PostgreSQL インサイド

データベースシステムの監視

監視方法と監視例

技術を知る

FUJITSU

- ☐ 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
- ☑ 監視
- ☐ データ連携
- ☐ 災害対策
- ☐ 豆知識

データベースシステムの監視の重要性については、「データベースシステムの監視 ～監視の概要～」で説明しました。ここでは、「死活監視、状態監視」、「メッセージ監視」、「容量監視」、「性能監視」に利用するコマンド、関数、統計情報ビューと、それらの実行例について説明します。なお、この記事は、PostgreSQL 11.1 をベースに作成しています。また、Linux を前提としています。

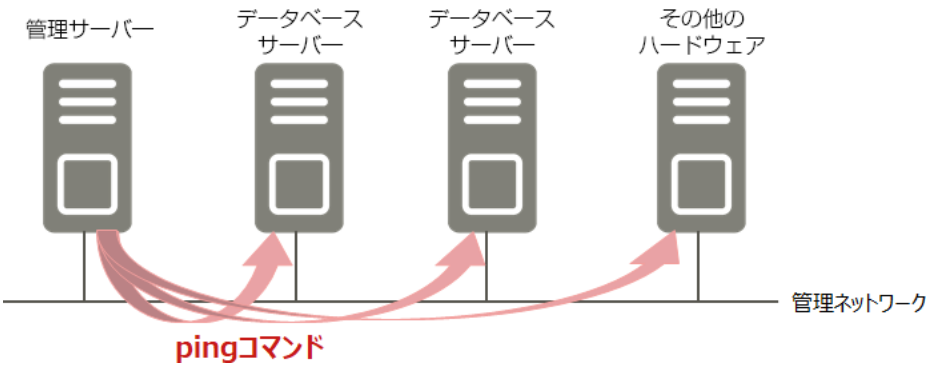
1. 死活監視、状態監視

サーバーと PostgreSQL の死活監視および状態監視は、OS のコマンドおよび PostgreSQL のコマンドで行います。以下に監視方法を示します。なお、表中の記号「N/A」は機能無しを意味しています。

分類	監視内容	監視方法	
		OS	PostgreSQL
死活監視	サーバーOS の応答	ping コマンド	N/A
	PostgreSQL プロセスの有無	ps コマンド	pg_isready コマンド
	SQL 実行可否 （SQL が正常に実行できるか確認）	N/A	「SELECT 1」などの簡単な SQL を実行
状態監視	CPU 使用率	sar コマンド vmstat コマンド mpstat コマンド	N/A
	メモリー使用率	free コマンド vmstat コマンド	N/A
	I/O ビジー率	sar コマンド iostat コマンド	N/A
	ネットワーク帯域使用状況	sar コマンド	N/A

各コマンドで示すオプションは一例になりますので、監視の方針に沿って他の有用なオプションとの併用も検討してください。コマンドの戻り値は、シェルスクリプトで確認するときに利用してください。

サーバーOS の応答（ping コマンド）



サーバーの死活監視は、OS の「ping コマンド」に「-c オプション」を指定して確認します。例のように「1」を指定すると 1 回試行します。ネットワークの一時的な断線などの影響を受けないようにするため、数回リトライするようにしてください。サーバーが動作している場合は、「1 packets transmitted, 1 received」と表示され、応答があったことがわかります。ping コマンドの戻り値は、0：応答を受信した、0 以外：応答を受信できない、または、エラーです。
ping コマンドは、管理サーバーなどの別サーバーから実行してください。

```
$ ping -c 1 192.0.2.1
PING 192.0.2.1 (192.0.2.1) 56(84) bytes of data.
64 bytes from 192.0.2.1: icmp_seq=1 ttl=57 time=1.56 ms

--- 192.0.2.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 1ms
rtt min/avg/max/mdev = 1.569/1.569/1.569/0.000 ms

$ echo $?
0
```

PostgreSQL プロセスの有無（ps コマンド）

PostgreSQL の死活監視は、規定の PostgreSQL プロセスが存在することを、OS の「ps コマンド」を使用して確認します。
PostgreSQL では、マスタープロセス（postmaster プロセス）の監視を推奨します。他の postgres 子プロセスの監視は必要ありません。一時的にマスタープロセスが 2 つ以上存在することがあるため、マスタープロセス数が 1 以上のときを正常としてください。
ps コマンドは、PostgreSQL データベースサーバー上で実行してください。

```
$ ps auxww | grep /bin/postgres
postgres  3297  0.0  0.0 107464   920 pts/1    S+   15:15   0:00 grep /bin/postgres
postgres 17099  0.0  0.0 264536  1828 ?        S    10月07   0:07 /usr/local/pgsql/bin/postgres
```

