# PGroonga & Zulip

Kouhei Sutou

*ClearCode Inc.*

*Zulip & PGroonga Night*
*2017-09-06*

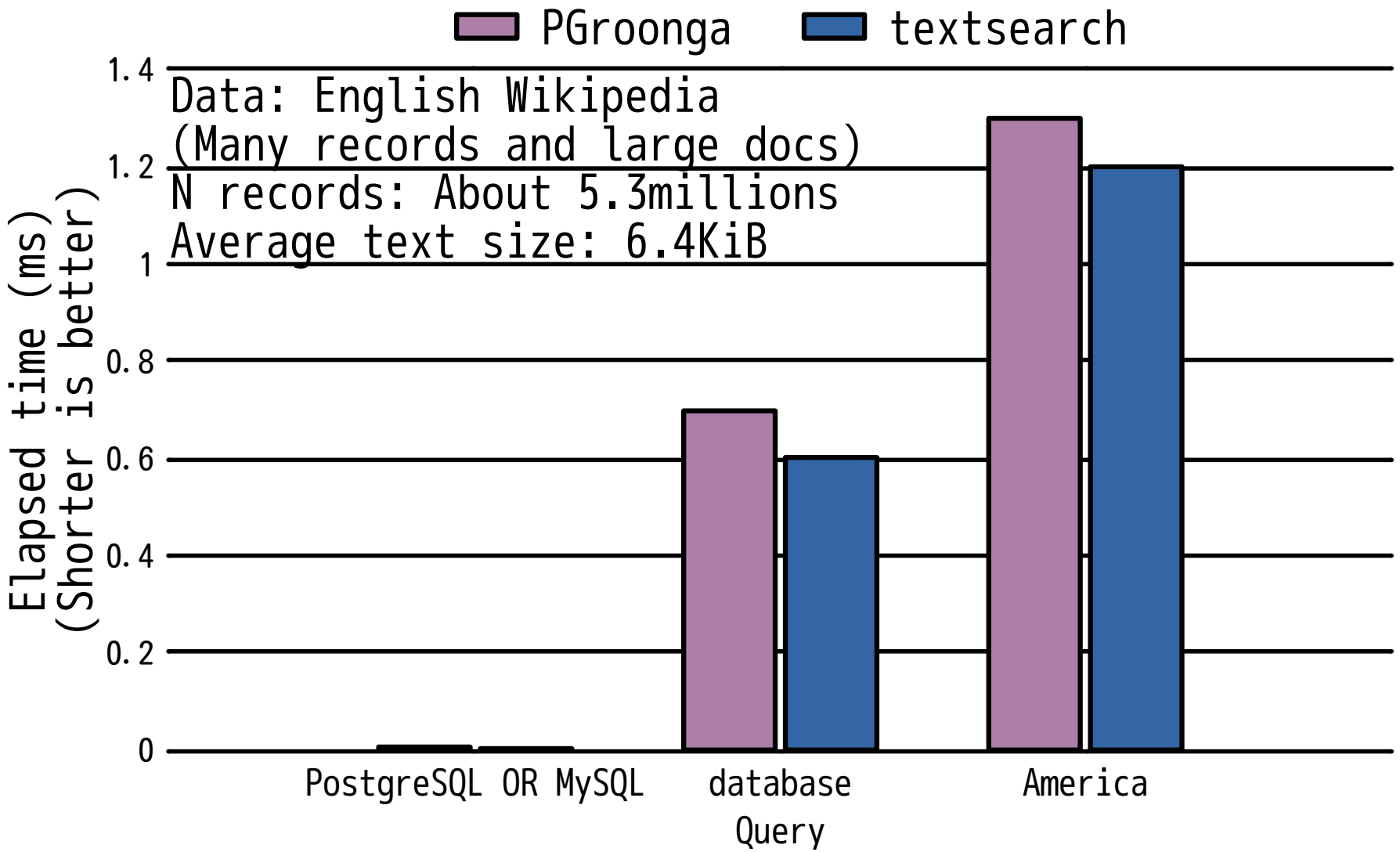
# PGroonga

- Pronunciation: pí:zí:lúngά
  読み方：ぴーじーるんが
- PostgreSQL extension
  PostgreSQLの拡張機能
  - Fast full text search
    高速全文検索機能
  - All languages are supported!
    全言語対応！

