PostgreSQL インサイド

pgBadger でログファイルを解析し、統計レポートを作成する

技術を知る

- [ ] 導入／環境設定
- [ ] 移行
- [ ] 性能
- [ ] チューニング
- [ ] バックアップ／リカバリー
- [ ] 冗長化／負荷分散
- [x] 監視
- [ ] データ連携
- [ ] 災害対策
- [ ] 豆知識

PostgreSQL システムを長期的に運用する上で、データベースのレスポンスやパフォーマンスに劣化が生じていないか定期的に検証し、必要に応じてシステムを調整／改善していくことは、非常に重要です。
PostgreSQL の周辺ツールの 1 つである pgBadger は、PostgreSQL のログファイルを解析して統計レポートを出力することができるため、システム運用時のパフォーマンスの確認や改善に役立ちます。

1. pgBadger とは

pgBadger は、PostgreSQL のログファイルを解析して、多数の視点からの統計レポートを、日次、週次単位で作成することができます。統計レポートは、デフォルトで HTML 形式になっており、ブラウザーで開くことで、多くの結果がグラフで表示されます。なお、統計レポートをテキスト形式や、JSON形式で出力することも可能です。

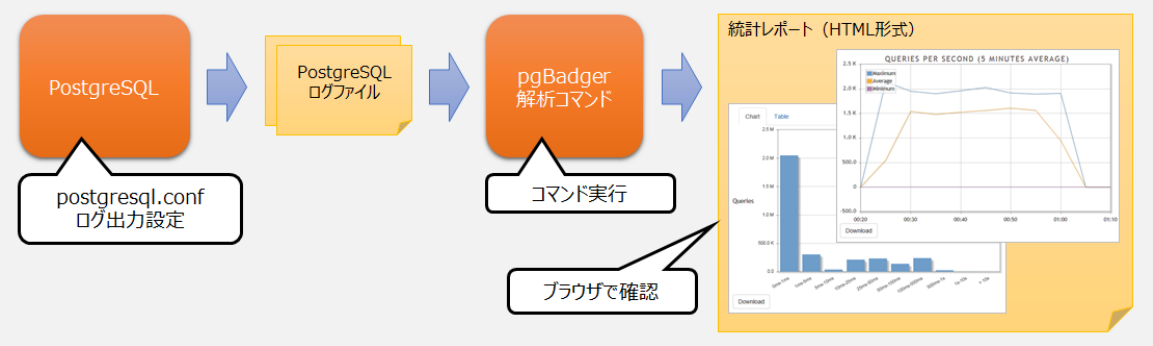

図 1 pgBadger の利用の流れ

統計レポートの内容は、以下の表のとおりです。データベースのパフォーマンスに影響を与えるクエリについてはランキング表示されるため、問題のあるクエリを早期に発見することができます。なお、pgBadger は、Linux 上で動作します。今回は、pgBadger 10.1 をベースに説明します。

表 1 pgBadger の統計レポートの内容

分類	統計レポートの内容
概要	時間あたりのクエリ実行数、ピーク日時、実行時間
コネクション	合計接続数、ピーク日時、データベース / ユーザー / 接続元ホスト別の接続数
セッション	合計セッション数、ピーク日時、セッション時間ごとの分布
チェックポイント	バッファ数、ピーク日時、実行時間、WAL 使用量、平均間隔

分類	統計レポートの内容
一時ファイル	合計ファイルサイズ、ピーク日時、合計ファイル数、平均サイズ、一時ファイルの生成回数 / サイズの大きい該当クエリのランキング
バキューム	autovacuum と autoanalyze の実行回数（注 1）
ロック	合計ロック数、ロックの待ち時間の長い該当クエリのランキング
クエリ	合計実行回数、キャンセルされたクエリ数とそのピーク日時、クエリ実行時間ごとの分布、スロークエリの各種ランキング、実行回数の多いクエリのランキング
サーバーログ	各レベルの合計ログ数、回数の多いエラーのランキング