PostgreSQL プロセスの有無（pg_isready コマンド）

PostgreSQL の死活監視は、PostgreSQL の「pg_isready コマンド」でも確認することができます。PostgreSQL サーバーの接続状態を検査することで、PostgreSQL プロセスの有無を確認することができる簡易的なコマンドです。pg_isready コマンドの戻り値は、0：接続を受け付けている、1：サーバーが接続を拒絶している、2：応答がない、3：無効なパラメーターなどにより試行が行われないです。
pg_isready コマンドは、PostgreSQL クライアント（別マシン）からも実行できます。


```
$ pg_isready -h 192.0.2.1
192.0.2.1:5432 -接続を受け付けています

$ echo $?
0
```

SQL 実行可否（「SELECT 1」などの簡単な SQL を実行）

最低限の SQL が実行できるか、「psql –c」で簡単な SQL を実行して確認します。ここでは、「SELECT 1」を実行し、適切な結果が返ることを確認します。psql コマンドの戻り値は、0：正常、0 以外：異常です。
psql コマンドは、PostgreSQL クライアント（別マシン）からも実行できます。

```
$ psql -c "SELECT 1" postgres
 ?column?
----------
        1
(1 行)

$ echo $?
0
```

CPU 使用率（sar コマンド）

CPU 使用率は、OS の「sar コマンド」を使用して確認します。全体を取得する場合は「-u オプション」を、コア単位に取得する場合は「-P ALL」を指定します。例のように、「1 60」と指定すると、「1 秒おきに 60 回」取得することができます。CPU 使用率は、「%user」（ユーザーが利用している CPU 使用率）、「%nice」（nice 値を変更しているプロセスの CPU 使用率）、「%system」（システムが利用している CPU 使用率）で確認することができます。「%user」が上昇している場合、特定のアプリケーションで CPU が消費されている可能性があるので、ps コマンドなどで対象のアプリケーションを確認してください。
sar コマンドは、PostgreSQL データベースサーバー上で実行してください。

```
$ sar -u 1 60
Linux 3.10.0-693.11.6.el7.x86_64 (VM123456.localdomain)       2019年10月03日
_x86_64_        (2 CPU)

18時01分52秒     CPU     %user     %nice   %system   %iowait    %steal     %idle
18時01分53秒     all      0.50      0.00      0.50      0.00      0.00     99.00
18時01分54秒     all      0.00      0.00      0.50      0.00      0.00     99.50
18時01分55秒     all      0.00      0.00      0.00      0.00      0.00    100.00
18時01分56秒     all      0.00      0.00      0.50      0.00      0.00     99.50
18時01分57秒     all      1.01      0.00      1.01      0.00      0.00     97.99
                                        :
                                        :
```

メモリー使用率（free コマンド）

メモリー使用率は、OS の「free コマンド」を使用して確認します。バイト単位での表示では見にくいので、メガバイト単位で表示する「-m オプション」と、物理メモリーとスワップメモリーの合計を表示する「-t オプション」を指定します。メモリー使用率は、「total」（総容量）と「available」（スワップせずに利用可能と見積もられたメモリー）を使用し、「メモリー使用率 ＝（(total － available）÷ total × 100）」で確認することができます。
free コマンドは、PostgreSQL データベースサーバー上で実行してください。


```
$ free -m -t
              total        used        free      shared  buff/cache   available
Mem:           3782         848         926         390        2007        2205
Swap:          3967         373        3594
Total:         7750        1222        4520
```

