

DB統計情報取得/可視化ツール pg_statsinfo と pg_stats_reporter のご紹介

**NTT OSSセンタ
近藤 光正**

自己紹介

• 正式な所属名称

- 日本電信電話株式会社 サービスイノベーション総合研究所
ソフトウェアイノベーションセンタ
研究員

• 担当業務

- PostgreSQL関連ツールの研究開発
 - pg_statsinfo (DB統計情報監視ツール)
 - pg_stats_reporter (DB統計情報可視化ツール)
 - PG-REX (PostgreSQLの高可用クラスタ PG + Pacemaker + pgsqIRRA)
- PostgreSQL のコミュニティ開発
 - 特に Disk IO 周りの改善

• 過去の業務

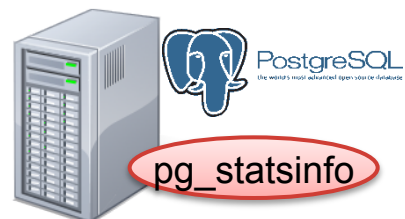
- データマイニング、自然言語処理、機械学習、情報推薦、情報検索
 - こっちの方が得意

• 趣味

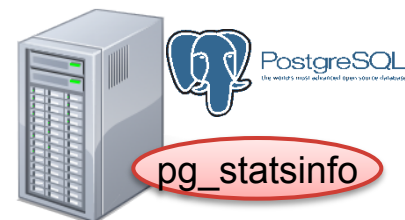
- カメラ
- オーディオ

本日紹介するツールの概要

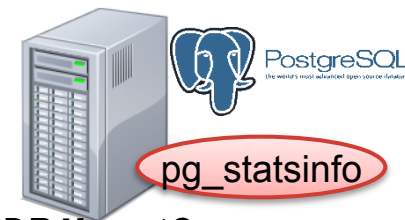
- **pg_statsinfo**
 - PostgreSQL の統計情報を取得/蓄積するツール
- **pg_stats_reporter**
 - pg_statsinfo で取得/蓄積した統計情報の可視化ツール



