Vol.2021-DBS-174 No.5
2021/12/27

# データベース管理システムの高速化に向けた プランナコストに基づいたパラメータチューニングの検討

丸尾 海月[1,a)]　新美 礼彦[2,b)]

**概要**：近年，ハードウェアの性能が向上しており，データベース管理システムを取り巻く環境が変化している．しかし，データベース管理システムのパラメータのデフォルト値は長らく変化していない．本研究では，現在のデータベース管理システムによりふさわしいパラメータ値があると考える．PostgreSQL を対象に，オプティマイザに着目し，問い合わせオプティマイザが使用するプランナコストのパラメータ値を自動パラメータチューニングにより変更することで，データベース管理システムの性能を向上させることができるかを検討する．

**キーワード**：データベース管理システム, パラメータチューニング, 自動化, オプティマイザ

# Parameter Tuning Based on Planna Cost in Database Management System

## 1. はじめに

データベース管理システム (以下 DBMS とする) とは，データベースを管理するためのミドルウェアである．DBMS は，データベースの定義，データの追加，更新，削除，整合性の維持といった機能を司る．DBMS では，システムの設定にパラメータを用いる．パラメータは数多く存在しており，パラメータは独立しておらず，互いに影響し合うという性質がある．

DBMS にはオプティマイザと呼ばれる，SQL を効率よく実行するための計画を作成する機能がある．オプティマイザは SQL を実行するための実行計画をいくつか作成し，それぞれの実行計画で実行する場合に必要な処理を，コストを用いて計算を行う．それぞれの実行計画のコストを計算し，最もコストが低くなる実行計画を選択する．例えば，PostgreSQL にはプランナコスト定数というパラメータがある．プランナコスト定数とは，オプティマイザが実行計画を評価するときに使用するコストの定数である．このプランナコスト定数を変更することで，オプティマイザが建てた実行計画を評価するコスト計算が変わる．これによって，オプティマイザが最適だと判断する実行計画が変化する．

近年，ハードウェアの性能が向上しており，DBMS を取り巻く環境が変化している．PostgreSQL では SSD をディスクとして用いている場合，プランナコスト定数の，random_page_cost を変更することをリファレンス上 [1] で推奨している．SSD は HDD に比べてランダムアクセスが高速であり，デフォルト値では SSD にふさわしい実行計画を作成することができないため，変更を推奨している．random_page_cost を始めとする，PostgreSQL のプランナコスト定数のデフォルト値は長らく変化していない．このことから，現在の DBMS を取り巻く環境に合った，適切なプランナコスト定数が存在するのではないかと考えた．しかし，各パラメータは独立していないため，各々のパラメータにふさわしい値を設定するのは非常に困難である．そこで，本研究では，DBMS のオプティマイザに着目し，オプティマイザが作成する実行計画を元に，最適なプランナコストを決定することができるのではないかと考える．

1 公立はこだて未来大学大学院システム情報科学研究科
Future University Hakodate,
Graduate School of Systems Information Science
2 公立はこだて未来大学システム情報科学部
Future University Hakodate,
Faculty of Systems Information Science
a) g2120048@fun.ac.jp
b) niimi@fun.ac.jp

ⓒ 2021 Information Processing Society of Japan

表 1　TPC-B による比較結果

| | HDD | SSD |
|---|---|---|
| Sustained Transfer Rate | 110MB/sec | 56MB/sec |
| レイテンシ | 8.33ms | 0.2ms |
| Transactions/sec | 864 | 3045 |
| CPU 使用率 | 20% | 65% |

表 2　TPC-C による比較結果

| | HDD*8(RAID0) | Enterprise SSD |
|---|---|---|
| Transactions/sec | 845 | 4026 |
| 消費電力 | 232W | 142W |
| CPU 使用率 | 14% | 54% |