- 備考　available は、Red Hat® Enterprise Linux® 7 から追加された項目です。

I/O ビジー率（iostat コマンド）

ディスクごとの I/O に関する情報は、OS の「iostat コマンド」を使用して確認します。デフォルトで、CPU 統計とディスク統計が表示されます。「-x オプション」を指定することで、より詳細なディスクの統計データを表示することができます。I/O ビジー率は「%util」（デバイスの帯域幅使用率）で確認することができます。また、「avgrq-sz」（リクエストの平均サイズ）、「avgqu-sz」（リクエストの平均キュー長）、「await」（リクエストの平均待ち時間（ミリ秒））から具体的な状況を確認することもできます。
iostat コマンドは、PostgreSQL データベースサーバー上で実行してください。

```
$ iostat -xkdt
Linux 3.10.0-693.11.6.el7.x86_64 (VM123456.localdomain)       2019年10月03日  _x86_64_      (2 CPU)

2019年10月03日  13時24分32秒
Device:         rrqm/s   wrqm/s     r/s     w/s    rkB/s    wkB/s avgrq-sz avgqu-sz   await r_await w_await  svctm  %util
sda               0.00     0.03    0.05    1.32     3.13    37.85    59.83     0.01    3.71    8.79    3.51   0.73   0.10
dm-0              0.00     0.00    0.05    1.34     3.11    37.79    58.99     0.01    3.70    8.86    3.51   0.72   0.10
dm-1              0.00     0.00    0.00    0.01     0.01     0.05     8.02     0.00   24.45    6.33   29.17   0.35   0.00
```

ネットワーク帯域使用状況（sar コマンド）

ネットワーク帯域に関する情報は、OS の「sar コマンド」に「-n DEV オプション」を指定して確認します。伝送路の使用状況は、「rxpck/s」（1 秒間に受信したパケット数）、「txpck/s」（1 秒間に送信されたパケット数）、「rxkB/s」（1 秒間に受信したパケットのキロバイト数）、「txkB/s」（1 秒間に送信されたパケットのキロバイト数）で確認することができます。
sar コマンドは、PostgreSQL データベースサーバー上で実行してください。

```
$ sar -n DEV
Linux 3.10.0-693.11.6.el7.x86_64 (VM123456.localdomain) 2019年10月03日  _x86_64_      (2 CPU)

00時00分01秒     IFACE   rxpck/s   txpck/s    rxkB/s    txkB/s   rxcmp/s   txcmp/s  rxmcst/s
00時10分02秒    ens192      0.40      0.17      0.09      0.01      0.00      0.00      0.01
00時10分02秒        lo      5.16      5.16      1.40      1.40      0.00      0.00      0.00
00時20分01秒    ens192    171.41    110.91    225.16      7.20      0.00      0.00      0.00
00時20分01秒        lo      2.75      2.75      1.00      1.00      0.00      0.00      0.00
00時30分01秒    ens192      0.22      0.00      0.03      0.00      0.00      0.00      0.00
00時30分01秒        lo      2.80      2.80      1.04      1.04      0.00      0.00      0.00

                    :         :
```

2. メッセージ監視

PostgreSQL は、インフォメーションやエラーなどのメッセージをサーバーログに出力しています。このログの内容を監視することで、エラーが発生した場合は、メッセージからエラー原因の分析を行い、対処することができます。

監視項目

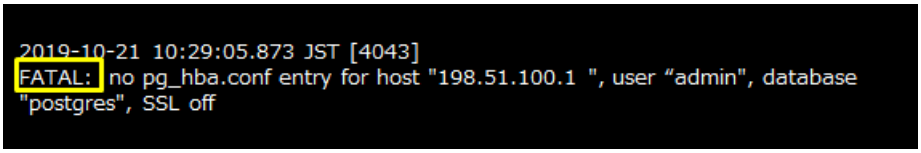
メッセージには、発生した問題の深刻度ごとに「メッセージレベル」が設定されます。「PANIC」「FATAL」「ERROR」はデータベースの安定稼働に影響する可能性があるため、この 3 つの監視をお勧めします。性能に関する監視を行う場合は、「LOG」も監視対象としてください。メッセージレベルと、レベルごとの意味を以下に示します。postgresql.conf の設定によりログの出力先を syslog に変更した場合、メッセージレベルは表の syslog 欄のレベルに変換されます。

メッセージレベル（深刻度）	意味	syslog
PANIC	すべてのデータベースセッションを中断させる原因となったエラーの情報	CRIT
FATAL	現在のセッションを中断させる原因となったエラーの情報	ERR
LOG	チェックポイントの活動のような、管理者が把握すべき情報	INFO
ERROR	現在のコマンドを中断させる原因となったエラーの情報	WARNING
WARNING	トランザクションブロック外での COMMIT のようなユーザーへの警告情報	NOTICE
NOTICE	長い識別子の切り詰めに関する注意など、ユーザーの補助になる情報	NOTICE
INFO	VACUUM VERBOSE の出力などの、ユーザーによって暗黙的に要求された情報	INFO
DEBUG1 から DEBUG5	開発者が使用する連続的で詳細な情報	DEBUG