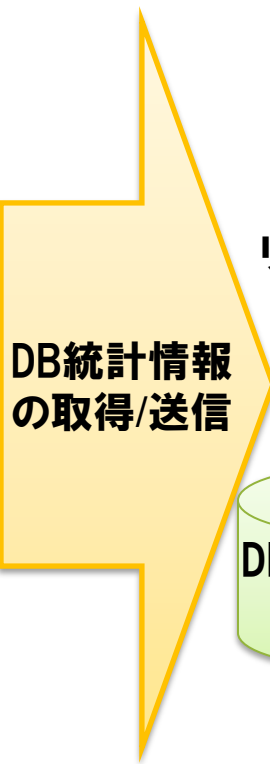
DBサーバA



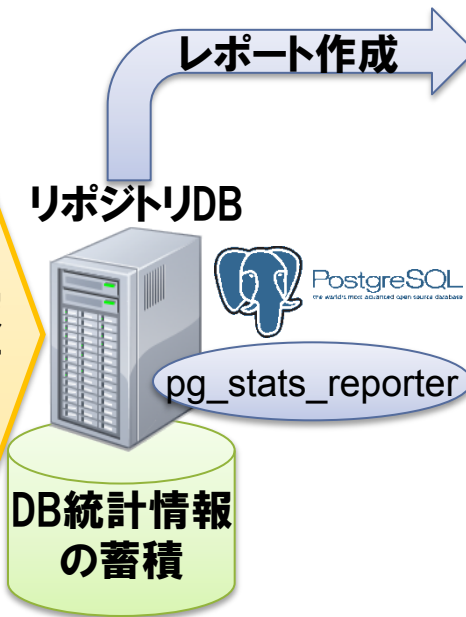
DBサーバB



DBサーバC



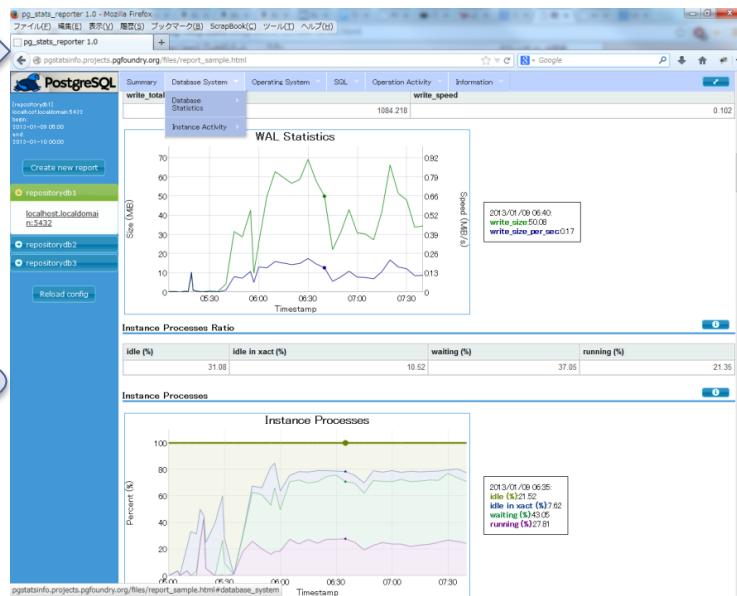
DB統計情報の取得/送信



リポジトリDB

DB統計情報の蓄積

レポート作成



pg_stats_reporter で作成したレポート

本日のアジェンダ

- **DB 統計情報取得ツール pg_statsinfo**
 - 概要
 - 機能紹介
 - デモ
- **DB 統計情報可視化ツール pg_stats_reporter**
 - 概要
 - 機能紹介
 - デモ
- **DBT-2 ベンチマークの可視化**
 - DBT-2 ベンチマークの概要
 - 作成されたレポートの紹介
 - チューニング TIPS