- （注 1） PostgreSQL では、デフォルトで、自動 VACUUM が行われます。自動 VACUUM は、データベースの不要領域の回収（VACUUM）と、データベースに関する統計情報の収集（ANALYZE）を実行します。

2. pgBadger の使い方

pgBadger を利用するためには、まず、必要な情報がログファイルに出力されるよう、PostgreSQL の設定ファイル（postgresql.conf）のパラメーターを変更します。その環境で PostgreSQL を動作させることで、パフォーマンス分析に必要な情報がログファイルに出力されます。pgBadger は、そのログファイルを解析し、パフォーマンスに関する統計レポートを出力します。

2.1 pgBadger のインストール方法

Github から圧縮ファイルをダウンロードしコマンド実行により、Red Hat Enterprise Linux や Ubuntu などの Linux 上にインストールできます。詳細は、pgBadger のオフィシャルページにアクセスし、Documentation の INSTALLATION を参照してください。

ちなみに、pgBadger を Windows へインストールする手順に関しては、pgBadger のオフィシャルページで公開されていません。そのため、pgBadger は Windows をサポートしていません。

参考

- pgBadger のインストール（pgBadger のオフィシャルページへ）
  https://pgbadger.darold.net/

FUJITSU Software Enterprise Postgres では、バージョン 10 から pgBadger を同梱しており、インストールは不要です。

2.2 ログ出力設定（postgresql.conf ）

pgBadger がログファイルを解析する上で必要な postgresql.conf パラメーターは、以下のとおりです。なお、「設定例」の値は、pgBadger が解析時に必要とする情報を、すべてログに出力させるための設定値です。

表 2 ログファイル解析に必要な postgresql.conf パラメーター

パラメーター	デフォルト値	設定例	説明
log_min_duration_statement	-1（ログ出力しない）	0	クエリの実行時間が、指定した時間（ミリ秒）以上掛かったものについて、SQL 構文と実行時間をログに残す（0 を指定するとすべてのクエリが対象）。
log_duration	off	off	on を指定することで、すべての完了したクエリについて、その経過時間をログに残す。
log_statement	'none'	'none'	どの SQL 構文をログに記録するかを制御する。指定可能な値は、none（off）、ddl、mod。
log_line_prefix	'%m [%p] '	【標準エラー出力を使う場合】 '%t [%p]: user=%u,db=%d,app=%a,client=%h ' 【syslog を使う場合】 'user=%u,db=%d,app=%a,client=%h '	ログメッセージの頭につける情報を指定する。 %t = タイムスタンプ（ミリ秒単位はなし） %p = プロセス ID %u = ユーザー名（省略可） %d = データベース名（省略可） %a = アプリケーション名（省略可） %h = 遠隔ホスト名 / IP アドレス（省略可）
log_checkpoints	off （注 2）	on	on を指定することで、チェックポイントの実行をログに残す。
log_connections	off	on	on を指定することで、クライアントの接続をログに残す。
log_disconnections	off	on	on を指定することで、クライアントの切断をログに残す。
log_lock_waits	off	on	on を指定することで、deadlock_timeout で指定した時間（デフォ

パラメーター	デフォルト値	設定例	説明
			ルト 1 秒）以上のロック待ちをログに残す。
log_temp_files	-1（ログ出力しない）	0	一時ファイルが指定したサイズ（KB）以上であったら、作成されたことをログに残す（0 はすべて）。
log_autovacuum_min_duration	-1（ログ出力しない）（注 3）	0	autovacuum の活動で、指定された時間（ミリ秒）以上掛かったログを残す（0 はすべて）。
log_error_verbosity	default	default	エラーログの詳細度は、default を指定する。
lc_messages	initdb で指定した値	'C'	ログメッセージはロケール無効（'C'）を指定する。

- （注 2） PostgreSQL 15 から、「on」に変更されました。
- （注 3） PostgreSQL 15 から、「10min」に変更されました。

なお、これらすべての情報をログに出力すると、オーバーヘッドが発生します。そのため、解析の目的に応じて、必要な範囲を絞ることや、不要な項目をログに出力しないことも検討してください。
例えば、すべてのクエリの実行時間と回数については確認したいが、実行時間が 1 秒以上掛かるクエリの詳細情報しか見ない場合は、以下のようにパラメーターを設定します。

```
log_duration = on
log_min_duration_statement = 1000
```

