Mroonga
と
PGroonga
Groongaを使って MySQLとPostgreSQLで日本語全文検索

須藤功平 クリアコード

MySQLとPostgreSQLと日本語全文検索
2016-02-09

Mroonga・PGroonga

- Mroonga（むるんが）
  - MySQLに
    高速日本語全文検索機能を追加する
    プロダクト
- PGroonga（ぴーじーるんが）
  - PostgreSQLに
    高速日本語全文検索機能を追加する
    プロダクト

すごい！使いたい！

- インストールして！
  - え。。。組み込みじゃないの。。。
    （MariaDBにはMroongaは組み込まれています！）
- パッケージあるから簡単だよ！
  - クラウドサービスで使えない。。。
- （クラウドサービスに入っていれば…！）
  - HerokuのPostgreSQLにPGroonga入れて！とお願いだ！

使いたい！？

HerokuのPostgreSQLで
PGroongaを使えるなら
Herokuを使いたい！
#herokujp

↑と思うならtweet！
(Herokuの人が観測します。)

高速？

ベンチマーク！

- 対象：Wikipedia日本語版
- レコード数：約185万件
- データサイズ：約7GB
- メモリー4GB・SSD250GB (ConoHa)

https://github.com/groonga/wikipedia-search/issues/4

（他人のベンチマークは参考程度）
（検討時はちゃんと実際の環境でベンチマークをとろう！）

速さ：検索1

キーワード：テレビアニメ

（ヒット数：約2万3千件）

InnoDB ngram	3m2s
InnoDB MeCab	6m20s
Mroonga:1	0.11s
pg_bigm	4s
PGroonga:2	0.29s

速さ：検索2

キーワード：データベース

（ヒット数：約1万7千件）

InnoDB ngram	36s
InnoDB MeCab:1	0.03s
Mroonga:2	0.09s
pg_bigm	2s
PGroonga:3	0.17s

速さ：検索3

キーワード：PostgreSQL OR MySQL

（ヒット数：約400件）

InnoDB ngram	N/A（エラー）
InnoDB MeCab:1	0.005s
Mroonga:2	0.028s
pg_bigm	0.185s
PGroonga:3	0.063s

速さ：検索4

キーワード：日本

（ヒット数：約63万件）

InnoDB ngram	1.3s
InnoDB MeCab	1.3s
Mroonga:1	0.21s
pg_bigm:2	0.84s
PGroonga	1s

速さ：検索まとめ

- Mroonga・PGroonga
  - 安定して速い
- InnoDB FTS MeCab・pg_bigm
  - ハマれば速い
- InnoDB FTS ngram
  - 安定して遅い

使いたい！？

HerokuのPostgreSQLで
PGroongaを使えるなら
Herokuを使いたい！
#herokujp

↑と思うならtweet！
(Herokuの人が観測します。)

速さ：データロード

約185万件・約7GB・SSD

InnoDB ngram	6m51s
InnoDB MeCab	6m22s
Mroonga:3	5m45s
pg_bigm:1	5m14s
PGroonga:2	5m22s

MySQLはbinlog有効、PostgreSQLはWAL有効
InnoDBはどっちも同じ処理
pg_bigmとPGroongaもどっちも同じ処理

速さ：インデックス作成

約185万件・約7GB・SSD

InnoDB ngram	3h06m58s
InnoDB MeCab	2h41m55s
Mroonga:1	22m24s
pg_bigm	3h43m23s
PGroonga:2	54m34s

MySQLはbinlog有効、PostgreSQLはWAL有効
バルクインデックス作成
＝データ投入後インデックス作成

速さ：ロードまとめ

- データロードは大差ない
- インデックス作成は大差
  - Mroonga・PGroongaは分単位
  - InnoDB・pg_bigmは時間単位

使いたい！？

HerokuのPostgreSQLで
PGroongaを使えるなら
Herokuを使いたい！
#herokujp

↑と思うならtweet！
(Herokuの人が観測します。)

サイズ：データ

InnoDB ngram	10GB
InnoDB MeCab	10GB
Mroonga	8.2GB
pg_bigm:2	5.1GB
PGroonga:1	4.3GB

InnoDBはどっちも同じ
pg_bigmとPGroongaはどっちも同じはずだけど…

サイズ：インデックス

InnoDB ngram	12GB
InnoDB MeCab:1	6GB
Mroonga:1	6GB
pg_bigm:3	7GB
PGroonga	10GB

InnoDBは一時ファイル（何十GB単位）を作る
PGroongaは元データ（8GB）のコピーもLZ4圧縮して持っている

サイズ：まとめ

- データサイズ
  - PostgreSQLは元データより小さめ
  - InnoDBは元データより大きめ
- インデックスサイズ
  - InnoDB MeCabは小さめ
    （ヒント：形態素解析ベースの方が小さくなる）
  - Mroonga・pg_bigmはN-gramなのに InnoDB MeCabと同じくらい

