PostgreSQLエンタープライズ・コンソーシアム 技術部会 WG#3

2018年度WG3活動報告書
Windows環境調査編(リソース監視、代替ツール)

目次

1. ライセンス

2. はじめに

2.1. PostgreSQLエンタープライズコンソーシアムとWG3について

PostgreSQLエンタープライズコンソーシアム(略称 PGECons)は、PostgreSQL本体および各種ツールの情報収集と提供、整備などの活動を通じて、ミッションクリティカル性の高いエンタープライズ領域へのPostgreSQLの普及を推進することを目的として設立された団体です。

PGECons 技術部会ではPostgreSQLの普及に資する課題を活動テーマとし、3つのワーキンググループで具体的な活動を行っています。

- WG1（新技術検証ワーキンググループ）
- WG2（移行ワーキンググループ）
- WG3（課題検討ワーキンググループ）

これら3つのワーキンググループのうち、WG1、WG3については 2015 年度まではそれぞれ、「性能ワーキンググループ」、「設計運用ワーキンググループ」という名称で活動してきました。2016年度は、従来の活動領域を広げる意図のもとでそれらを再定義し、上記のような名称に改めました。

これに伴い、WG3ではPostgreSQLの設計運用を中心としたさまざまな課題の解決のための調査検証を行い、PostgreSQLが広く活用される事を推進していくこととしました。

2.2. 本資料の概要と目的

2017年度、Windows上でのPostgreSQLに関して実際にWindows環境での検証と机上調査を通して情報を整備しました。その中で、特に周辺ツールについてはWindowsに対応していないツールが多いことが明らかになりました。

これを受け、2018年度はWindowsにおけるPostgreSQLについて、特に運用面(リソース監視、バックアップ)にフォーカスして調査・検証し、結果をまとめました。

本資料では、以下の内容をご紹介しています。

- リソース監視：Windows環境で使用できるリソース監視ツールの調査および実機検証に
- バックアップ：PostgreSQL本体の機能での代替可否の調査、整理

これにより、Windows上でのPostgreSQLの運用設計の際の参考として役立てていただくことを狙いとしています。

2.3. 本資料の構成

- はじめに
- Widnows環境でのリソース監視
  - Widnows環境のOS層のリソース監視
  - Widnows環境でのPostgreSQLのリソース監視
- Windows環境でのPostgreSQLのバックアップ
- 著者

2.4. 想定読者

本資料の読者は以下のような知識を有していることを想定しています。

- DBMSを操作してデータベースの構築、保守、運用を行うDBAの知識
- PostgreSQLを利用する上での基礎的な知識

Windows環境でのリソース監視

本章では、Windows環境でリソースを監視する方法について調査した結果を記載します。

1. 調査の背景および方針

2017年度のPGEConsの成果物「Windows環境調査編」では、調査した外部ツールにはWindowsに対応したモニタリングツールが存在しませんでした。
本年度は運用時に必須になるリソース監視においてWindows対応している外部ツールの調査を進めました。
OS層とPostgreSQLのリソース監視をするためにPrometheusというOSSを選定しました。Prometheusには数少ないWindowsのOS層を監視できるエージェント(wmi_exporter)が存在したのが理由です。

本章では、以下の調査結果を記載します。

- Windows環境でのリソース監視ができる外部ツールの調査
    - エージェント(exporter)からの情報を収集するリソース監視基盤
        - Prometheus
    - OS層のリソース監視
        - Prometheus + wmi_exporter
        - typeperf
    - PostgreSQL層のリソース監視
        - Prometheus + postgres_exporter

2. Prometheusの利用調査

WindowsのOS層のリソース監視をするOSSに Prometheus があります。
Prometheusは CFCN (Cloud Native Computing Foundation)でGraduatedステータスになっている監視ソフトウェアです。
Graduatedステータスとは CNCFの技術統括委員会が認めた成熟したコミュニティであることを示すステータスです。
Inception, Incubating, Graduatedと続きます。https://www.cncf.io/projects/
Prometheus特徴は以下になります。