なお、log_statement パラメーターは、SQL 構文をログに残す設定ができますが、使用する上で以下の注意点があります。

- log_min_duration_statement パラメーターや log_duration パラメーターと一緒には指定できません。
- log_statement パラメーターに'all'は指定できません。

2.3 ログファイルの解析と統計レポートの出力

実際に pgBadger を動作させて、ログファイルの解析と統計レポートの出力を行います。また、統計レポートの内容についても確認してみます。postgresql.conf の設定は、表 2 の設定例のとおりとします。サンプルデータの収集には、PostgreSQL に同梱されているベンチマークツール「pgbench」を利用します。ここでは、testdb というデータベース上で、50 クライアント、10000 トランザクションで pgbench を実行してみます。

```
$ pgbench -i testdb
$ pgbench -c 50 -t 10000 testdb
```


PostgreSQL のログファイル名を指定し、pgbadger コマンドを実行します。なお、統計レポートの出力先を-O オプションで指定します。

```
$ pgbadger $PGDATA/pg_log/postgresql-2019-05-31_152206.log -O /var/log/output
[=======================>] Parsed 18835219 bytes of 18835219 (100.00%), queries: 97789, events: 1
LOG: Ok, generating html report...
```

pgBadger のログファイル解析が完了すると、指定された出力先に out.html という名前の統計レポートが作成されます。なお、統計レポートは英語表記のみです。以下に、統計レポートの一部を紹介します。先頭画面の「Overview▼」などのメニューから、各種解析結果を切り替えることができます。