## 2. 関連研究

DBMS をチューニングしている研究について記述する．

### 2.1 ストレージを単に SSD に置き換えて比較した研究

Sang-Won らの研究 (2008)[2] では，32GB の SSD と 250GB の HDD を DBMS のストレージとして用いて比較を行っている．古い SSD を対象に実験しているため，単純なデータ転送速度は HDD の方が高い．ベンチマークには TPC-B を使用している．SSD のほうが Transactions/sec が高く，CPU 使用率も高いということが判明した．

Sang-Won らの研究 (2009)[3] では，エンタープライズ SSD を用いた DBMS と 8 本の HDD を使用して RAID0 を構築した DBMS の比較を行っている．ベンチマークには TPC-C を使用している．8 本の HDD を使用するよりも，エンタープライズ SSD を使用するほうが高速かつ省電力であることが判明した．

### 2.2 DBMS を SSD 向きに改良を行った研究

Nakandala らの研究 (2014)[4] では，DBMS のオプティマイザが，ランダムアクセスを，シーケンシャルアクセスに比べて非常にコストが大きい処理であると解釈されているということに着目した．SSD では，HDD に比べてランダムアクセスは 100 倍以上高速である．そこで，SSD 向けに最適化するために積極的にインデックススキャンやを行うように DBMS の設定を変更し，DBMS を高速化している．

### 2.3 DBMS を自動パラメータチューニングするシステムを提案した研究

Dana らの研究 (2017)[5] では，機械学習を用いて DBMS を自動チューニングする OtterTune というシステムを提案している．DBMS のパラメータを変更し，変更した状態でベンチマークを行い，データの収集と学習を行いながらパラメータをチューニングする手法を提案している．

## 3. 提案手法

先行研究から，HDD と SSD には DBMS のストレージとして使う場合大きな差があることが判明した．また，DBMS のストレージとして SSD を用いる場合，プランナコスト定数の 1 種である，ランダムアクセスのコストをチューニングすることで DBMS の高速化が行えることが示唆された．これらから，DBMS のストレージに SSD を用いる場合，プランナコスト定数に改良の余地があることが示唆された．

DBMS のパラメータを自動チューニングする手法はすでに提案されているが，学習用のデータを集める必要がある．学習データを集めるためには，時間の掛かるベンチマークを実行し，手間を掛けてチューニングする必要がある．

本研究では，より素早く DBMS をチューニングするために，DBMS のオプティマイザが作成する実行計画を用いてプランナコスト定数をチューニングする手法を提案する．提案手法を検討するために，PostgreSQL を使用する．PostgreSQL には以下のプランナコスト定数が存在する．

- seq_page_cost (浮動小数)
- random_page_cost (浮動小数)
- cpu_tuple_cost (浮動小数)
- cpu_index_tuple_cost (浮動小数)
- cpu_operator_cost (浮動小数)
- effective_cache_size (整数)

先行研究では，プランナコスト定数のうち，シーケンシャルアクセスとランダムアクセスのコスト定数のみに着目していた．また，先行研究で自動パラメータチューニングの対象となっていたパラメータには．プランナコスト定数以外のパラメータも含まれていた．本研究では，ランダムアクセス以外のプランナコスト定数にも着目しているという点と，プランナコスト定数のみを用いて DBMS の自動パラメータチューニングを検討するという点で新規性があると考える．

実行計画は，PostgreSQL では ANALYZE にて容易に取得することができる．プランナコスト定数を変化させた時の実行計画を取得し，プランナコスト定数と実行計画のコストの合計値の関係から，パラメータ値に当たりをつけることができるのではないかと考える．実行計画のコストは，各処理の重さを推定し，実行計画を評価するために用いられているものであり，実実行時間と直接的な関係があるわけではない．しかし，プランナコスト定数を用いて実行計画に含まれる処理のコストを計算し，コストが最も小さくなる実行計画が選択されるため，プランナコスト定数を変更すると，選択される実行計画が変化する．オプティマイザが作成した実行計画のうち，総コストが最も小さい実行計画が選択されるため，実行計画の総コストと実行