# Fast? (高速？)

- Need to measure to confirm
  確認するには測定しないと
- Targets (測定対象)
  - textsearch (built-in) (組み込み)
  - pg_bigm (third party) (外部プロダクト)

# PGroona and textsearch



# As fast as textsearch

## textsearchと同じくらいの速さ

- textsearch uses word based full text search
  textsearchは単語ベースの全文検索実装
- PostgreSQL has enough performance for the approach
  PostgreSQLはこの方法では十分な性能を出せる

# textsearch and Japanese
## textsearchと日本語

- Asian languages including Japanese aren't supported
  日本語を含むアジア圏の言語は非サポート
  - Need plugin（プラグインが必要）
  - Plugin exists but isn't maintained
    プラグインはあるがメンテナンスされていない

# Japanese support
## 日本語対応

- Need one of them（どちらかが必要）
  - N-gram approach support
    N-gramというやり方のサポート
  - Japanese specific word based approach support
    日本語を考慮した単語ベースのやり方のサポート

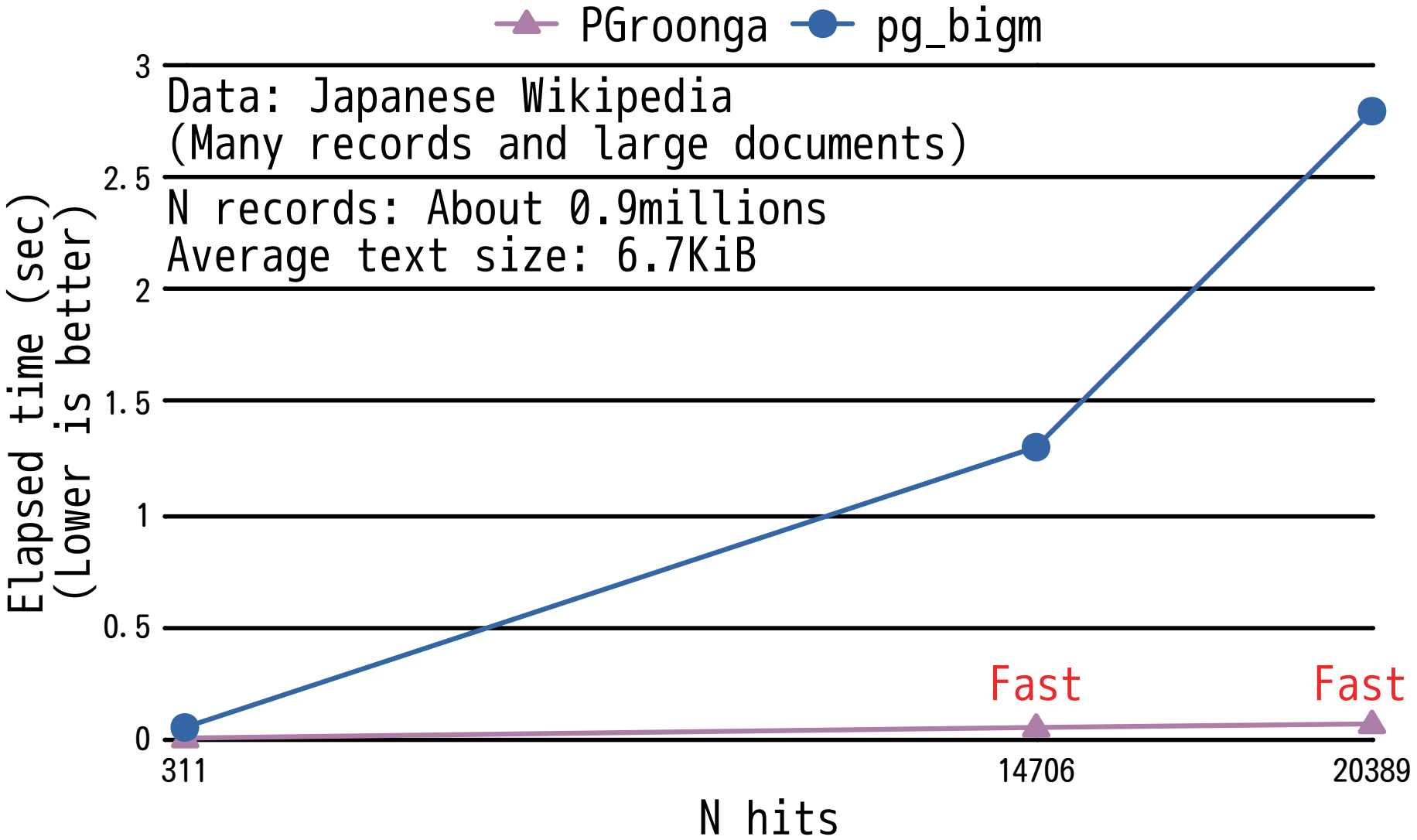
PGroonga supports both of them

# PostgreSQL and N-gram

## PostgreSQLとN-gram

- PostgreSQL is slow with N-gram approach
  PostgreSQLでN-gramというやり方を使うと遅い
- N-gram approach:
  - pg_trgm (contrib)
    - Japanese isn't supported by default
      デフォルトでは日本語に対応していない
  - pg_bigm (third-party)

# PGroona and pg_bigm



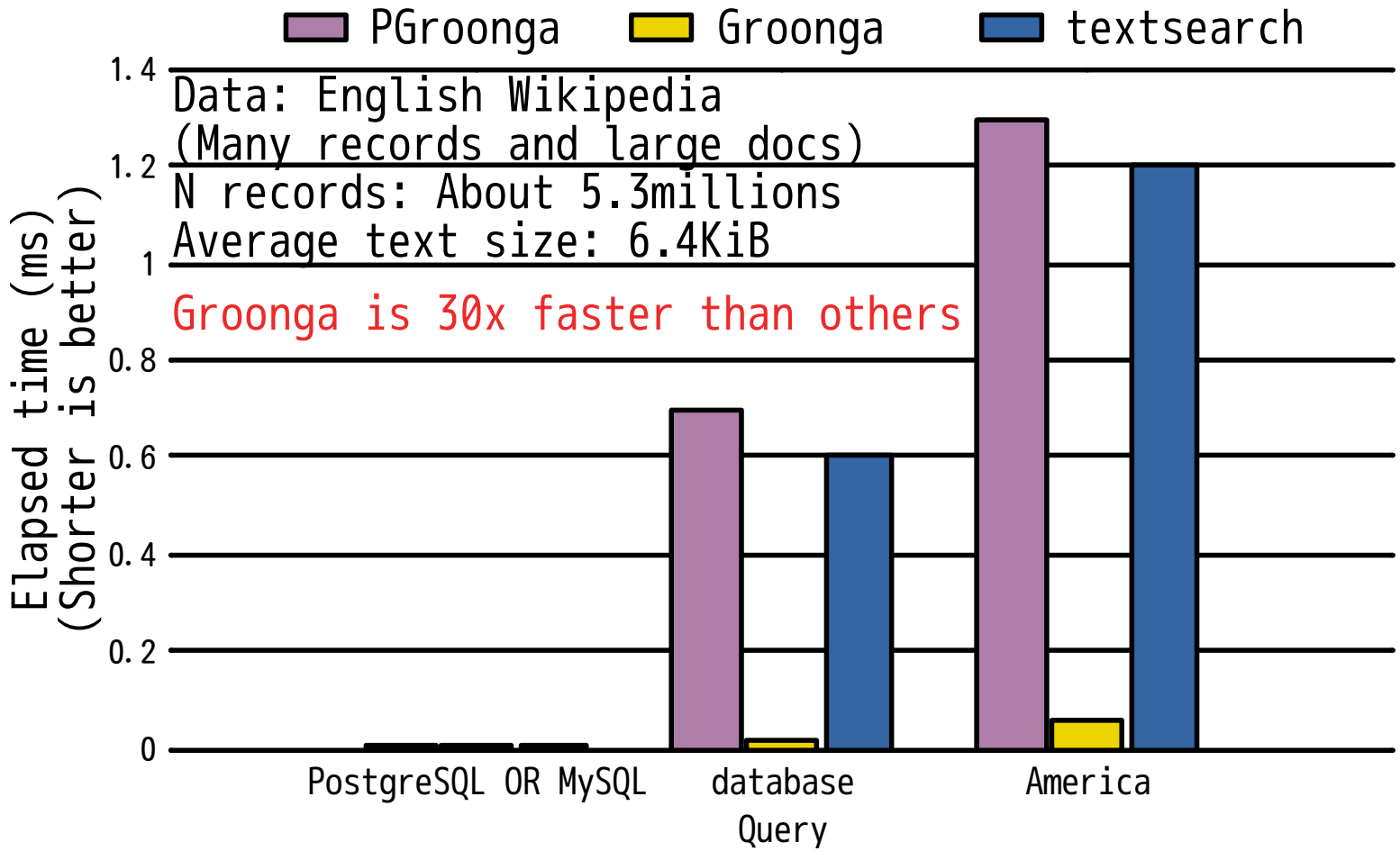
# PGroonga is fast stably
## PGroongaは安定して速い