pg_statsinfo の概要

- PostgreSQL に格納された統計情報を取得/蓄積するツール

- 取得する統計情報

- 統計情報テーブル(pg_catalog)の定期的な取得/蓄積
- ログ情報からの情報取得/蓄積
- OSリソースの情報取得/蓄積

- その他機能

- 簡易レポート作成機能
- アラート機能
- ログ管理機能
- 自動メンテナンス機能

- その他各種情報

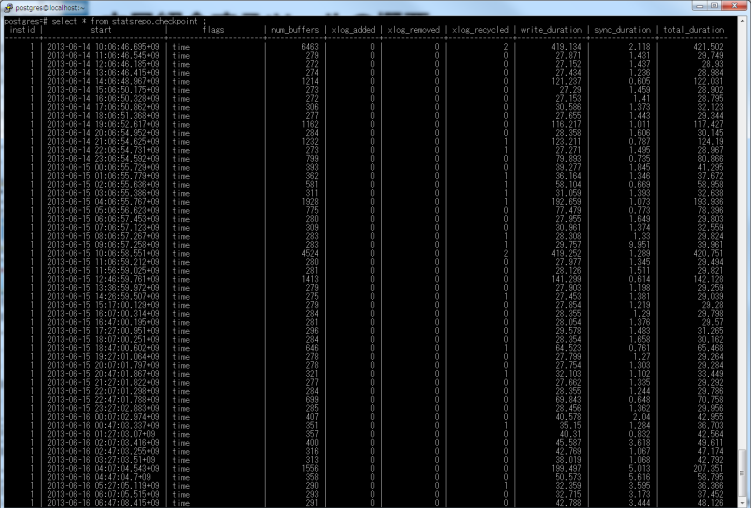
- BSDライセンス

- 最新バージョン 2.5.0

- http://pgfoundry.org/frs/?group_id=1000422
- PostgreSQL 9.3 にも対応

- 詳細マニュアルあり

- http://pgstatsinfo.projects.pgfoundry.org/pg_statsinfo-ja.html



relid	start	tags	run_buffers	xlog_added	xlog_removed	xlog_recycled	write_duration	sync_duration	total_duration
1	2013-08-14 10:08:46.595+09	time	683	0	0	2	419.134	2.118	421.502
1	2013-08-14 11:09:46.545+09	time	276	0	0	0	27.671	1.451	29.743
1	2013-08-14 12:09:46.195+09	time	174	0	0	0	37.192	1.457	38.93
1	2013-08-14 13:09:46.415+09	time	274	0	0	0	27.434	1.236	28.684
1	2013-08-14 14:09:46.850+09	time	124	0	0	0	141.097	0.605	122.011
1	2013-08-14 15:09:50.175+09	time	272	0	0	0	27.29	1.459	28.602
1	2013-08-14 16:09:50.290+09	time	272	0	0	0	27.165	1.441	28.795
1	2013-08-14 17:09:52.190+09	time	277	0	0	0	39.598	1.378	39.422
1	2013-08-14 18:09:52.190+09	time	1182	0	0	0	27.695	1.443	29.344
1	2013-08-14 19:09:52.190+09	time	294	0	0	0	116.711	1.011	117.427
1	2013-08-14 20:09:54.490+09	time	294	0	0	0	28.398	1.605	30.145
1	2013-08-14 21:09:54.781+09	time	1252	0	0	0	192.211	0.929	193.14
1	2013-08-14 22:09:54.781+09	time	273	0	0	1	27.271	1.455	28.967
1	2013-08-15 00:09:55.729+09	time	788	0	0	0	39.393	1.759	39.988
1	2013-08-15 01:09:55.729+09	time	393	0	0	0	39.277	1.445	41.295
1	2013-08-15 02:09:55.688+09	time	581	0	0	0	38.184	1.348	39.888
1	2013-08-15 03:09:55.688+09	time	311	0	0	1	38.104	0.668	38.668
1	2013-08-15 04:09:55.767+09	time	1928	0	0	1	11.693	1.328	12.838
1	2013-08-15 05:09:55.429+09	time	726	0	0	1	27.473	0.723	28.098
1	2013-08-15 06:09:57.460+09	time	238	0	0	0	27.695	1.449	29.013
1	2013-08-15 07:09:57.729+09	time	209	0	0	0	30.861	1.374	32.599
1	2013-08-15 08:09:57.367+09	time	253	0	0	0	28.398	1.623	29.824
1	2013-08-15 09:09:57.298+09	time	935	0	0	1	29.757	0.951	30.981
1	2013-08-15 10:09:58.361+09	time	4524	0	0	2	419.292	1.292	420.767
1	2013-08-15 11:09:58.212+09	time	280	0	0	0	27.877	1.345	29.484
1	2013-08-15 12:09:59.165+09	time	281	0	0	0	28.108	1.511	29.921
1	2013-08-15 12:46:59.761+09	time	1413	0	0	0	141.699	0.814	142.723
1	2013-08-15 13:09:59.772+09	time	729	0	0	0	27.893	1.158	29.299
1	2013-08-15 14:09:59.507+09	time	272	0	0	1	27.453	1.321	29.038
1	2013-08-15 15:09:59.190+09	time	279	0	0	0	27.695	1.129	29.299
1	2013-08-15 16:09:59.014+09	time	284	0	0	0	28.955	1.29	30.488
1	2013-08-15 17:09:59.290+09	time	298	0	0	0	28.694	1.483	31.295
1	2013-08-15 18:09:59.290+09	time	298	0	0	0	29.678	1.395	31.495
1	2013-08-15 19:09:59.290+09	time	298	0	0	0	29.824	0.953	30.163
1	2013-08-15 20:09:59.060+09	time	646	0	0	0	14.723	0.486	15.486
1	2013-08-15 21:09:59.060+09	time	278	0	0	0	27.289	1.227	29.284
1	2013-08-15 22:09:59.190+09	time	278	0	0	0	27.764	1.303	29.384
1	2013-08-15 23:09:59.190+09	time	321	0	0	1	32.103	1.102	33.449
1	2013-08-16 00:09:59.350+09	time	297	0	0	0	27.192	1.355	29.592
1	2013-08-16 01:09:59.190+09	time	294	0	0	0	28.955	1.244	30.788
1	2013-08-16 02:09:59.190+09	time	689	0	0	0	68.845	0.945	70.195
1	2013-08-16 03:09:59.190+09	time	295	0	0	0	28.458	1.362	29.655
1	2013-08-16 04:09:59.190+09	time	687	0	0	0	68.845	0.945	70.195
1	2013-08-16 05:09:59.190+09	time	291	0	0	1	28.115	1.284	29.703
1	2013-08-16 06:09:59.190+09	time	357	0	0	0	35.711	1.352	37.315
1	2013-08-16 07:09:59.190+09	time	400	0	0	0	40.877	0.618	40.611
1	2013-08-16 08:09:59.190+09	time	318	0	0	0	42.189	1.067	43.714
1	2013-08-16 09:09:59.190+09	time	311	0	0	0	38.018	0.868	40.702
1	2013-08-16 10:09:59.190+09	time	1506	0	0	0	189.497	0.813	207.551
1	2013-08-16 11:09:59.190+09	time	268	0	0	0	26.573	0.818	27.795
1	2013-08-16 12:09:59.190+09	time	290	0	0	1	32.959	1.355	34.398
1	2013-08-16 13:09:59.190+09	time	293	0	0	0	27.715	1.423	29.482
1	2013-08-16 14:09:59.190+09	time	291	0	0	0	42.788	3.444	45.128

取得/蓄積したDB統計情報

pg_statsinfo のアーキテクチャ

• プログラミング言語

- C言語



• 動作方法

- PostgreSQL の shared_preload_library 経由で起動します
- postgresql.conf に設定を記述し、通常起動すれば動作します