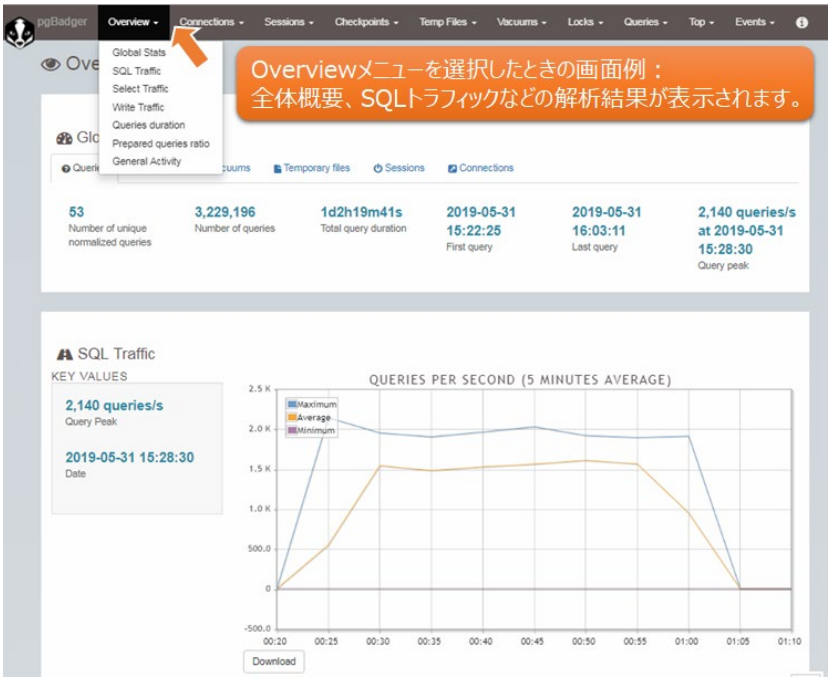

図 2 pgBadger の統計レポート例（Overview メニュー）

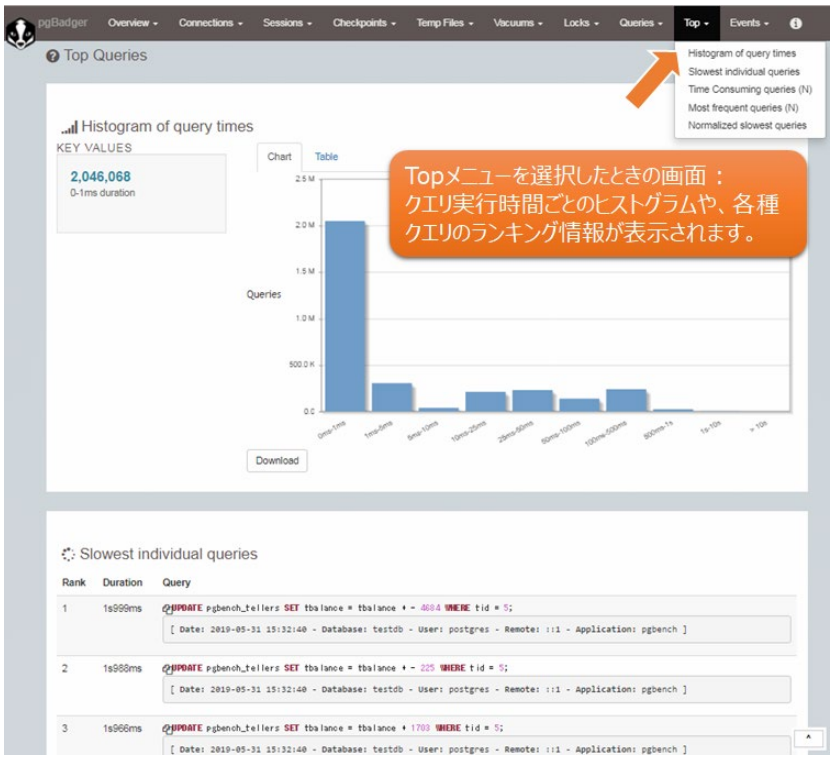

図 3 pgBadger の統計レポート例（Top メニュー）

継続的にログを解析して統計レポートを出力する

pgBadger を利用した運用を考えた場合、ログファイル解析と統計レポート出力は継続的に行う必要があります。そのためには、pgBadger のインクリメンタルモードを利用し、定期的にコマンドを自動実行することで解決できます。その際、統計レポートは、週次単位、および、日次単位に整理して格納されます。まずは、インクリメンタルモードを利用したときの pgBadger の動作を見てみましょう。pgbadger コマンドを実行するときに、-I オプションを指定します。

```
$ pgbadger -I $PGDATA/pg_log/postgresql-2019-05-31_152206.log -O /var/log/output
[=======================>] Parsed 504033059 bytes of 504033059 (100.00%), queries: 2601498, events: 265
LOG: Ok, generating HTML daily report into /output/2019/05/31/...
LOG: Ok, generating HTML weekly report into /output/2019/week-22/...
LOG: Ok, generating global index to access incremental reports...
```

pgBadger のログファイル解析が完了すると、以下のフォルダ構成のように、年 / 週 / 日の各フォルダと、先頭ページ、週次レポート、日次レポートが作成されます。また、pgBadger を次に実行したときに、一度解析した部分を省略して解析できるようにするため、LAST_PARSED ファイルに最終解析行が記録されます。

