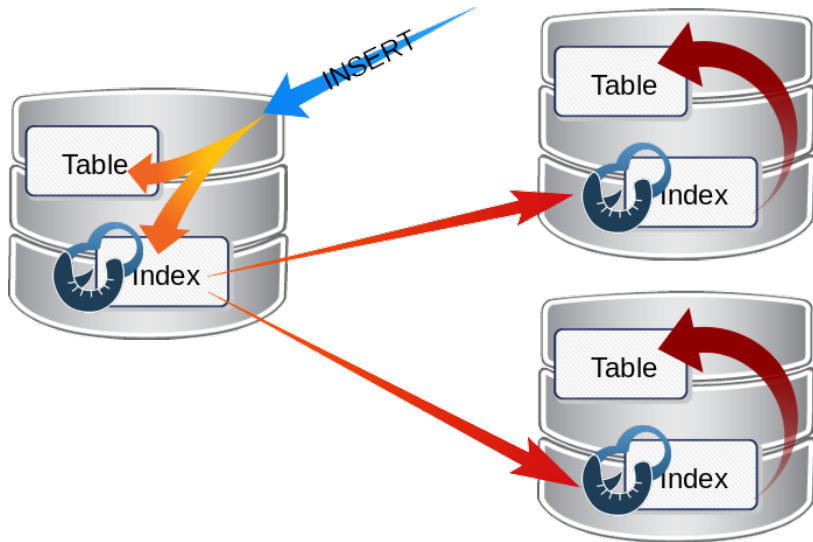
```
SELECT pgroonga_wal_apply('インデックス名');
```



ストリーミング レプリケーション 構成の場合



PGroongaのWALの運用方法





全ての セカンダリーDB でWALを適用



PGroongaのWALの運用方法

1. プライマリーDBでテーブルをロック
2. 全てのセカンダリーDBでWALを適用
3. プライマリーDBでWALを削除
4. テーブルのロックを解除



メンテナンス方法の選択

1. REINDEXによるインデックス再構築
 2. WALを定期的に削除
- のどちらが良いのか？



メンテナンス方法の選択

■ REINDEX

- 😞 時間がかかる
- 😞 REINDEX中は検索、更新がブロックされる
 - ブロックせずにREINDEXする方法もある
- 😊 手間は少ない



メンテナンス方法の選択

- REINDEX CONCURRENTLY
 - PostgreSQL 12 以降で使える



メンテナンス方法の選択

- 🙄 インデックスの構築にかなり時間がかかる
- 🙄 インデックス構築中はパフォーマンスが低下する



メンテナンス方法の選択

- WALの定期的な削除
 - 🙄 手間がかかる
 - テーブルのロックが必要
データの更新ができない
 - 明示的なWALの適用が必要
 - 😊 REINDEXより短時間



メンテナンス方法の選択

- REINDEX向きのケース
 - 長時間のサービス停止が許容できる
 - 例えば夜間に実行するなど
 - テーブルが小さい



メンテナンス方法の選択

- WALの削除向きのケース
 - 長時間のサービス停止が難しい
 - テーブルが大きい



PostgreSQLのWAL

- PostgreSQLのWALはこのような作業は必要ない。



PostgreSQLのWAL

- PostgreSQLはWALの最大サイズを制御できる
 - `max_wal_size`
 - `wal_keep_segments`



PostgreSQLのWAL

- PGroongaには、PostgreSQLと同程度の機能はまだない。



PostgreSQLのWAL

- 今後、改善予定
 - PostgreSQLと同じような仕組みを追加



まとめ

PGroongaの
WALを
うまく使うことが
できました！！



最後に

- PGroongaについての疑問等は、GitHub、Gitterにて
- ドキュメントも充実
 - <https://pgroonga.github.io/ja/>



最後に

より突っ込んだお話がしたい場合は

↓ ↓

問い合わせ先：

<https://www.clear-code.com/contact/>