• システム構成

- 統計情報を取得する監視対象DBにpg_statsinfoをインストール
 - 統計情報を蓄積するリポジトリDBにはインストール不要
 - 監視対象DBとリポジトリDBは同一でもOKです



pg_statsinfo 機能紹介 1/5

• 取得する統計情報

- PostgreSQLの統計情報コレクタが収集する全ての情報
 - 統計情報コレクタの詳細については以下を参照してください☺
 - <http://www.postgresql.jp/document/9.2/html/monitoring-stats.html>
 - スナップショットとして一定間隔で取得します
 - デフォルトで10分おきに取得
- PostgreSQL ログのテキスト解析
 - ログにしか出力されない情報を抽出
 - チェックポイント情報のログ
 - VACUUM の実行情報ログ
- /proc に格納されるOSリソース情報の取得
 - 5秒間隔で情報を取得し、スナップショット時には平均値を格納
 - CPU使用状況情報 (idle, iowait, system, user, Load Average)
 - メモリ使用状況 (memfree, buffers, cached, swap, dirty)
 - ディスク使用状況 (IO size, IO time, ディスク使用状況)

pg_statsinfo 機能紹介 2/5

• 簡易レポート機能

- コマンドライン上から、テキスト形式でレポート結果を出力する
 - 利用例) コマンドライン上からレポートを見たい運用者/開発者向け
- pg_stats_reporterから見れるレポートは、ほぼ網羅