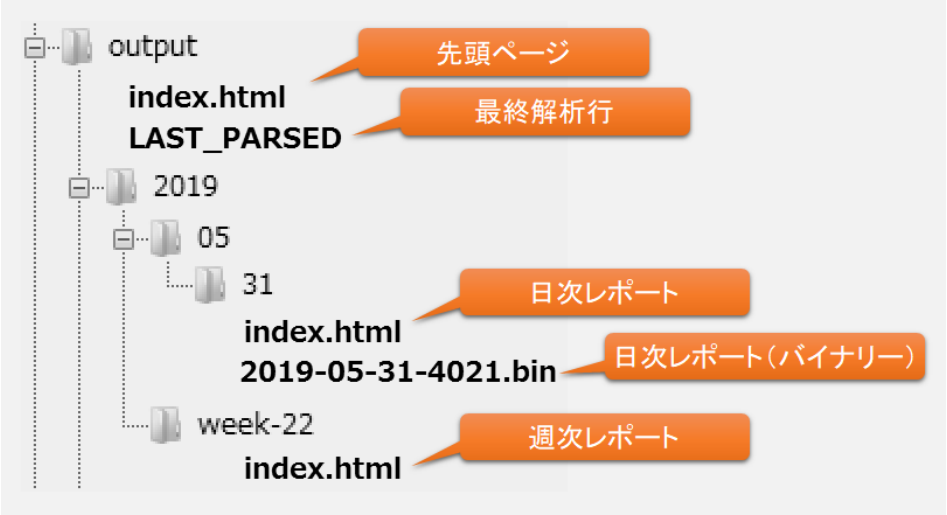

図 4 pgBadger のインクリメンタルモードによる統計レポート出力結果

先頭ページの index.html ファイルを開くと、以下の画面が表示されます。週次のリンク、日次のリンクから、それぞれの統計レポートを開くことができます。なお、インクリメンタルモードで統計レポートを蓄積していくことで、順次、カレンダーやリンクが追加されていきます。

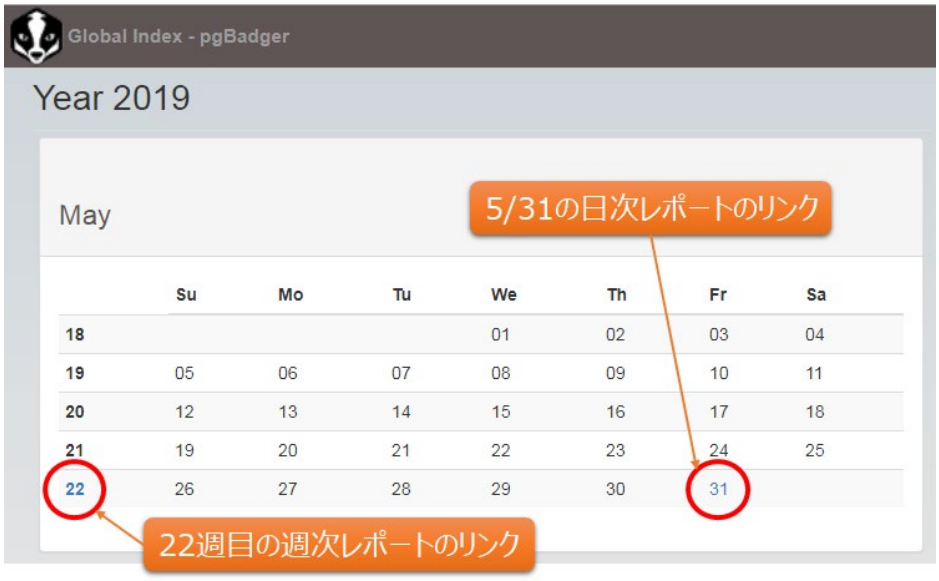

図 5 pgBadger 先頭画面

次に、pgBadger の自動実行の設定を行います。自動実行には、多くの UNIX 系 OS で標準的に利用できる常駐プログラム cron を使ってみます。cron は、定期的に繰り返しコマンドを実行することができます。以下の設定により、毎日 5 時に pgBadger を実行させて、統計レポートを蓄積していきます。
cron を設定するために エディタを起動します。

```
$ crontab -e
```

以下の 1 行を追加します。なお、-q オプションを指定することで、標準出力への結果表示を抑止しています。

```
0 5 * * * /usr/local/bin/pgbadger -I -q /usr/local/pgsql/data/pg_log/postgresql*.log -O /var/log/output
```

- （備考） cron の設定は、「分 時 日 月 曜日 コマンド」という順で指定します。

2.4 統計レポートの活用例

pgBadger が出力する統計レポートの中には、クエリの実行時間や頻度の順にランキングを出力している項目があります。これらのレポートを確認することで、早期にパフォーマンスのボトルネックを特定し、改善に役立てることができます。以下に、その例を示します。