- Go言語で実装されているのでどのOSでも動く
- pull型のアーキテクチャを採用 * Prometheusが能動的に登録サーバのリソース情報を取得する
- 各サーバでexporterと言われるagentを起動して、Prometheusの設定をすると監視対象になる

- exporterは特定のフォーマットを返すhttp://example.com:[port]/metrics APIを実装すれば自作できる
    - exporterはコミュニティや個人から大量に出ている

2.1. Prometheusの導入方法

2.1.1. 入手先

prometheusはGo言語で書かれたミドルウェアのため、バイナリを入手して起動するだけで簡単に動きます。
バイナリは 公式ページ内の「DOWNLOAD」 か GitHubページの「releases」 から取得できます。

2.1.2. 検証環境

検証には以下の環境を用意しました。

項目	バージョン
OS	Windows Server 2016 DataCenter
CPU	Intel Xeon CPU E5-2684 v4 @ 2.30GHz
Memory	16.0GB
Prometheus	2.7.1

2.1.3. 導入方法

展開したアーカイブの中の.exeファイルを起動するだけです。
同梱されているprometheus.yamlのファイルで設定します。
今回は後述するwmi_exporterとpostgres_exporterを配置するので、そのtargetの記載例を以下に示します。

```
global:
  scrape_interval: 5s
  scrape_timeout: 5s
scrape_configs:
- job_name: node
  static_configs:
  - targets:
    # wmi_exporter
    - localhost:9182
    # postgres_exporter
    - localhost:9187
```

上記設定が完了したらprometheus.exeを実行します。
Prometheusには他にも閾値によるアラートや簡易なUI(9090ポート)があります。
また Grafana と連携することで高級なUIによる分析もできます。

3. Windows環境のOS層のリソース監視

本章では、Windows環境のOS層のリソースを監視する方法の調査結果を記載します。
OSのリソース監視の方法として wmi_exporter と typeperf の2つを調査しました。

3.1. wmi_exporter

3.1.1. wmi_exporterの概要

wmi_exporterは上記で示したPrometheusのexporter(agent)の一種です。
exporterをWindowsサーバ上で常駐させることでPrometheusがリソース情報を収集します。
WMIはWindows Management Instrumentation の略称でありWindowsシステムの構成要素の情報収集や各種通知をするインターフェイスです。
wmi_exporterはそのインターフェイスを利用してリソースの情報を収集します。

3.1.2. wmi_exporterの利用方法

今回利用したwmi_exporterのバージョンはv0.6.0です。
wmi_exporterは GitHubからバイナリが配布 されているため簡単に起動できます。
取得したバイナリをcmd.exeから実行します。

```
C:\Users\Administrator>cd "C:\Program Files\exporter"

C:\Program Files\exporter>dir
Volume in drive C has no label.
Volume Serial Number is 7C6C-45DB

Directory of C:\Program Files\exporter

02/27/2019  05:15 AM    <DIR>          .
02/27/2019  05:15 AM    <DIR>          ..
02/27/2019  05:14 AM        13,268,992 postgres_exporter.exe
02/27/2019  05:14 AM        12,689,920 wmi_exporter-0.6.0-amd64.exe
               2 File(s)     25,958,912 bytes
               2 Dir(s)  10,447,429,632 bytes free

C:\Program Files\exporter>.\wmi_exporter-0.6.0-amd64.exe --collectors.enabled
"cpu,cs,logical_disk,net,os,tcp,system"
```