以下にメッセージ例を示します。

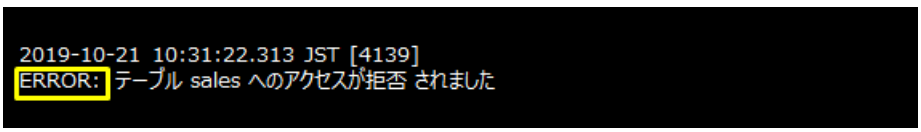
例 1

接続拒否（pg_hba.conf 外からの接続に対してのメッセージ）



例 2

ルールに反した接続（テーブルアクセス権限なし）



ポイント

エラーが発生した場合、ログに記録される「メッセージ」が、原因の特定や分析に役立ちます。また、PostgreSQL では、エラーメッセージのほかにエラーの発生箇所やヒント情報をログに出力します。これらを利用して、エラーの分析や調査を行ってください。

カテゴリー	内容
STATEMENT	エラー起因となった実際の処理内容
LOCATION	エラーが発生したコード上の位置
HINT	発生したエラーの原因や回避策
CONTEXT	エラーが発生したコンテキスト（関数など）

ログ出力の設定

メッセージ監視をするためには、メッセージをログファイルに書き出すように設定しておく必要があります。以下にメッセージ監視に必要な主な postgresql.conf のパラメーターを示します。

ログの出力先

- log_destination（ログの出力先）
- logging_collector（ログメッセージの内容をファイルに保存するかどうか）
- log_directory（ログファイルを格納するディレクトリー）
- log_filename（ログファイル名）

いつログを取得するか

- log_min_messages（サーバーログに書き込むメッセージのレベル）
- log_min_error_statement（エラー原因の SQL をサーバーログに書き込むメッセージのレベル）
- log_min_duration_statement（設定時間以上かかった SQL 文のみをサーバーログに書き込む）

何をログに出力するか

- log_checkpoints（チェックポイントおよびリスタートポイトに関する情報の出力の有無）
- log_connections（サーバーへの接続に関する情報の出力の有無）
- log_disconnections（サーバーの切断に関する情報の出力の有無）
- log_lock_waits（ロック獲得のために一定期間以上待たされたときの情報の出力の有無）

3. 容量監視

データベースクラスタ領域（PGDATA 環境変数で指定したディレクトリー）、TABLESPACE 領域、アーカイブしたトランザクションログ（WAL：Write-Ahead Log）を格納しておく領域のディスク容量が足りなくなると、以下のような支障をきたします。

- データの更新ができなくなる
- 新しいデータベースやテーブルが作成できなくなる
- WAL のディスク容量不足になると強制終了する
- データや WAL が破損する可能性がある

これらの問題を防ぐために、ディスクの使用率や空き容量を監視してください。そして、容量不足を起こす前に以下のような対処を行ってください。

- 不要なファイルの削除
- テーブル空間の機能を用いて、データベースや表などを複数のディスクに分散配置
- インデックスの肥大化が発生している場合は、REINDEX によるインデックスの再編成
- バックアップを実施することによって不要なアーカイブログの削除
- 新しいディスクを増設して、データを移動

例えば、他のツールと連携して「ディスクの空き容量が 20%未満になったらアラームを挙げる」などの設定をしておくことで、容量不足を起こす前にディスクを増設するといった対処をすることができます。この「20%」という閾値は、対処する時間も考慮して、余裕をもった値に設定してください。
PostgreSQL が扱うファイルで監視すべきものを以下に示します。なお、表中の記号「N/A」は機能無しを意味しています。

<table>
<tr><th rowspan="2">監視データ</th><th rowspan="2">監視対象
（ディレクトリー名）</th><th colspan="2">監視方法</th></tr>
<tr><th>OS</th><th>PostgreSQL</th></tr>
<tr><td rowspan="2">各テーブルやインデックス</td><td>$PGDATA/base</td><td rowspan="8">du コマンド
ls コマンド</td><td>pg_database_size 関数
pg_total_relation_size 関数
pg_table_size 関数
pg_indexes_size 関数</td></tr>
<tr><td>テーブル空間
（TABLESPACE）用に指定されたディレクトリー</td><td>pg_tablespace_size 関数</td></tr>
<tr><td>ディスクソートやハッシュ処理などの一時領域</td><td>$PGDATA/base/pgsql_tmp</td><td rowspan="5">N/A</td></tr>
<tr><td>サーバーログ</td><td>$PGDATA/log（注 1）</td></tr>
<tr><td>syslog</td><td>/var/log（注 2）</td></tr>
<tr><td>WAL</td><td>$PGDATA/pg_wal</td></tr>
<tr><td>WAL アーカイブ</td><td>WAL アーカイブ用に指定されたディレクトリー</td></tr>
</table>