- PostgreSQL needs "recheck" for N-gram approach
  PostgreSQLはN-gramのときは「recheck」が必要
  - Seq search after index search
    インデックスサーチのあとにシーケンシャルサーチ
- PGroonga doesn't need
  PGroongaでは必要ない
  - Only index search
    インデックスサーチだけでOK

# Wrap up
## まとめ

- textsearch is fast but Asian langs aren't supported
  textsearchは速いけどアジア圏の言語を未サポート
- pg_bigm supports Japanese but is slow for large hits
  pg_bigmは日本語対応だがヒット数が多くなると遅い
- PGroonga is fast and supports all languages
  PGroongaは速くて全言語対応

# FYI: textsearch, PGroonga and Groonga



# Zulip and PGroonga

- Zulip uses textsearch by default
  Zulipはデフォルトでtextsearchを使用
  - Japanese isn't supported
    日本語非対応
- Zulip supports PGroonga as option
  ZulipでPGroongaも使うこともできる
  - Implemented by me
    私が実装

# Zulip: full text search
## Zulipと全文検索

- Zulip is chat tool
  Zulipはチャットツール
  - Latency is important for UX
    UX的にレイテンシーは大事
- Index update is heavy
  インデックスの更新は重い
  - Delay index update
    インデックスの更新を後回しにしている

# Delay index update
## インデックス更新を後回し

```
CREATE TABLE zerver_message (
  rendered_content text,
  -- ... ↓Column for full text search
  search_tsvector tsvector
); -- ↓Index for full text search
CREATE INDEX zerver_message_search_tsvector
  ON zerver_message
  USING gin (search_tsvector);
```

# Delay index update

## インデックス更新を後回し

```
-- Execute append_to_fts_update_log() on change
CREATE TRIGGER
  zerver_message_update_search_tsvector_async
  BEFORE INSERT OR UPDATE OF rendered_content
    ON zerver_message
  FOR EACH ROW
    EXECUTE PROCEDURE append_to_fts_update_log();
```