実行時のオプションの–collectors.enabledによって取得するリソースを指定できます。
取得できるリソースの一覧は README.mdのCollectorsを参照してください 。
起動すると9182ポートを受け付けるのでAPIを発行するとPrometheusが収集する値がわかります。

```
$ curl example.com:9182/metrics -i
HTTP/1.1 200 OK
Content-Length: 20465
Content-Type: text/plain; version=0.0.4; charset=utf-8
Date: Fri, 01 Mar 2019 08:48:49 GMT

(略)
# HELP wmi_cpu_time_total Time that processor spent in different modes (idle, user, system, ...)
# TYPE wmi_cpu_time_total gauge
wmi_cpu_time_total{core="0",mode="dpc"} 4.90625
wmi_cpu_time_total{core="0",mode="idle"} 24476.2490999
wmi_cpu_time_total{core="0",mode="interrupt"} 10.15625
wmi_cpu_time_total{core="0",mode="privileged"} 140.5625
wmi_cpu_time_total{core="0",mode="user"} 187.28125
wmi_cpu_time_total{core="1",mode="dpc"} 0.859375
wmi_cpu_time_total{core="1",mode="idle"} 24620.4848363
wmi_cpu_time_total{core="1",mode="interrupt"} 1.640625
wmi_cpu_time_total{core="1",mode="privileged"} 126.21875
wmi_cpu_time_total{core="1",mode="user"} 180.8125
wmi_cpu_time_total{core="2",mode="dpc"} 0.765625
wmi_cpu_time_total{core="2",mode="idle"} 24621.9940862
wmi_cpu_time_total{core="2",mode="interrupt"} 1.578125
wmi_cpu_time_total{core="2",mode="privileged"} 121.90625
wmi_cpu_time_total{core="2",mode="user"} 173
wmi_cpu_time_total{core="3",mode="dpc"} 0.8125
wmi_cpu_time_total{core="3",mode="idle"} 24389.811079699997
wmi_cpu_time_total{core="3",mode="interrupt"} 1.921875
wmi_cpu_time_total{core="3",mode="privileged"} 256.71875
wmi_cpu_time_total{core="3",mode="user"} 282.03125
(略)
```

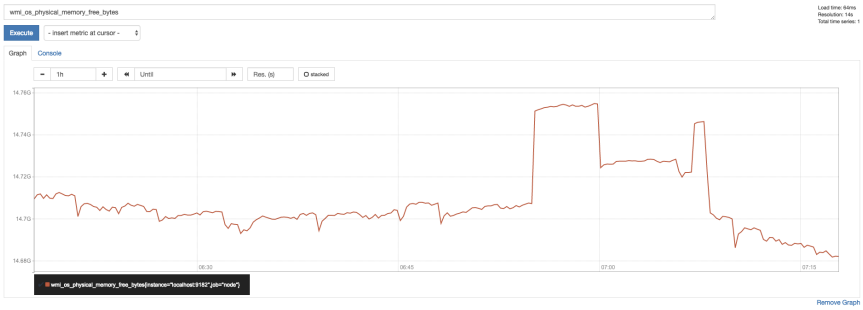
上記のようにexporterが公開しているhttp://example.com:9182/metricsのAPIエンドポイントをPrometheusが発行して値を収集しています。
ここではcpuの情報を一例に出しましたが、他にもmemoryやdiskなどの情報を取得しています。

3.1.3. 収集データの確認

収集したデータはPrometheusのGUIを使って時系列データとして確認できます。
GUIにアクセスするにはPrometheusを起動しているノードの9090ポートにアクセスします。
データを参照するときに PromQL を利用する場面もありますが、自動でグラフィカルなデータを確認できる強力さがあります。(PromQLは「積算値の参照」の章で利用例を示します。)
以下に収集したデータの参照例を示します。
以下は一例ですが、wmi_が接頭辞に付いているものはwmi_exporterが収集したデータになります。

3.1.3.1. 瞬間値の参照

wmi_os_physical_memory_free_bytes(メモリの空き)のような瞬間値はそのまま確認できます。
クエリ実行部分にwmi_os_physical_memory_free_bytesを入力してExecuteボタンを押下してGraphタブを開きます。
以下のようなグラフを参照できます。