コマンド例: 10月1日から現在までの、すべてのインスタスの、すべてのレポート項目を作成する

```
$ pg_statsinfo -U postgres -B 2013-10-01 -r ALL | less
```

```
mitsu-ko@localhost:~$ pg_statsinfo -U postgres -B 2013-10-01 -r ALL | less
/* Checkpoint Activity */
-----
Total Checkpoints      : 157
Checkpoints By Time   : 157
Checkpoints By XLOG   : 0
Written Buffers Average : 1526.752
Written Buffers Maximum : 6172.000
Write Duration Average : 154.121 sec
Write Duration Maximum : 615.188 sec

/* Autovacuum Activity */
-----

/* Vacuum Basic Statistics (Average) */
-----
Table      Count  Removed Rows  Remain Rows  Index Scans  Duration  Duration(Max)

/* Vacuum I/O Statistics (Average) */
-----
Table      Page Hit  Page Miss  Page Dirty  Read Rate  Write Rate

/* Analyze Statistics */
-----
Table      Duration(Total)  Duration(Avg)  Duration(Max)  Count

postgres_statsrepo_column_20131004  353.189 s      9.380 s      95.768 s      40
postgres_statsrepo_column_20131004  188.880 s      4.825 s      27.040 s      35
postgres_statsrepo_index_20131003   98.420 s      2.338 s      9.740 s      40
postgres_statsrepo_index_20131003   89.580 s      1.783 s      7.520 s      40
postgres_statsrepo_index_20131004   45.360 s      1.312 s      5.820 s      35
postgres_statsrepo_table_20131004   31.850 s      0.804 s      4.610 s      35
postgres_pg_catalog_pg_stats        0.100 s      0.100 s      0.100 s      1
postgres_pg_catalog_pg_trigger      0.070 s      0.070 s      0.070 s      1
postgres_pg_catalog_pg_inherits     0.010 s      0.010 s      0.010 s      1

/* Query Activity */
-----

/* Functions */
-----
OID      Database  Schema      Function      Calls  Total Time  Self Time  Time/Call

71589    postgres  statsrepo  partition_inser 5624010 644598.131 ms 644598.131 ms 0.114 ms
71584    postgres  statsrepo  aet_snap_date  5624010 294729.941 ms 294729.941 ms 0.052 ms
71545    postgres  statsrepo  aet_acl_fendency_report 11 97261.179 ms 30402.150 ms 7532.889 ms
71517    postgres  statsrepo  tpa 7848828 51815.328 ms 51815.328 ms 0.007 ms
71640    postgres  statsrepo  alert_barbage 472 24448.014 ms 22376.271 ms 51.792 ms
71599    postgres  statsrepo  create_partition 472 2698.052 ms 311.319 ms 5.059 ms
71518    postgres  statsrepo  div 575718 2170.843 ms 2170.349 ms 0.004 ms
71599    postgres  statsrepo  aet_xlog_fendency 1418 2078.718 ms 2075.019 ms 1.461 ms
71599    postgres  statsrepo  aet_xlog_fendency 2 1159.008 ms 418.884 ms 579.504 ms
71639    postgres  statsrepo  alert_wact 472 932.840 ms 776.050 ms 1.707 ms
71599    postgres  statsrepo  devicestats 472 795.377 ms 767.259 ms 1.605 ms
71554    postgres  statsrepo  aet_xlog_stats 1 764.070 ms 1.075 ms 764.070 ms
71641    postgres  statsrepo  alert_query 472 738.522 ms 738.525 ms 1.560 ms
71621    postgres  statsrepo  xlog_location_diff 2084 716.058 ms 102.326 ms 0.342 ms
```

```
mitsu-ko@localhost:~$ pg_statsinfo -U postgres -B 2013-10-01 -r ALL | less
/* Heavily Accessed Tables */
-----
Database  Schema  Table      Seq Scans  Read Rows  Read Rows/Scan  Cache Hit  Ratio(%)

postgres  statsrepo  loadsave  2 120280 60145.000 88.200
postgres  statsrepo  cpu 1 60123 60123.000 89.500
postgres  statsrepo  snapshot 40 2401052 60026.500 100.000
postgres  statsrepo  autoanalyze 2 68830 24315.000 57.700
postgres  statsrepo  xlog 4 120290 30072.500 76.600
postgres  statsrepo  checkpoint 2 34341 17170.500 46.400
postgres  statsrepo  autovacuum 2 32708 16354.000 0.500
postgres  statsrepo  column_20131004 6 90183 15030.500 88.300
postgres  statsrepo  column_20131003 6 85226 1421.000 88.300
postgres  statsrepo  table_20131004 13 21192 1630.895 89.300
postgres  statsrepo  table_20131003 13 21114 1624.154 87.300
postgres  statsrepo  index_20131004 8 763 1368.500 87.300
postgres  statsrepo  index_20131003 8 7812 1392.000 85.900
postgres  statsrepo  instance 2205 324411 147.125 89.300
postgres  statsrepo  alert 472 69608 139.000 89.200

/* Low Density Tables */
-----
Database  Schema  Table      Live Tuples  Logical Pages  Physical Pages  Logical Page Ratio(%)

linux     public  ixr_status  0 0 0 1938
avro      public  ixr_usage  0 0 0 1642
cassandra public  ixr_indexes 0 0 0 5925
cassandra public  ixr_usage  0 0 0 45989
cdh2_cloudera-deskto public  ixr_usage  0 0 0 6072
cdh2_hadoop public  ixr_indexes 0 0 0 2980
cdh2_hadoop public  ixr_usage  0 0 0 28123
cdh2_hive public  ixr_usage  0 0 0 3758
cdh2_ozie public  ixr_usage  0 0 0 5115
cdh2_flume public  ixr_usage  0 0 0 1974

/* Fragmented Tables */
-----
Database  Schema  Table      Column      Correlation

avro      public  ixr_files  filename  -0.623
avro      public  ixr_files  filename  -0.623
cdh2_flume public  ixr_files  filename  -0.576
cdh2_flume public  ixr_files  filename  -0.576
linux     public  ixr_releases  releaseid  -0.528
hadoop   public  ixr_files  filename  -0.440
hadoop   public  ixr_files  filename  -0.440
cdh2_hadoop public  ixr_declarations  ansid  -0.218
cdh2_sooop public  ixr_declarations  ansid  -0.218
cdh2_ozie public  ixr_declarations  ansid  -0.218
jlibatus public  ixr_declarations  ansid  -0.218
cdh2_hive public  ixr_declarations  ansid  -0.218
cdh2_hive public  ixr_declarations  ansid  -0.218
knotcabinet public  ixr_declarations  ansid  -0.218
outout     public  ixr_declarations  ansid  -0.218
membase   public  ixr_declarations  ansid  -0.218
emacached public  ixr_declarations  ansid  -0.218
```


pg_statsinfo 機能紹介 3/5

• 自動メンテナンス機能

- リポジトリDBに蓄積した統計情報を自動削除する機能
 - pg_statsinfo は、1日間隔で自動パーティショニングを行ってデータを格納しているため、削除対象のデータはパーティションを丸ごと TRUNCATE される
 - 高速で低負荷なデータ削除を実現

• TIPS

- 削除対象の期間は、監視インスタンスの設定で最も期間の近いものが優先されて実行されるので注意



pg_statsinfo 機能紹介 4/5

• ログ管理機能

- PostgreSQLのログを扱いやすく管理する
- 2段階ログ出力機能
 - PostgreSQLログの出力レベルから、さらにpg_statsinfoのログレベルを設定可能
 - 利用例) PostgreSQLのログは詳細レベルに設定し、普段見るログは見やすいログレベルに設定する
 - ログファイル名を固定できるので、監視ミドルウェアのログ監視にも使えます
- 複数ログ出力機能
 - syslog と pg_log に同時に書き込む
- ログレベル変更機能
 - 特定のログに対して、ログレベルを変更して出力する機能
 - 例) ログレベルINFOの特定のログ出力をLOGに変更
- ログ圧縮機能
 - 1日経過してログローテートされたログに対して、圧縮を実行



pg_statsinfo 機能紹介 5/5

・アラート関数

- DB統計情報の値が想定値を超えた場合に、ログに出力する機能
 - 利用例) 出力されたログを監視ミドルウェアで監視する
 - アラート関数は、スナップショット契機で実行されます
- デフォルトで以下の値が設定されているので適宜変更すること
 - 変更方法は、statsrepo.alert テーブルの各値をUPDATE文で変更する