- 備考　$PGDATA：データベースクラスタのディレクトリーです。
- （注 1）　postgresql.conf の log_directory パラメーターで変更できます。
- （注 2）　/etc/syslog.conf または/etc/rsyslog.conf で変更できます。

運用時のポイント

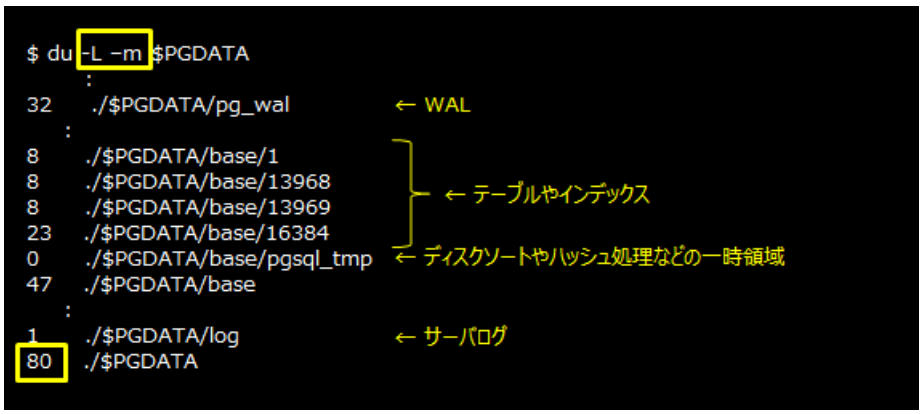
例えば、以下の SQL コマンドを実行する場合は、ディスクの空き容量に注意が必要です。

- VACCUM を実行
  WAL が大量に出力されるため、「$PGDATA/pg_wal」の監視が必要です。
- REINDEX や ALTER TABLE を実行
  対象のテーブルやインデックスがあるディレクトリーを圧迫するため、「$PGDATA/base」の監視が必要です。また、WAL が大量に出力されるため、「$PGDATA/pg_wal」の監視が必要です。

ここでは、「du コマンド」と「pg_database_size 関数」を例にとって説明します。

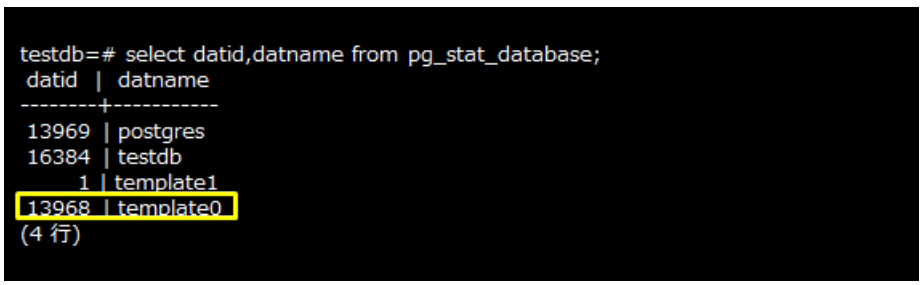
du コマンド

ディスクの使用量は、OS の「du コマンド」を使用して確認します。すべてのシンボリックを辿れるように「-L オプション」を指定します。また、サイズの表示単位を、「-k」（キロバイト単位）または「-m」（メガバイト単位）で指定します。以下のように指定することで、$PGDATA 配下のディスク使用量を、メガバイト単位で表示することができます。



pg_database_size 関数

データベースで使用されるディスク容量は、PostgreSQL の「pg_database_size 関数」を使用して確認します。データベースの情報が格納されているシステムカタログ（pg_database）から、データベースで使用されるディスク容量を取得し、サイズの単位をつけた書式に変換して表示します。