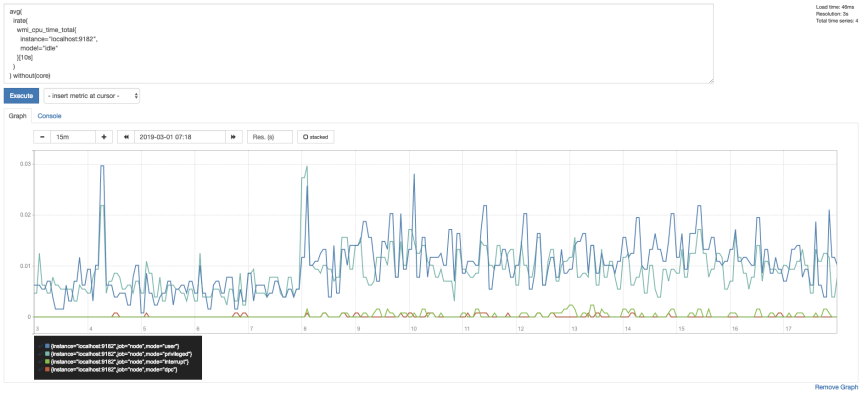


3.1.3.2. 積算値の参照

wmi_cpu_time_total(CPU使用率)やwmi_logical_disk_write_bytes_total(ディスク書き込み量)などの積算値はPromQLを使って時系列データを参照できます。
クエリ実行部に以下のようなPromQLを記載してExecuteボタンを押下します。

```
avg(
    irate(
        wmi_cpu_time_total{
            instance="localhost:9182",
            mode!="idle"
        }[10s]
    )
) without(core)
```

このクエリで以下のようなCPU使用率のグラフを参照できます。



3.2. typeperf

3.2.1. typeperfの概要

typeperfはWindows OSが標準で提供しているCUIツールです。
コマンドの説明は MicrosoftのHP に記載してあります。
Linuxのvmstatやsarのコマンド相当のリソースを取得できます。
取得できるリソース項目(counter)の一覧は typeperf -qx で取得できます。取得結果は以下になります。
typeperf_counter_list.txt

3.2.2. 利用方法

typeperfの-hを見ると以下が表示されます。

```
C:\Users\Administrator>typeperf -h

Microsoft ® TypePerf.exe (10.0.14393.0)

Typeperf writes performance data to the command window or to a log file. To stop Typeperf, press
CTRL+C.

Usage:
    typeperf { <counter [counter ...]> | -cf <filename> | -q [object] | -qx [object] } [options]

Parameters:
    <counter [counter ...]>       Performance counters to monitor.

Options:
    -?                            Displays context sensitive help.
    -f <CSV|TSV|BIN|SQL>          Output file format. Default is CSV.
    -cf <filename>                File containing performance counters to monitor, one per line.
    -si <[[hh:]mm:]ss>            Time between samples. Default is 1 second.
    -o <filename>                 Path of output file or SQL database. Default is STDOUT.
    -q [object]                   List installed counters (no instances). To list counters for one
object, include the
                                    object name, such as Processor.
    -qx [object]                  List installed counters with instances. To list counters for one
object, include the
                                    object name, such as Processor.
    -sc <samples>                 Number of samples to collect. Default is to sample until CTRL+C.
    -config <filename>            Settings file containing command options.
    -s <computer_name>            Server to monitor if no server is specified in the counter path.
    -y                            Answer yes to all questions without prompting.

Note:
    Counter is the full name of a performance counter in
    "\\<Computer>\<Object>(<Instance>)\<Counter>" format,
    such as "\\Server1\Processor(0)\% User Time".

Examples:
    typeperf "\Processor(_Total)\% Processor Time"
    typeperf -cf counters.txt -si 5 -sc 50 -f TSV -o domain2.tsv
    typeperf -qx PhysicalDisk -o counters.txt
```

typeperfはコマンドの引数にリソース取得項目(カウンタ)を指定できます。
カウンタはtypeperf -qxで取得できます。
カウンタは大量にあるので、今回は以下の様なリソース取得ファイルを用意しました。
typeperf_list.txt

```
PS C:\Users\Administrator> typeperf.exe -qx > .\typeperf_list.txt
PS C:\Users\Administrator> Get-Content list.txt
\Hyper-V VM Virtual Device Pipe IO(*)\Receive Message Quota Exceeded
\Hyper-V VM Virtual Device Pipe IO(*)\Receive QoS - Total Message Delay Time (100ns)
\Hyper-V VM Virtual Device Pipe IO(*)\Receive QoS - Exempt Messages/sec
\Hyper-V VM Virtual Device Pipe IO(*)\Receive QoS - Non-Conformant Messages/sec
\Hyper-V VM Virtual Device Pipe IO(*)\Receive QoS - Conformant Messages/sec
\Hyper-V Virtual Machine Health Summary\Health Critical
\Hyper-V Virtual Machine Health Summary\Health Ok
\Network Virtualization(Provider Routing Domain)\Unicast Replicated Packets out
\Network Virtualization(Provider Routing Domain)\Inbound Packets dropped
\Network Virtualization(Provider Routing Domain)\Outbound Packets dropped
(略)
```