高速？

ベンチマークで
確認

Mroonga・PGroonga

- Mroonga（むるんが）
  - MySQLに
    高速日本語全文検索機能を追加する
    プロダクト
- PGroonga（ぴーじーるんが）
  - PostgreSQLに
    高速日本語全文検索機能を追加する
    プロダクト

実現方法

- Mroonga（むるんが）
  - MySQLに
    Groonga（ぐるんが）を組み込み
- PGroonga（ぴーじーるんが）
  - PostgreSQLに
    Groonga（ぐるんが）を組み込み

Groonga

- 国産の高速全文検索エンジン
  - 日本語バッチリ
- ライブラリーとして使える
  - 組み込みやすい
- マルチスレッド対応
  (MySQL組み込み時にうれしい)
- マルチプロセス対応
  (PostgreSQL組み込み時にうれしい)

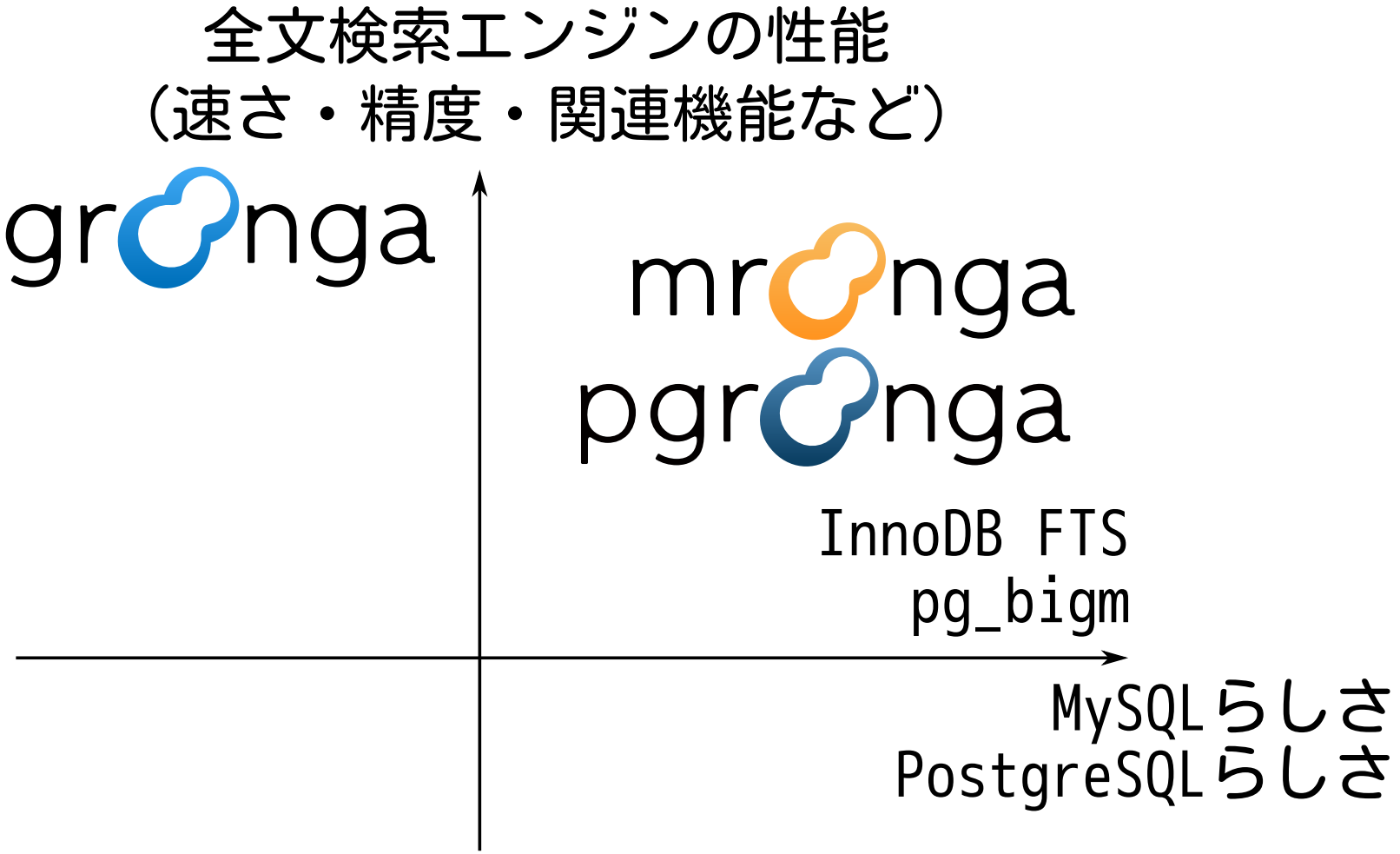
組み込み方針

- Groongaをできるだけ活かす
- 使い勝手はMySQL・PostgreSQLに寄せる

↓

SQLで使えるGroonga

ポジション


全文検索エンジンの性能
（速さ・精度・関連機能など）
groonga
mroonga
pgroonga
InnoDB FTS
pg_bigm
MySQLらしさ
PostgreSQLらしさ