アラート設定テーブル

カラム名	デフォルト値	説明
instid	-	監視対象インスタンスID
rollback_tps	100	秒間のロールバック数
commit_tps	1000	秒間のコミット数
garbage_size	20000	監視インスタンス中の不要領域のサイズ(MB)
garbage_percent	30	監視インスタンスに占める不要領域の割合(%)
garbage_percent_table	30	各テーブルに占める不要領域の割合(%)
response_avg	10	クエリの平均レスポンス時間(秒)
response_worst	60	クエリの最長レスポンス時間(秒)
enable_alert	true	アラート対象判定フラグ

pg_statsinfo のインストール方法

1. RPMのインストール

```
$ su  
# rpm -ivh pg_statsinfo-2.4.0-1.pg92.rhel6.x86_64.rpm
```

2. postgresql.conf に以下の内容を追加

```
#最小設定  
shared_preload_libraries = 'pg_statsinfo'           # 事前ロードを行う  
log_filename = 'postgresql-%Y-%m-%d_%H%M%S.log' # ログファイル名を指定する
```

3. PostgreSQLを起動

```
$ pg_ctl -D data start
```

4. 以下のログが出力されればインストール成功！

```
server starting  
LOG: loaded library "pg_statsinfo"  
LOG: pg_statsinfo launcher started  
LOG: start  
LOG: installing schema: statsinfo  
LOG: installing schema: statsrepo_partition
```

インストール手順は**Web**マニュアルにも記載されています☺
http://pgstatsinfo.projects.pgfoundry.org/pg_statsinfo-ja.html#install

pg_statsinfo の TIPS

- **1 回のスナップショットのサイズは、約300KB ~ 800KB**
 - スナップショット取得間隔を調整して、ディスクフルにならないように注意する
- **統計情報取得の際の性能劣化は、ほぼありません**
 - DBT-2 ベンチマークで 2 %程度の劣化を確認
 - ただし、レポジトリDBを分けた構成での話です
- **遠隔のレポジトリDBを使用したい場合は、postgresql.conf に pg_statsinfo.repository_server を設定する**
 - 設定しない場合は、'host=localhost port=5432'が設定されます
- **レポジトリDBにパスワードを使用している場合は、/var/lib/pgsql/.pgpass にパスワードを設定する**

pg_stats_reporter の概要

• pg_statsinfo で取得/蓄積した統計情報の可視化ツール

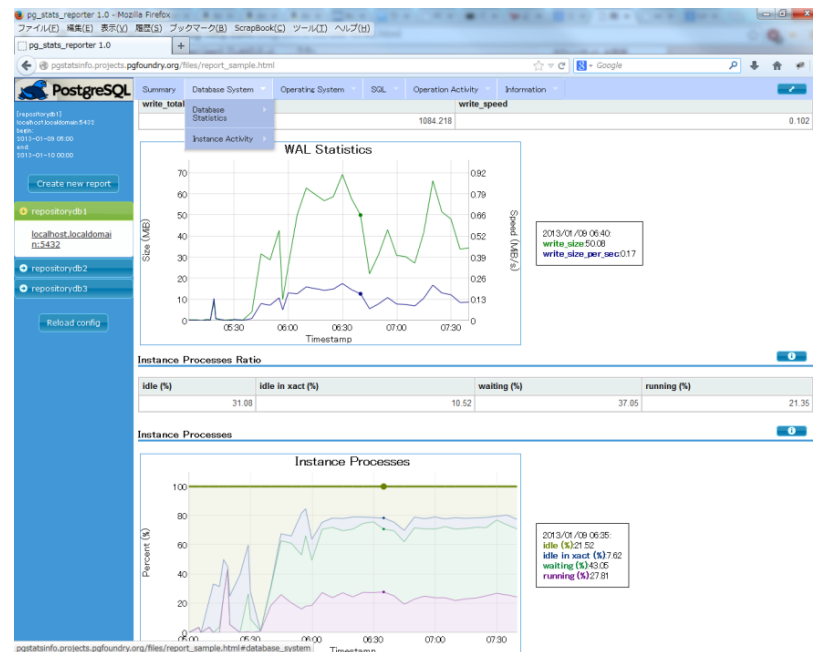
- 作成できるレポート例
- トランザクション状況
- DBサイズの推移
- OSリソース状況
- WALの出力量
- レプリケーション状況
- デッドロック情報