スロークエリや、実行回数が多いクエリを見つける

クエリについては、以下の 4 つの観点でランキングを確認できます。それぞれのレポートを確認するための、メニューの場所も示します。

レポート内容	メニューの場所
最も遅い個々のクエリのランキング	Top - Slowest individual queries
合計実行時間の長い、正規化されたクエリのランキング	Top - Time Consuming queries (N)
合計実行回数の多い、正規化されたクエリのランキング	Top - Most frequent queries (N)
平均実行時間の長い、正規化されたクエリのランキング	Top - Normalized slowest queries

なお、メニューに“Normalized”（正規化）の文字や、“(N)”が付いている統計レポートは、クエリのパラメーターの値を「？」に置き換えて正規化したもので集計しています。これらの出力結果には、「Examples」などのボタンが付いており、このボタンを押して個々のクエリの該当データベース名、ユーザー名、アプリケーション名、実行した日時などを確認することができるため、データベースのパフォーマンスに影響を与えているクエリの検出に役立ちます。
以下に、統計レポート「Time Consuming queries (N)」の表示例を示します。

図 6 統計レポートのランキング例 「Time Consuming queries (N)」

ランキングされたクエリごとに、「Details」、「Examples」、「User(s) involved」、「App(s) involved」のボタンがあり、これらのボタンを押すことで詳細情報が展開されて表示され、さらに詳細を確認することができます。Rank の 1 番目について、これらのボタンを押したときの表示例を示します。

