ICT活用総合実習
中間発表

中西 悠元 ／ 20122055

-研究タイトル-

Go言語におけるGORMとRaw SQLのデータベース操作性能比較

-GORMとは-

Go言語向けのORM（Object-Relational Mapping）ライブラリで、SQLを直接書かずにGoの構造体を使ってデータベースを操作できるようにするもの

-INSERT例-

```go
newUser := User{
    Name:  "Alice",
    Email: "alice@example.com",
}
// --- GORMでのINSERT ---
if err := db.Create(&newUser).Error; err != nil {
    return err
}

// --- Raw SQLでのINSERT ---
if _, err := sqlDB.Exec("INSERT INTO users(name,email) VALUES(?,?)", "Bob", "bob@example.com"); err != nil
    return err
}
```

-研究背景-

- ORMは開発効率を高める一方で、性能オーバーヘッドが指摘されている
- Go/GORMに関する体系的比較は乏しい
- 実務ではORMとRaw SQLの切替基準が不明確

-研究目的-

Go言語環境において、ORMとRaw SQLの間でデータベース操作性能にどの程度の差があるかを定量的に測定・分析し、開発現場での技術選択指針を提供する。

-副目的-

Go特有要素: 並行処理、メモリ管理、型安全性がORM性能に与える影響を分析

実務指針: どの規模・条件でGORM→Raw SQL切り替えが妥当かの定量的基準提示

-新規性-

- Go言語という文脈での比較

- 多くの比較記事は「サンプルコード単体」でクエリ速度を測るのみですがこの研究では 同一仕様のWebアプリケーション

- 性能以外の観点（保守性・開発効率）も含む

-研究方法-

データベースでよく使用される4つの処理

追加（INSERT）、更新（UPDATE）、削除（DELETE）、取得（SELECT）それぞれ単純なものから複雑なJOINまでテスト

GORM

(例) INSERT（データ量=10,000件、バッチサイズ=100、並行度=1）

実行時間=1.9s,Ops/sec=5,260
メモリ=85MB,p95=2.2s

(例) SELECT（JOIN、LIMIT 50、ヒット件数多、並行度=10）

実行時間=420ms、Ops/sec=238、
クエリ数/操作=1.0、p95=520ms

VS

Raw SQL

INSERT
実行時間=1.2s,Ops/sec=8,330
メモリ=62MB,p95=1.4s

SELECT
実行時間=310ms、Ops/sec=322、
クエリ数/操作=1.0、p95=380ms

-測定する項目-

①基本性能	実行時間,操作性,メモリ使用量など
②詳しい性能分析	平均や中央値など
③データベース関連	クエリ数など
④並行処理性能	スループット,競合など
⑤比較・分析の条件	データ量や並行度など
⑥ORMとRaw SQL	性能比較

-レイヤードアーキテクチャ-

Presentation（表示層）→実験結果の表示

Business（ロジック層）→ベンチマークの測定ロジック

Data Access（データ層）→DB操作（GORMとRaw SQL両方）

Infrastructure（基盤層）→DB接続などの設定

-開発の進め方-

Phase1　基盤構築(GORMで作る) Done

- データモデル（User, Post, Comment, Tag）を設計
- GORM実装（CRUD操作,バッチ操作,複雑クエリ）
- ベンチマークの基本フレームワークを構築
- テストデータ生成ツール作成

-開発の進め方-

Phase2 Raw SQL実装

- GORMの処理ロジックをRaw SQLで再現

  CRUD操作（INSERT, SELECT, UPDATE, DELETE）,バッチ処理,複雑クエリ

- 共通インターフェースを定義して切り替え可能に

  gormUserRepository と rawUserRepository の両方をこのインターフェースで実装

  実行時に --repo=gorm または --repo=raw のように切り替え可能にする

-開発の進め方-

Phase3　測定拡張

- **統計的分析**

  → 結果を統計的に信頼できる形で示す

  平均・中央値・標準偏差,p95/p99パーセンタイル

- **プロファイリング（原因分析）**

  CPU使用率、GC時間・回数

  メモリ・ブロックプロファイル

  → 処理のボトルネックを特定可能に

- **DB詳細指標（内部可視化）**

  クエリ数カウント、EXPLAIN結果

  インデックス使用状況、接続プール利用率

  → GORMとRaw SQLの動作をSQLレベルで比較

-開発の進め方-

Phase4　結果の分析

- 基本性能の比較分析
- 統計的検証
- プロファイリング分析
- DB挙動の分析

-今後のスケジュール-

10/31　　11/1----------　　12/1----------

中間発表　RawSQL実装　ベンチマーク結合　予備実験

------- 1/1----------　　2/5

本番実験　分析　論文　レビュー　最終修正　最終発表