参考

データベースで使用されるディスク容量を du コマンドで確認する場合は、ディスクの使用量を確認する前にどのデータベースがどの oid かを調べ、その値を元に確認します。以下の例では、「testdb」の oid は「16384」とわかります。先に説明した du コマンドの結果を見ると、「./$PGDATA/base/16384」のディスクの使用量は「23」となっており、pg_database_size 関数の結果と一致します。

```
testdb=# SELECT datname, pg_size_pretty( pg_database_size(datname) ) FROM pg_database;
  datname  | pg_size_pretty
-----------+----------------
 postgres  | 8035 kB
 testdb    | 23 MB
 template1 | 7897 kB
 template0 | 7897 kB
(4 行)
```

4. 性能監視

性能に関する情報は、差分を見ることができるよう定期的に取得してください。最低でも 1 日 1 回の情報取得をお勧めします。監視の間隔については、「データベースシステムの監視 ～監視の概要～」の「3.2 監視間隔、異常とみなす値（閾値）」を参照してください。

PostgreSQL の「統計情報ビュー」には、稼働状況に関するさまざまな情報を収集し蓄積しています。「統計情報ビュー」を監視することで、性能に関する情報が取得できます。

監視対象	監視方法
遅いクエリの有無	pg_stat_statements ビュー
長時間放置されているトランザクション有無	pg_stat_activity ビュー
ロック待ち時間の長いクエリ有無	pg_locks ビュー pg_stat_activity ビュー pg_class カタログ
同時接続数	pg_stat_activiry ビュー
スループット（commit/rollback 回数）	pg_stat_database ビュー
レプリケーションの遅延状況（レプリケーション構成の場合）	pg_stat_replication ビュー pg_stat_wal_receiver ビュー
不要領域（データの挿入や削除によって発生した空き領域）	pg_stat_all_tables ビュー
インデックスのばらつき （インデックスの並び順とテーブルの並び順が揃っているか）	pg_stats ビュー

ポイント

統計情報ビューを表示する場合、表示形式のデフォルトは横長な状態のため、列数が多いと 1 行が折り返されて見づらくなってしまいます。メタコマンドで拡張テーブル形式モード（¥x）を有効にすることで、縦表示に変更することができ、見やすくなります。元に戻すときは、再度「¥x」を実行してください。

```
testdb=# ¥x
拡張表示は on です。
```

ここでは、「pg_stat_statements ビュー」、「pg_stat_activity ビュー」、「pg_stat_database ビュー」を例にとって説明します。

pg_stat_statements ビュー

「pg_stat_statements ビュー」を使用することで、遅いクエリの有無を確認することができます。「userid」（SQL 文を実行したユーザーの OID）、「query」（実行された SQL 文）、「calls」（SQL 文の実行回数）、「total_time（注 1）」（SQL 文の実行に費やした総時間）を確認することができます。
pg_stat_statements ビューを利用した例は、「チューニング ～ SQL チューニングの概要 ～」の「3.1 統計情報ビューを利用して検出する方法」および「チューニング ～ SQL チューニングを実施する ～」の「2. 処理が遅い SQL を検出」を参照してください。

- 注 1　PostgreSQL 13 から、パラメーター名が total_exec_time に変更されました。

pg_stat_activity ビュー

「pg_stat_activity ビュー」を使用することで、長時間放置されているトランザクションの有無や、同時接続数などを確認することができます。長時間放置されているトランザクションの有無は、「xact_start」（トランザクションが開始した時刻）を確認してください。トランザクション実行中でなければ値は入りません。何か値が入っていて長時間経過していれば、長時間放置されているということになります。同時接続数の確認は、pg_stat_activity ビューの行数（[RECORD x] の数）を確認してください。

```
testdb=# SELECT * FROM pg_stat_activity ;
-[ RECORD 1 ]----+------------------------------
       :
       :
-[ RECORD 3 ]----+------------------------------
datid            | 16384
datname          | testdb
pid              | 52101
usesysid         | 10
usename          | fsepyu
application_name | psql
client_addr      |
client_hostname  |
client_port      | -1
backend_start    | 2019-10-24 16:12:31.215987+09
xact_start       | 2019-10-24 16:18:33.466929+09
query_start      | 2019-10-24 16:18:33.466929+09
state_change     | 2019-10-24 16:18:33.466934+09
wait_event_type  |
wait_event       |
state            | active
backend_xid      |
backend_xmin     | 625
query            | select * from pg_stat_activity;
backend_type     | client backend
-[ RECORD 4 ]----+------------------------------
       :
       :
```

pg_stat_activity には他にも、「backend_start」（プロセスが開始した時刻）、「query_start」（クエリが開始した時刻）、「wait_event_type」（待機の状態）、「wait_event」（待機イベント名）、「state」（プロセスの状態）、「query」（最後に実行したクエ