typeperfのカウンタを整理したら、typeperf -cf [ファイルパス]のコマンドでリソースを取得できます。
取得間隔や回数やフォーマットもオプションで調整ができます。

- 出力オプション
    - -cf ファイルを指定することで取得するカウンタを指定する
    - -si 取得間隔[sec]

- -sc 取得回数[回]
- -f 出力ファイルフォーマット ex)csv, tsv, bin, SQL
- -o 出力ファイル

実際にコマンドを発行すると以下の様な結果を取得できます。

```
# 上記リソース項目を5秒間隔で3回取得して、typep.csvにcsv形式で取得します。
PS C:\Users\Administrator\Documents> typeperf.exe -cf typeperf_list.txt -si 5 -sc 3 -o typep.csv

The command completed successfully.
```

取得した結果は以下の様に出力されます。
typep.csv

4. Windows環境でのPostgreSQLのリソース監視

本章では、Windows環境でのPostgreSQL層のリソース監視について調査した結果を記載します。

- PostgreSQLをモニタリングするexporter(postgres_exporter)とPrometheusを組み合わせることによるモニタリングの試行
- pg_statsinfoでモニタリング可能なリソースとの機能比較

4.1. 調査に利用した環境

4.1.1. Prometheusの入手先

Prometheusのソースや利用方法などの情報は、公式ページ 配下より入手できます。また、Windows環境の実行バイナリは、公式ページ内の「DOWNLOAD」よりtar.gz形式で入手できます。

4.1.2. postgres_exporterの入手先

postgres_exporterのソースや利用方法などの情報は、プロジェクトのgithub 配下より入手できます。また、Windows環境の実行バイナリは、Release よりzip形式、tar.gz形式で入手できます。

4.1.3. 検証環境

以下の環境を利用して、postgres_exporterの動作検証を実施しました。

- OS:Windows 8.1 Pro 64bit
- CPU:Xeon E5-2697 v2 @ 2.70GHz (2プロセッサ)
- Memory:4GB

ソフトウェアのバージョンは、以下の通りです。

- Prometheus 2.7.1
- postgres_exporter 0.4.7
- PostgreSQL 11.0

また、検証のためPrometheusとPostgreSQLの環境は同一サーバとしました。

4.2. 環境構築

4.2.1. Prometheus、postgres_exporterのアーカイブの展開

ダウンロードしてきたアーカイブを展開します。

4.2.2. Prometheusの環境構築

postgres_exporterからのデータを収集するために、prometheus.ymlに以下を追記します。

```
- job_name: 'PostgreSQL'
  static_configs:
  - targets: ['localhost:9187']
# postgres_exporterが使用するデフォルトのポート番号が9187のため
```

4.2.3. 監視対象のPostgreSQLの環境構築

pg_stat_activity、pg_stat_replicationの情報を一般ユーザから取得するために、ユーザの作成と権限付与、ビューの作成などを実施します。

```
CREATE USER postgres_exporter PASSWORD 'password';
ALTER USER postgres_exporter SET SEARCH_PATH TO postgres_exporter,pg_catalog;

-- If deploying as non-superuser (for example in AWS RDS), uncomment the GRANT
-- line below and replace <MASTER_USER> with your root user.
-- GRANT postgres_exporter TO <MASTER_USER>
CREATE SCHEMA postgres_exporter AUTHORIZATION postgres_exporter;

CREATE VIEW postgres_exporter.pg_stat_activity
AS
SELECT * from pg_catalog.pg_stat_activity;

GRANT SELECT ON postgres_exporter.pg_stat_activity TO postgres_exporter;

CREATE VIEW postgres_exporter.pg_stat_replication AS
SELECT * from pg_catalog.pg_stat_replication;

GRANT SELECT ON postgres_exporter.pg_stat_replication TO postgres_exporter;
```