SQLで使えるGroonga

- Groongaのフル機能は諦める
  - 速度など譲れない部分はがんばる
- その分、使いやすさを重視
  - 使いやすさ1＝
    MySQL・PostgreSQLとなじんでいる
  - 使いやすさ2＝
    MySQL・PostgreSQLの不便を解消

なじみ度：Mroonga

インデックス作成：MySQLと同じ

```
CREATE TABLE ... (
  ...,
  FULLTEXT INDEX (column)
) ENGINE=Mroonga;
```

なじみ度：Mroonga

全文検索：MySQLと同じ

```
SELECT * FROM ...
  WHERE
    MATCH(column)
    AGAINST('キーワード'
            IN BOOLEAN MODE);
```

不便解消：Mroonga

デフォルトOR→AND

```
-- ↓AまたはBが含まれていればマッチ
AGAINST('A B' IN BOOLEAN MODE);
AGAINST('+A +B' IN BOOLEAN MODE);
-- ↑↓AとBが含まれていればマッチ
-- ↓Mroongaの拡張
AGAINST('*D+ A B' IN BOOLEAN MODE);
```

不便解消：Mroonga

重み指定

```
-- titleかcontentにAがあればマッチ
--  （便利。PostgreSQLではできない。）
MATCH(title, content)
AGAINST('A' IN BOOLEAN MODE)
-- でもtitleの方を重要視したい！
-- ↓Mroongaの拡張
AGAINST('*W1:10,2:1 A' IN BOOLEAN MODE)
```

不便解消：Mroonga

全文検索+ORDER LIMIT高速化

```
SELECT * FROM tweets
  WHERE
    MATCH(content)
    AGAINST('...' IN BOOLEAN MODE)
  ORDER BY timestamp DESC
  LIMIT 10;
```

ORDER LIMIT高速化

- なぜ速いか
  - Groongaでソートし、LIMIT件だけMySQLに返しているから
  - MySQLよりGroongaでやった方が速い
    （ヒント：カラムストア）

さらにORDER LIMIT高速化

```
SELECT * FROM tweets
  WHERE
    MATCH(content)
    AGAINST('...' IN BOOLEAN MODE) AND
    timestamp >= '2016-02-01'
    -- ↑今月の分だけ対象にしたい
  ORDER BY timestamp DESC
  LIMIT 10;
```

さらにORDER LIMIT高速化

- なぜ速いか
  - Groongaで**絞り込んで**ソートし、LIMIT件だけMySQLに返しているから
  - MySQLよりGroongaでやった方が速い
    （ヒント：カラムストア）
- 場合によっては10倍以上高速化
  - http://tech.gmo-media.jp/post/69542751128/mroonga-311-new-optimization

PGroonga

- Groongaのフル機能は諦める
  - 速度など譲れない部分はがんばる
- その分、使いやすさを重視
  - 使いやすさ1＝
    MySQL・**PostgreSQL**となじんでいる
  - 使いやすさ2＝
    MySQL・**PostgreSQL**の不便を解消

なじみ度：PGroonga

インデックス作成：
PostgreSQLと同じ

```
CREATE INDEX name ON texts
  USING pgroonga (content);
```

なじみ度：PGroonga

全文検索：
PostgreSQLのtextsearchとほぼ同じ

```
SELECT * FROM ...
  WHERE
    column @@ 'キーワード';
```

textsearchとの違い

構文

```
-- textsearch
-- プログラムのよう
'(A & B) | C'
-- PGroonga（不便解消）
-- Web検索エンジンのよう
'(A B) OR C'
```

不便解消：PGroonga

Windows用バイナリーあり

- 商用ログ管理製品
  「VVAULT AUDIT」が採用
  http://vvault.jp/product/vvault-audit/
  - アクセスログに対して
    ユーザー名・パスを全文検索
- 決め手：高速・省スペース

不便解消：PGroonga

JSONデータを全文検索

```
CREATE TABLE logs (record jsonb);
CREATE INDEX i ON logs
  USING pgroonga (record);
-- ログのどこかに「error」があればマッチ
SELECT * FROM logs
  WHERE record @@ 'string @ "error"';
```

JSON全文検索例

以下は全部マッチ

```
{"message": "Error!"}
{"tags": ["web", "error"]}
{"syslog": {"message": "error!"}}
```

使いたい！？

HerokuのPostgreSQLで
PGroongaを使えるなら
Herokuを使いたい！
#herokujp

↑と思うならtweet！
(Herokuの人が観測します。)

まとめ1

- Groonga（ぐるんが）
  - 国産の高速全文検索エンジン
- Mroonga（むるんが）
  - MySQLからGroongaを使える！
- PGroonga（ぴーじーるんが）
  - PostgreSQLからGroongaを使える！

まとめ2

実装方針

- Groongaをできるだけ活かす
  （例：速度）
- MySQL/PostgreSQLっぽく使える
- MySQL/PostgreSQLをより便利に

次回予告

- トランザクションは？
- クラッシュしたら？
- レプリケーションは？
- もっと速くならないの？