計画には関係がある．つまり，プランナコスト定数を変更することで，実行計画の総コストが変化し，最終的にオプティマイザが効率的であると判断する実行計画が変化するため，プランナコスト定数と実行計画の総コストからプランナコスト値に当たりをつけることができるのではないかと考える．予めプランナコスト定数とコストをグリッドサーチし，パラメータ値とコストのデータセットを作成する．このデータセットに対し，コストを最小とするようなハイパーパラメータチューニングを行い，パラメータ値にいくつか当たりをつけた上でベンチマークを実行する．

## 4. 実験

実験目的，環境，手順，結果，考察について記述する．すでに行った実験 1 とこれから行う計画である実験 2 について記述する．

### 4.1 実験目的

DBMS を高速化するための 1 つの手段として，プランナコスト定数を用いてチューニングするのが妥当なのかを確認する．プランナコスト定数を変更することで，オプティマイザが作成する実行計画が変更されるかを確認し，変更された実行計画によってクエリの実実行時間が改善されるかどうかを検討する．

### 4.2 実験環境

実験環境には AWS を使用し，DBMS には PostgreSQL の 12.5 を用いた．

**OS** Ubuntu 20.04
**CPU** Xeon E5-2686 v4 @2.30GHz
**メモリ** 4GB
**I/O** EBS SSD 30GB
**DBMS** PostgreSQL 12.5
**ソフト** hammerDB 4.0

#### 4.2.1 ベンチマーク

Transaction Processing Performance Council[6] が制定した TPC-C と TPC-H と呼ばれるベンチマークを使用した．TPC-C ではあまり差が見られなかったため，実験では主に TPC-H を使用した．TPC-H のデータサイズは 10 倍に変更した．

### 4.3 TPC-H と PostgreSQL

PostgreSQL において，TPC-H を実行すると，特定のクエリで著しく時間がかかってしまうという問題が生じた．外部キー制約の宣言では参照列のインデックスが自動的に作成されることがないという PostgreSQL の仕様によって生じた問題であった．この問題を解決するために，インデックスを作成した．作成したインデックスは下記の通りである．

**表 3** query21 のコストと実行時間

| | random_page_cost=1 | random_page_cost=4 |
|---|---|---|
| cost | 1262407 | 2125845 |
| query 21 | 18193.674sec | 797.388sec |

- customer (C_NATIONKEY)
- lineitem (L_COMMITDATE)
- lineitem (L_ORDERKEY)
- lineitem (L_RECEIPTDATE)
- lineitem (L_SUPPKEY)
- lineitem (L_SHIPDATE)
- lineitem (L_PARTKEY,L_SUPPKEY)
- nation (N_REGIONKEY)
- orders (O_CUSTKEY)
- orders (O_ORDERDATE)
- partsupp (PS_PARTKEY)
- partsupp (PS_SUPPKEY)
- supplier (S_NATIONKEY)

インデックス作成前は効率の悪いネステッドループ結合が使われており，かつクエリの中にクエリが含まれるサブクエリになっていたため，著しく時間がかかっていたことが判明した．

### 4.4 実験 1

先行研究で効果があったとされる random_page_cost を変更し，実行計画が変化し，クエリの実実行時間が改善されるかどうかを検討した．

#### 4.4.1 実験手順

ベンチマーク用のデータベースを作成する．ベンチマークを実行するたびにベンチマークが不可逆に変更されるため，ベンチマーク実行前にデータベースをバックアップしておく．random_page_cost を変更し，ベンチマークを実行する．ベンチマーク終了後にデータベースをレストアし，再び random_page_cost を変更させ実験を行った．

#### 4.4.2 結果

random_page_cost を変更した上で実行計画を取得した．query20 ではオプティマイザが計算に使うプランナコストが変化しており，コスト計算上での差はあったが，作成される実行計画は全く同じものであった．次に，query21 では実行計画も変化していることが確認できた．しかし，SSD 向けにチューニングを行ったが，実実行時間は 20 倍ほど増えるという結果であった．

#### 4.4.3 考察

プランナコスト定数を変更することで，実行計画が変更されることは確認できた．しかし，今回変更したパラメータ値が適切ではなかったため，クエリの実行時間が増加してしまうという結果になった．オプティマイザが算出するコストは，あくまでクエリに含まれる各処理の重さを推定