4.2.4. postgres_exporterの環境構築

以下の環境変数を設定します。

```
$ set DATA_SOURCE_NAME=postgresql://postgres_exporter:password@localhost:5432/postgres?
sslmode=disable
```

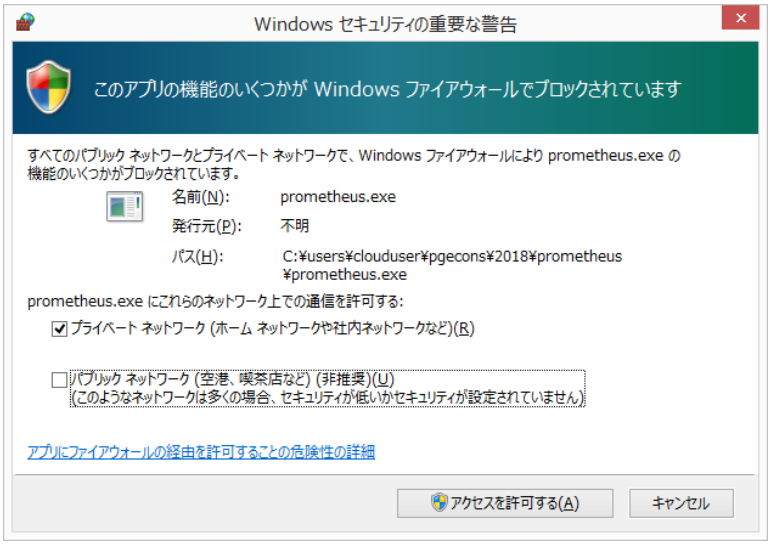
また、別途githubより入手したqueries.yamlを配置します。

4.2.5. Prometheus、postgres_exporterの起動

1. Prometheusを起動します。
2. postgres_exporterを起動します。

```
> postgres_exporter.exe --extend.query-path="[queries.yamlの配置先パス]"
```

Windowsのセキュリティ警告画面が出ることがありますが、ここではプライベートネットワーク内での通信に閉じるため、上のチェックボックスにチェックを入れて「アクセスを許可する」をクリックします。

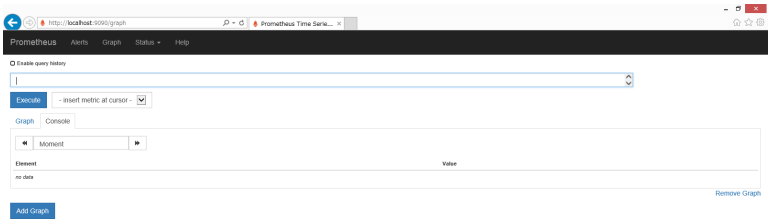


4.3. 動作確認

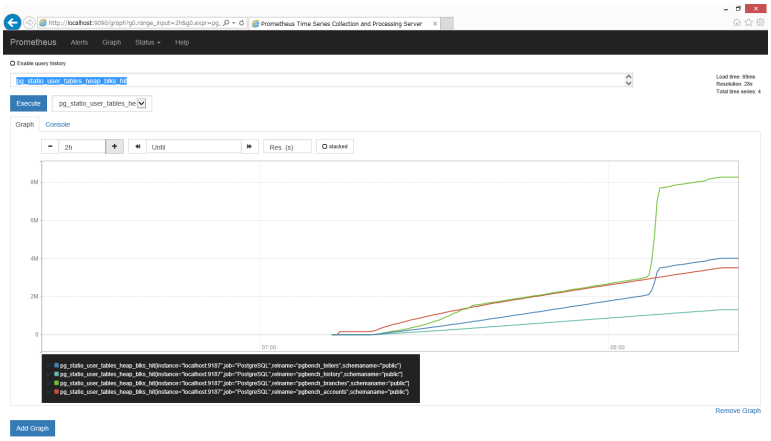
Prometheus、PostgreSQLをインストールしたサーバの9090番ポートにhttpでアクセスします。

```
http://localhost:9090
```