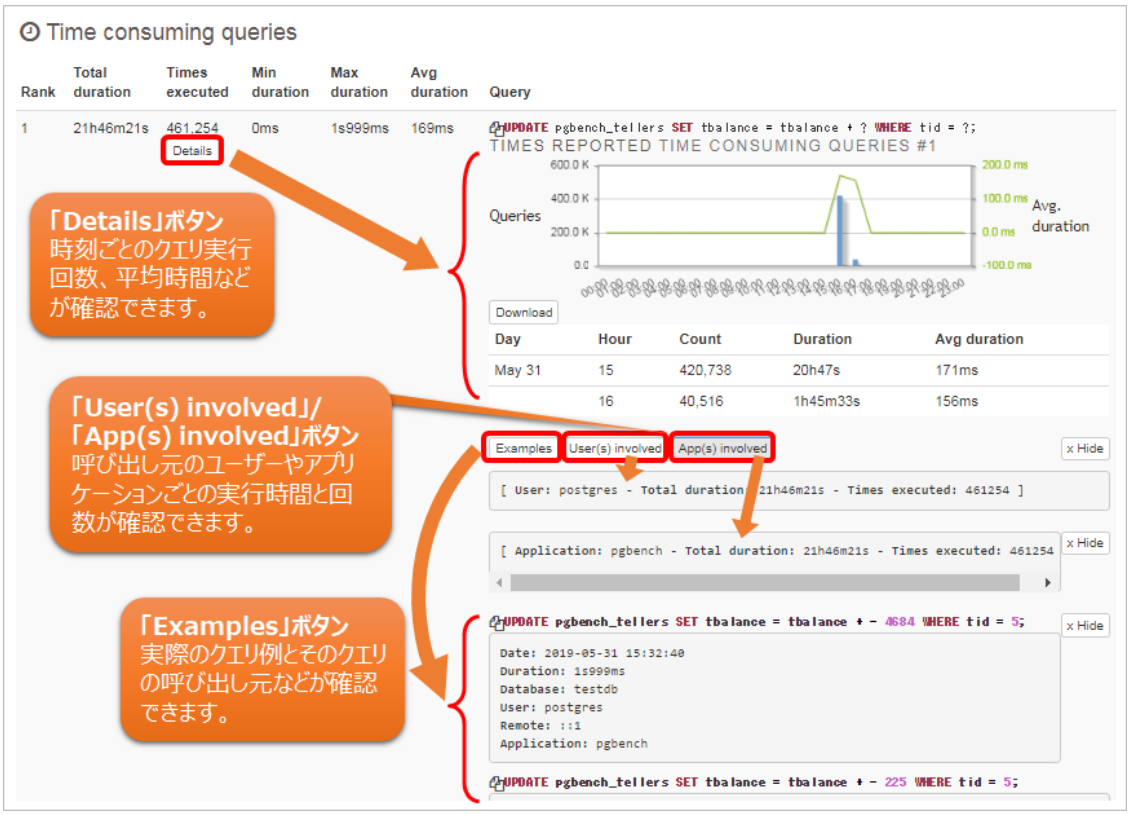

図 7 ボタン「Details」、「Examples」、「User(s) involved」、「App(s) involved」を押したときの例

一時ファイルを多く生成しているクエリを見つける

一時ファイルについては、以下の 2 つの観点でランキングを確認できます。

レポート内容	メニューの場所
一時ファイルの生成回数の多い、正規化されたクエリのランキング	Temp Files - Queries generating the most files (N)
一時ファイルの生成サイズの大きい、（個々の）クエリのランキング	Temp Files - Queries generating the largest files

一時ファイルについては、ソート処理やハッシュ処理、一時的な問い合わせの結果のために使われるデータがメモリ内に収まりきらない場合に生成されます。そのため、統計レポートから該当クエリ、一時ファイルの生成回数とサイズを確認し、クエリのメモリ使用量や、設定ファイル（postgresql.conf　の work_mem　など）について、妥当性を確認してください。
以下に、統計レポート「Queries generating the most files (N)　」の表示例を示します。

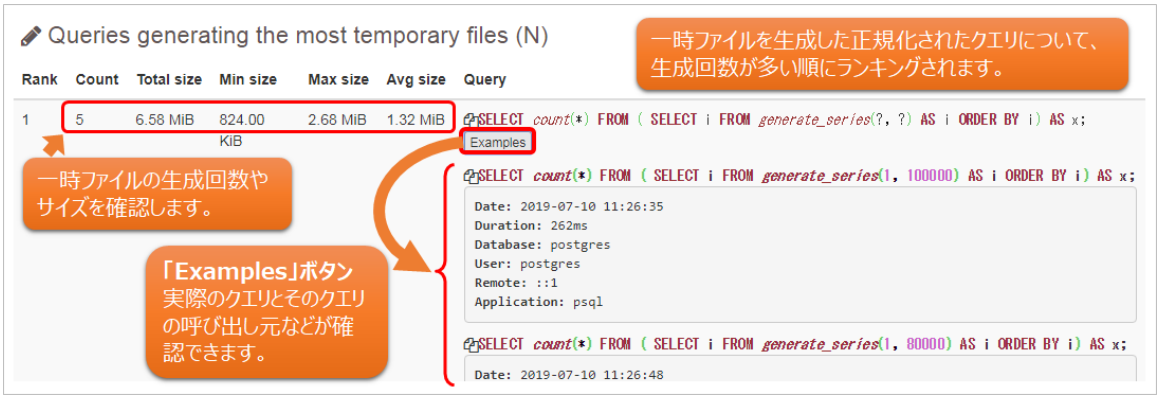

図 8 統計レポートのランキング例 「Queries generating the most files (N)」

ロックが発生しているクエリを見つける

ロックについては、以下の 2 つの観点でランキングを確認できます。

レポート内容	メニューの場所
待ち時間の合計の大きい、正規化されたクエリのランキング	Locks - Most frequent waiting queries (N)
待ち時間の長い、（個々の）クエリのランキング	Locks - Queries that waited the most

複数のトランザクションが同一資源にアクセスする際の待ち時間や回数が確認できます。想定より大きな結果が出ている場合は、必要に応じて、ロックモードや、トランザクション処理の妥当性などについて確認してください。
以下に、統計レポート「Most frequent waiting queries (N) 」の表示例を示します。

図 9 統計レポートのランキング例 「Most frequent waiting queries (N) 」

3. pgBadger 利用時の注意点

ここでは、実際に pgBadger を利用する際に発生する可能性のある課題とその対策について紹介します。また、他の統計情報収集ツールとの比較も載せますので、pgBadger の利用判断にご利用ください。