# Delay index update
## インデックス更新を後回し

```
-- Insert ID to fts_update_log table
CREATE FUNCTION append_to_fts_update_log()
  RETURNS trigger
  LANGUAGE plpgsql AS $$
    BEGIN
      INSERT INTO fts_update_log (message_id)
        VALUES (NEW.id);
      RETURN NEW;
    END
$$;
```

# Delay index update

## インデックス更新を後回し

```
-- Keep ID to be updated
CREATE TABLE fts_update_log (
  id SERIAL PRIMARY KEY,
  message_id INTEGER NOT NULL
);
```

# Delay index update
## インデックス更新を後回し

```
-- Execute do_notify_fts_update_log()
-- on INSERT
CREATE TRIGGER fts_update_log_notify
  AFTER INSERT ON fts_update_log
  FOR EACH STATEMENT
    EXECUTE PROCEDURE
      do_notify_fts_update_log();
```

# Delay index update
## インデックス更新を後回し

```
-- NOTIFY to fts_update_log channel!
CREATE FUNCTION do_notify_fts_update_log()
  RETURNS trigger
  LANGUAGE plpgsql AS $$
    BEGIN
      NOTIFY fts_update_log;
      RETURN NEW;
    END
  $$;
```

# Delay index update
## インデックス更新を後回し

```
cursor.execute("LISTEN ftp_update_log") # Wait
cursor.execute("SELECT id, message_id FROM fts_update_log")
ids = []
for (id, message_id) in cursor.fetchall():
  cursor.execute("UPDATE zerver_message SET search_tsvector = "
                 "to_tsvector('zulip.english_us_search', "
                              "rendered_content) "
                 "WHERE id = %s", (message_id,))
  ids.append(id)
cursor.execute("DELETE FROM fts_update_log WHERE id = ANY(%s)",
               (ids,))
```

# PGroonga: index update

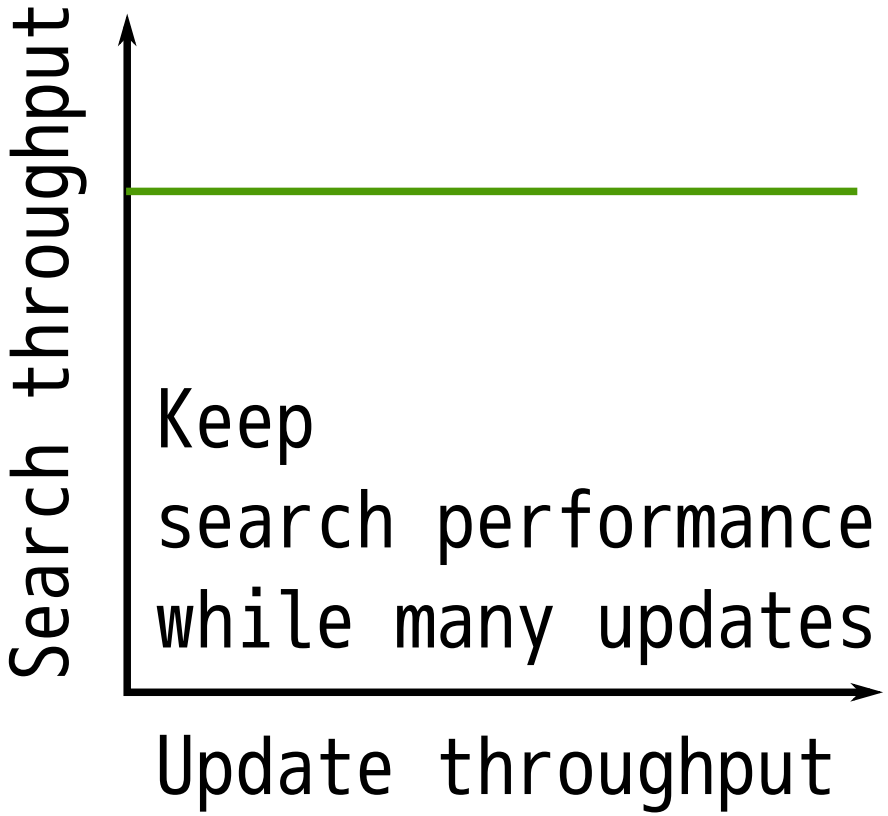
## PGroongaとインデックス更新

- PGroonga's index update is fast too
  PGroongaはインデックス更新も速い
- PGroonga's search while index update is still fast
  PGroongaはインデックス更新中の検索も速い