しているだけであり，実実行時間と直接的な関係があるわけではない．しかし，コスト計算の結果から，オプティマイザが作成した実行計画のうち，最もコストが小さくなる，すなわち最も効率的だと判断した実行計画が選択されるため，コストと実実行時間の間には何らかの関係があると考えられる．

### 4.5 実験 2

実験 1 にて，プランナコスト定数を変更することで実実行時間が変化することを確認した．しかし，プランナコスト定数として不適切な値を設定してしまうと実実行時間が遅くなることが判明した．そこで，どのような値を設定するべきなのかという問題が生じた．理想的なパラメータ値の決定方法としては，パラメータ値を変更するたびに実実行時間を計測し，比較することである．しかし．ベンチマークを実行して実実行時間を取得するのは時間が掛かる．そこで，オプティマイザが作成する実行計画を取得し，実行計画からパラメータ値に当たりをつけることができるのではないかと考えた．予めプランナコスト定数とコストをグリッドサーチし，パラメータ値とコストのデータセットを作成する．このデータセットに対し，コストを最小とするようなハイパーパラメータチューニングを行い，パラメータ値にいくつか当たりをつけた上でベンチマークを実行する．実験によって，プランナコスト定数を用いて DBMS をチューニングするのが妥当かどうかを検証する．

#### 4.5.1 実験手順

プランナコスト定数を変更し，実行計画を取得する．取得した実行計画から，総コストを取得し，パラメータと総コストのデータセットを作成する．作成したデータセットを Optuna にて最適化を行う．Optuna[7] の結果を元にベンチマークを実行し，実実行時間からパラメータの妥当性を検討する．

#### 4.5.2 パラメータの最適化方法

DBMS には多種多様なパラメータが存在しており，各パラメータが互いに関係している．これらのパラメータは，ベンチマークを行い，結果と比較しながら調整する．この特徴は，機械学習のハイパーパラメータチューニングと同じ性質を持っている．ハイパーパラメータ自動最適化ツールである，Optuna を用いて DBMS のパラメータの最適化を行えると考える．

## 5. おわりに

本論文では，DBMS を高速化するための 1 つの手段として，プランナコスト定数を用いてチューニングするのが妥当なのかを検討した．これまでに，プランナコスト定数を変更することで，クエリを実行する計画が変更され，実実行時間が変化することを確認した．グリッドサーチを行い，プランナコスト定数とオプティマイザが算出するコストのデータを作成した．今後はグリッドサーチにて収集したデータの補間手法を検討し，作成したデータを用いて，プランナコストを自動パラメータチューニングにより最適化できるかを確認する．

## 参考文献

[1] PostgreSQL 12.4 文書 19.7. 問い合わせ計画. `https://www.postgresql.jp/document/12/html/runtime-config-query.html`. (Accessed on 11/24/2021).

[2] Sang-Won Lee, Bongki Moon, Chanik Park, Jae-Myung Kim, and Sang-Woo Kim. A case for flash memory SSD in enterprise database applications. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pp. 1075–1086, 2008.

[3] Sang-Won Lee, Bongki Moon, and Chanik Park. Advances in flash memory SSD technology for enterprise database applications. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*, pp. 863–870, 2009.

[4] Supun Nakandala, Sachith Withana, Dinu Kumarasiri, and Hirantha Jayawardena. A study on SSD aware scan operation optimization in PostgreSQL database. In *Information Technology Research Unit RESEARCH SYMPOSIUM, University of Moratuwa 2014 (ITRU - 2014)*, pp. 16–19, 2014.

[5] Dana Van Aken, Andrew Pavlo, Geoffrey J. Gordon, and Bohan Zhang. Automatic database management system tuning through large-scale machine learning. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pp. 1009–1024, 2017.

[6] TPC. `http://tpc.org/`. (Accessed on 11/24/2021).

[7] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 2623–2631, 2019.

ⓒ 2021 Information Processing Society of Japan