課題と対策

pgBadger の解析対象から、データベースをバックアップする時間帯を外したい

バックアップ中は、時間の掛かる SQL を実行する可能性が高いため、pgBadger の解析結果において、スロークエリのランキングの上位に入ってしまいます。それを避けるために、以下のような対策ができます。
例えば、2019 年 6 月の 13 時と 23 時の 2 回、pg_dump によるバックアップに、それぞれ 30 分程度掛かるような場合、--exclude -time オプションを使って、13 時台および 23 時台の時間帯を解析対象から外すことができます。

```
$ pgbadger --exclude-time "2019-06-.* (13|23):.*" $PGDATA/pg_log/postgresql-*.log -O /var/log/output
```

別の方法としては、--exclude-appname オプションを使い、アプリケーション名で除外することもできます。


```
$ pgbadger --exclude-appname "pg_dump" $PGDATA/pg_log/postgresql-*.log -O /var/log/output
```

pgBadger の統計レポートのグラフ目盛りがずれる

pgBadger の HTML 形式の統計レポート内のグラフは、JavaScript で描かれます。その際、日時の目盛りは GMT（グリニッジ標準時）が使われます。そのため、日本時間に合わせるためには、--timezone オプションを使い、「GMT+9:00」になるように指定します。

```
$ pgbadger $PGDATA/pg_log/postgresql-*.log --timezone +9 -O /var/log/output
```

Internet Exploere などで統計レポートを開くと、表示が崩れたり、グラフが表示されなかったりする

pgBadger の統計レポートを Internet Explorer などのブラウザーで開くときは、ブラウザーの設定を確認する必要があります。ブラウザーの設定にて、URL を信頼済みに設定、あるいは、ブロックされたコンテンツの許可を行うことで正しく表示されるようになります。

他の統計情報収集ツールとの比較

PostgreSQL のパフォーマンス監視や改善を目的とした統計情報の収集ツールや手法は様々あります。そのため、システムの環境やポリシーに見合ったツールや手法を選択する必要があります。ここでは、pgBadger に近い統計情報収集ツールである pg_statsinfo との簡単な比較を示しますので、参考にしてください。
なお、表中の「有り」「無し」は対応状態を表しています。

表 3 pgBadger と pg_statsinfo との比較

比較項目		pgBadger 10.1	pg_statsinfo 11.0
取得情報	コネクション	有り	無し
	セッション	有り	無し
	チェックポイント	有り	有り
	一時ファイル	有り	有り
	バキューム	有り	有り
	ロック	有り	有り
	クエリ	有り	有り
	サーバーログ	有り	有り
	データベース情報（トランザクション、容量、バックエンドプロセス、レプリケーションなど）	無し	有り

比較項目		pgBadger 10.1	pg_statsinfo 11.0
	OS リソース（CPU、メモリ、ディスク I/O、ロードアベレージ）	無し	有り
ツールの動作		データベース外で動作（単独コマンド）	データベース内のエージェントとして動作
情報取得方法		PostgreSQL サーバーログから取得	データベース内の各種テーブル / ビュー / 関数、PostgreSQL サーバーログ、OS 情報から取得
収集した統計情報の形式		HTML、テキスト、JSON	テキスト
統計情報の表示（グラフなど）		ツール出力結果の HTML を開く	別途、pg_stats_reporter を用いる
その他の機能		無し	サーバーログの蓄積やフィルタリング、アラート機能（エラー検出時の通知）

ここまで説明してきましたが、pgBadger は比較的簡単な手順で利用できることがわかりました。pg_statsinfo と比較すると、統計情報の対応範囲が狭いですが、システムポリシー上、データベース内に処理やオブジェクトを追加してのパフォーマンス分析が難しいケースにおいても利用可能です。利用シーンに合わせて、ご利用を検討してください。

参考

FUJITSU Software Enterprise Postgres に同梱している pgBadger は、富士通の 24 時間 365 日保守サポートにより、PostgreSQL 本体および pgBadger を含む周辺ツールのご質問、トラブル対応およびバグ修正にも迅速に対応しますので、ご利用を検討ください。

2023 年 4 月 12 日