- pg_reporterの後継ツール
- pg_reporterで作成できるレポート

+ α

• その他各種情報

- BSDライセンス
- 最新バージョン 2.0.0
- http://pgfoundry.org/frs/?group_id=1000422
- 詳細マニュアルあり
- http://pgstatsinfo.projects.pgfoundry.org/pg_stats_reporter-ja.html



pg_stats_reporterの画面

pg_stats_reporter のアーキテクチャ

• ソフトウェア構成

- Apache + PHP + PostgreSQL
 - PHP + PostgreSQL のみでも動作可能
 - PostgreSQL 8.3 以降で動作



PostgreSQL

• プログラミング言語

- PHP + javascript + SQL



• 利用ライブラリ

- PHP フレームワーク
 - Smarty
- ユーザインタフェース
 - jQuery, jQuery UI, tablesorter, Superfish
- グラフ作成
 - dygraphs, jqPlot



dygraphs



レポートの作成方法 1/2

- Web ブラウザからレポート作成する方法
 - 数クリックで簡単にレポート作成

The screenshot shows the pg_stats_reporter web interface. Three callouts are present:

- ① レポート作成対象のDBを選択**: Points to the left sidebar where databases (repositorydb1, repositorydb2, repositorydb3) are listed.
- ② レポート作成ボタンを押す**: Points to the "Create new report" button in the sidebar.
- ③ レポート作成の対象期間を設定**: Points to the "Create new report" dialog box, which includes a date range selector (begin: 2013-10-03 00:00, end: 2013-10-10 15:30) and a calendar widget for selecting the date.

The main interface displays a "Summary" section with the following data:

instname	5831402967448641535
hostname	localhost.localdomain
port	5432
pg_version	9.2.2
snap_begin	2013-01-09 05:00:00
snap_end	2013-01-09 07:40:00
duration	02:40:00
total_dbsize	72 MiB
total_commits	
total_rollback	

Below the summary is a "Database System" section with "Database Statistics" and "Transaction Statistics" tables. The "Database Statistics" table shows metrics for 'bench' and 'postgres' databases. The "Transaction Statistics" section includes a line graph showing "Transaction per second (xact/s)" over time (Timestamp) from 05:30 to 07:30. The graph shows four data series: bench commit_tps, bench rollback_tps, postgres commit_tps, and postgres rollback_tps.

レポートの作成方法 2/2

• コマンドラインからレポートを作成する方法

- PHP のスタンドアローンモードで動作します
- 利用シーン
 - コマンドラインから、レポートを作成したい場合
 - cron 等で定期的にレポートを作成/蓄積したい場合
- コマンドラインのみの利用の場合 Apache のインストール不要
 - システム要件等で、Apache をインストールできない場合に
- レポジットリDBのデータは定期的に消したいが、レポート結果は長期間保持しておきたい場合

コマンド例: 10月1日から10月12日までのレポートを repot_dir に作成する

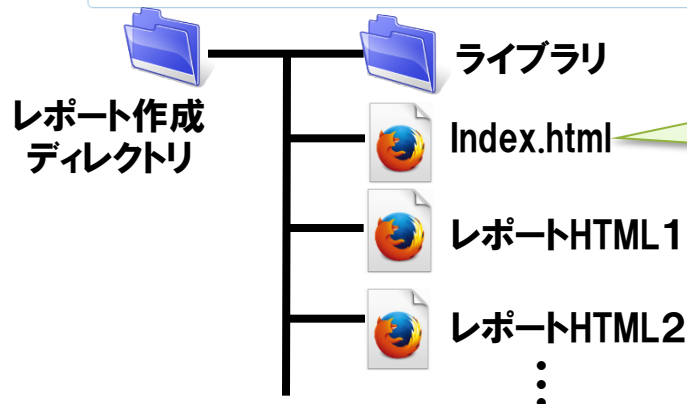
```
$ pg_stats_reporter -B 2013-10-01 -E 2013-10-08 -O report_dir  
[LOG] Report file created: sample_localhost_5432_1_20131008-1419_20131008-1945.html
```