リ)、「backend_type」（バックエンドの種別）のような、確認しておくべき項目がたくさんあります。これらも併せて確認してください。

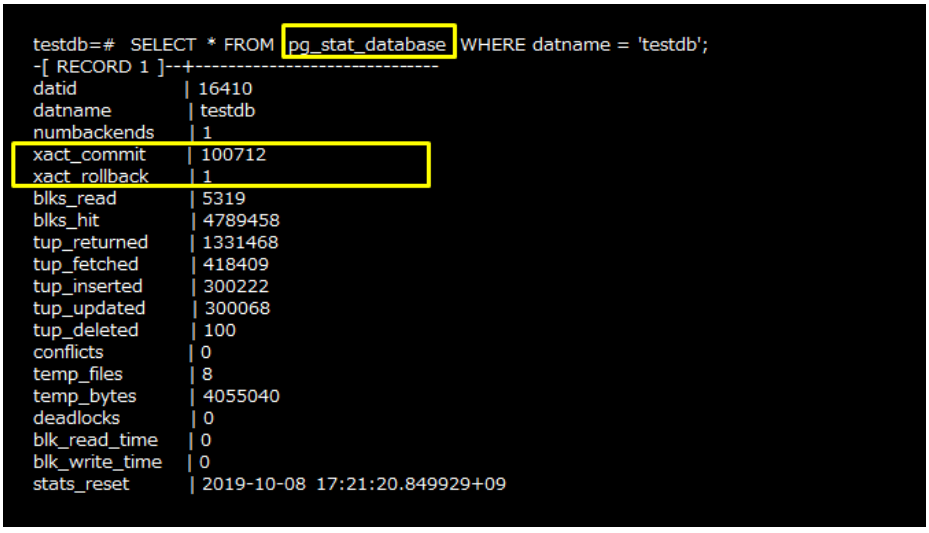
ポイント

「state」（プロセスの状態）に表示される値と状態を以下に示します。

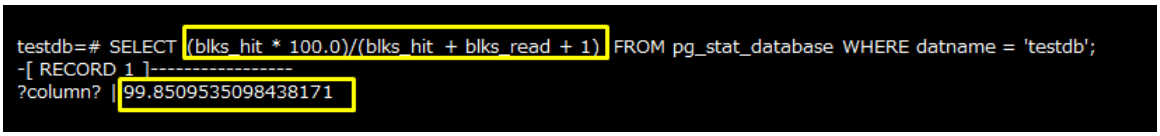
値	状態
active	問い合わせを実行中です。
idle	新しいクライアントからのコマンドを待機しています。
idle in transaction	トランザクションブロック内にいますが、現在実行中の問い合わせがありません。
idle in transaction（aborted）	idle in transaction と似ていますが、トランザクション内のある文がエラーになっています。
fastpath function call	近道関数（近道インターフェイスを利用する関数）を実行中です。
disabled	track_activities が無効（off）になっています。

pg_stat_database ビュー

「pg_stat_database ビュー」を使用することで、コミットやロールバック回数を確認することができるため、スループットを算出することができます。「xact_commit」（コミットされたトランザクション数)、「xact_rollback」（ロールバックされたトランザクション数）を定期的に取得してください。この情報を元に、「((今回取得時の xact_commit）－（前回取得時の xact_commit))÷ 情報取得間隔の時間」の計算をすることで、「単位時間あたりのコミット数」を確認することができます。同様に、「単位時間あたりのロールバック数」も計算し、確認してください。



また、「blks_read」（共用メモリーに無かったブロックの読み込み回数）と、「blks_hit」（共用メモリーにあったブロックの読み込み回数）を利用することで、キャッシュヒット率を算出することができます。メモリーチューニングでは、これを 100 に近づけることを目標とします。



データベースシステムのどのような情報を、どのように監視すれば良いのか、「監視対象」と「監視方法」に着目して説明しました。データベースシステムの要件に応じて様々な観点で監視を行うことで、異常やその兆候をより早く検出することができます。
可用性の高いシステムを実現するには、データベースシステムが安定稼動している状態を知り、その状態が維持されていることを監視し、異常時の対策を十分に行っておくことが重要です。

2021 年 1 月 22 日