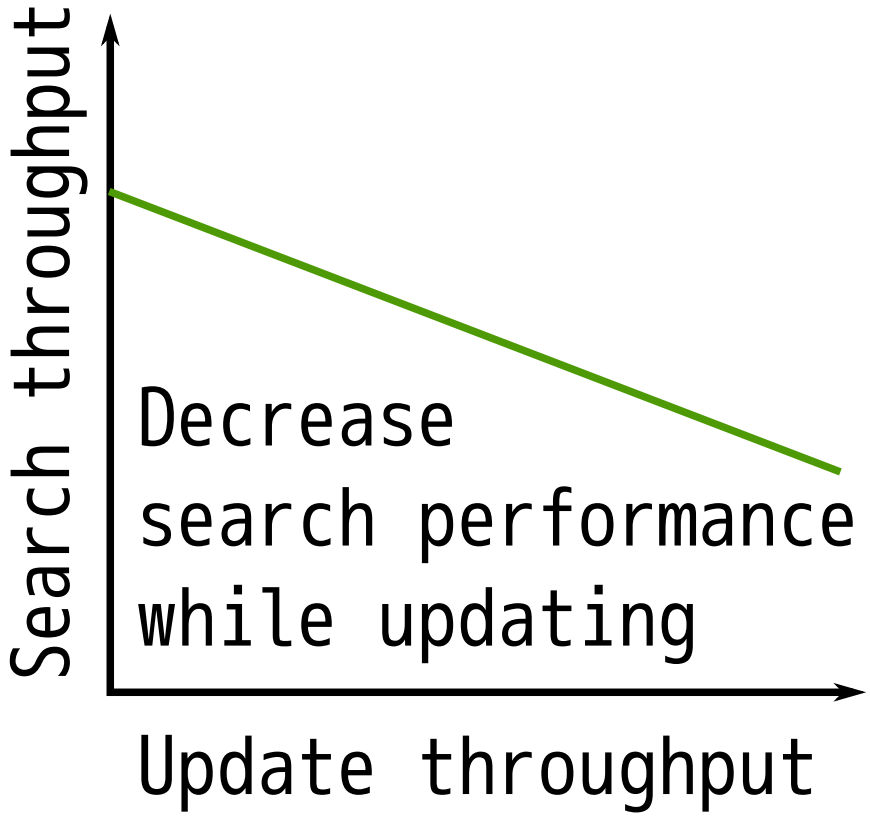
# Perf characteristics

## 性能の傾向

PGroonga




GIN

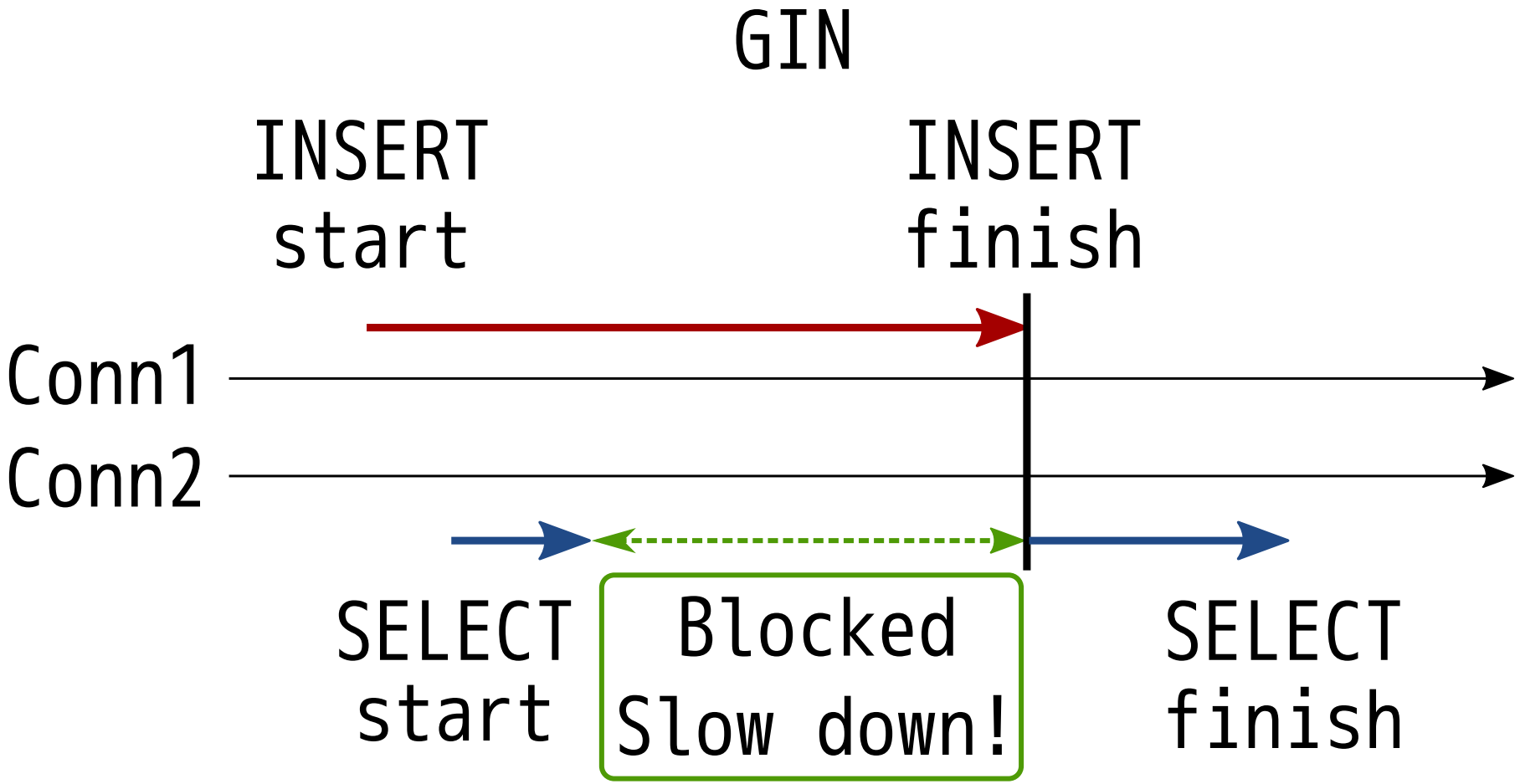


# Update and lock
## 更新とロック

- Update without **read** locks
  **参照**ロックなしで更新
  - **Write** locks are required
    **書き込み**ロックは必要