以下の画面が表示されるので、Graph →「-insert metrics at a Cursor-」のあるプルダウンをクリックし、PostgreSQLに関するメトリクスが表示されることを確認します。



以下の画面は、pgbenchをスケールファクタ100で1時間程度実施した際の例です。



5. Windows環境でのリソース監視のまとめ

Windows環境でのリソース取得をPrometheusとWindows標準機能を使って調査しました。
それぞれのメリットとデメリットを以下にまとめました。

実現ツール	メリット	デメリット
Prometheus	・agnetを配置するだけで取得リソースを増やせる ・PostgreSQL層の情報もある程度取得できる ・GUI(Prometheus, Grafana)を利用することで簡単にグラフ化したリソースをリアルタイムで確認できる ・Prometheusの他の機能(アラートなど)を利用できる ・さらに詳細を取得したい場合はexporterを自作できる	・運用するミドルウェアが増える
typeperf	・Windowsの標準機能のため、新規のミドルウェアを導入する必要がない ・wmi_exporterに比べると詳細な情報が取得できる	・GUIがないのでリアルタイムで値の確認がしづらい

5.1. pg_statsinfoとの機能比較

PostgreSQLのリソース監視にはLinux版のPostgreSQLで多く利用されます。
pg_statsinfoが収集できるリソース項目は 2015年のWG3の調査したもの があるので、今回の調査結果と比較しました。
比較結果は以下の通りです。
pg_statsinfoと今年度の調査結果の機能比較（pdf形式）
上記のファイルを参照するとpg_statsinfoほど詳細なデータを取得はできません。
より詳細なPostgreSQLの監視をするには自身でツールを作る必要性があります。
一方で既存の物を組み合わせるだけでも一般的なリソース監視の枠組みは実現できることがわかりました。

3. Windows環境でのPostgreSQLのバックアップ

3.1. 調査の背景および方針

2017年度のPGEConsの成果物「Windows環境調査編」では、周辺ツールのうち、Windowsに対応しているPostgreSQLのバックアップツールがないことを記載しました。

本章では、以下の調査結果を記載します。

- pg_rmanとPostgreSQLのコマンドラインとの機能比較

3.2. 調査に利用した環境

3.2.1. 調査方針

2017年度のPGEConsのWG3で調査したバックアップツール3つのうち、Googleでの検索結果、Googleトレンドの結果を勘案し、利用頻度が高いと考えられるpg_rmanを選定しました。

また、調査方法としてはソースを見ながら内部ロジックを確認する机上調査を実施しました。

3.2.2. pg_rmanの入手先

pg_manのソースや利用方法などの情報は、pg_rmanのgithub より入手できます。

3.3. pg_rmanとの機能比較

pg_rmanでのコマンド例を基準にした、Windowsでの代替手段の調査結果は以下の通りです。

pg_rmanの機能とWindows版PostgreSQLでの代替手段の調査結果（pdf形式）

3.4. まとめ

本検証では、pg_rmanとPostgreSQLのコマンドラインとの機能比較を実施しました。

- 基本機能となる全体のバックアップとリストアについては、PostgreSQLのコマンドラインで代用可能です。
- 一方、以下の機能は独自のロジックや運用でカバーする必要があります。
    - バックアップカタログの作成や参照
    - 増分バックアップ
    - バックアップファイルの削除
    - バックアップの検証

4. 著者

（企業・団体名順）

版	所属企業・団体名	部署名	氏名
第1.0版 (2018年度WG3)	NTTテクノクロス株式会社	IoTイノベーション事業部	勝俣 智成
	NTTテクノクロス株式会社	IoTイノベーション事業部	山本 育
	富士通株式会社	ミドルウェア事業本部	山本 貢嗣