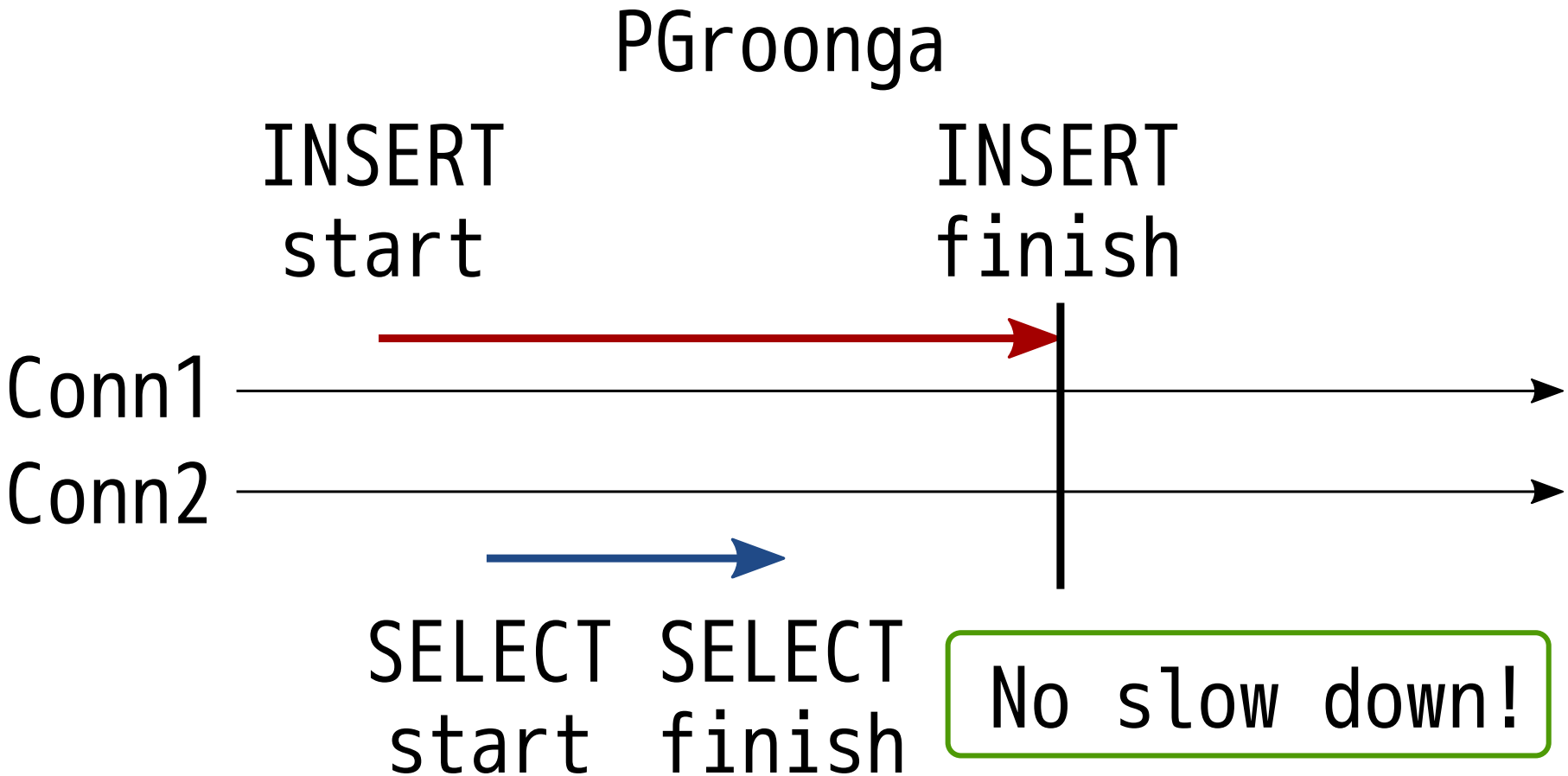
# GIN: Read/Write

## GIN：読み書き



# PGroonga: Read/Write

## PGroonga：読み書き



# Wrap up
## まとめ

- Zulip: Low latency for UX
  ZulipはUXのために低レイテンシーをがんばっている
  - Delay index update
    インデックスの更新は後回し
- PGroonga: Keeps fast search with update
  PGroongaは更新しながらでも高速検索を維持
  - Chat friendly characteristics
    チャット向きの特性

# More PGroonga features
## PGroongaの機能いろいろ

- Query expansion（クエリー展開）
  - Support synonyms（同義語検索をサポート）
- Similar search（類似文書検索）
  - Find similar messages
    類似メッセージ検索
- Fuzzy search（あいまい検索）
- Stemming（ステミング）