レポートの作成方法 2/2

• レポートインデックス機能

- コマンドラインモードで、レポート作成の際に指定したディレクトリ配下に、レポートのインデックスが自動作成されます
- 過去のレポート一覧を見る際に便利です

InstID	Host:Port	begin	end	term	report file
1	localhost:5432	2013-10-08 17:00	2013-10-08 18:00	0 days 01:00	sample_localhost_5432_1_20131008-1419_20131008-1735.html
2	localhost:5432	2013-10-08 17:00	2013-10-08 18:00	0 days 01:00	sample_localhost_5432_1_20131008-1419_20131008-1730.html
3	localhost:5432	2013-10-08 17:00	2013-10-08 18:00	0 days 01:00	sample_localhost_5432_1_20131008-1419_20131008-1735.html
1	localhost:5432	2013-10-08 16:00	2013-10-08 17:00	0 days 01:00	sample_localhost_5432_1_20131008-1419_20131008-1730.html
2	localhost:5432	2013-10-08 16:00	2013-10-08 17:00	0 days 01:00	sample_localhost_5432_1_20131008-1419_20131008-1735.html
3	localhost:5432	2013-10-08 16:00	2013-10-08 17:00	0 days 01:00	sample_localhost_5432_1_20131008-1419_20131008-1730.html



過去に作成した
レポート群

レポート
インデックス

pg_stats_reporter のインストール方法

1. RPM のインストール

```
$ su
# rpm -ivh httpd-2.2.15-15.el6_2.1.x86_64.rpm ￥￥
php-5.3.3-3.el6_2.8.x86_64.rpm ￥￥
php-common-5.3.3-3.el6_2.8.x86_64.rpm ￥￥
php-pgsql-5.3.3-3.el6_2.8.x86_64.rpm ￥￥
php-intl-5.3.3-3.el6_2.8.x86_64.rpm ￥￥
pg_stats_reporter-1.0.0-1.el6.noarch.rpm
```

2. pg_stats_reporter.ini を書き換え（デフォルトで以下の内容になっています）

```
# vim /etc/pg_stats_reporter.ini
----- リポトリDBの接続設定を記述 -----
host = localhost
port = 5432
dbname = postgres
username = postgres
password =
```

3. Apache を起動

```
# service httpd start
```

4. Web ブラウザから、以下の URL にアクセス

```
http://localhost/pg_stats_reporter/pg_stats_reporter.php
```

インストール手順は **Web** マニュアルにも記載されています☺

http://pgstatsinfo.projects.pgfoundry.org/pg_stats_reporter-ja.html#install

pg_stats_reporter の TIPS

- Android と iPad でも、レポートの閲覧ができます
- jQuery UI ベースで IF をデザインしているため、IF の配色変更が容易にできます
 - 画面左上のロゴ画像も、ファイル置換で変更可能です
- **セキュリティ対策**
 - Apache ベースなので、.htaccess が使えます
 - その他 Apache のセキュリティ対策がそのまま応用できます
- **表示するレポート項目の絞り込みができます**
 - /etc/pg_stats_reporter.ini に設定を行うことで、運用に必要なレポート項目のみを表示できます

DBT-2 とは？

- **Open Source Development Labs(OSDL)が開発したTPC-Cの実装**

- 部品の卸売り業者のデータベース操作のシミュレート
- 製品やサービスの管理、販売、配送を行う業務モデル
 - <http://www.tpc.org/tpcc/>
- 一定時間内に応答が返ったものをスコアとしてカウントする
- I/Oネックのベンチマーク

- **主なパラメータ**

- warehouse
 - 倉庫サイズ
 - 1パラメータ毎に10万レコードが追加される
 - 主にデータベースサイズの調整に用いる
- TPW
 - Transaction per warehouse
 - 倉庫サイズに応じたクライアント数が準備される。デフォルト値10
 - TPWをデフォルト値より大きくし、かつ warehouse がメモリサイズ以下の場合にはCPUネックのベンチマークになる

DBT-2 のトランザクション傾向

• 主なボトルネック

- ランダムリード/ライト
 - 大半のSQLがインデックス スキャン
- ランダムIO性能とキャッシュ性能が結果を大きく左右
- 同時実行性能も重要
 - 同時に多くのクライアントを捌けると有利

• その他特徴

- SQLのプランは比較的単純
- ベンチマークの理論値が存在
 - 制限時間内にすべての応答を捌けたら理論値になる
- 性能限界のDBサイズはサーバ搭載メモリの約2倍程度
- WALの出力量はpgbenchより少なめ

検証に用いたマシンと設定

サーバ	HP DL360 G7
CPU	Xeon E5640 2.66GHz (1P/4C)
Memory	DDR3-10600R-9 18GB
RAID card	P410i / 256MB cache
Disk	4 x 146GB (1.5krpm) RAID 1 + 0

主な postgresql.conf 設定

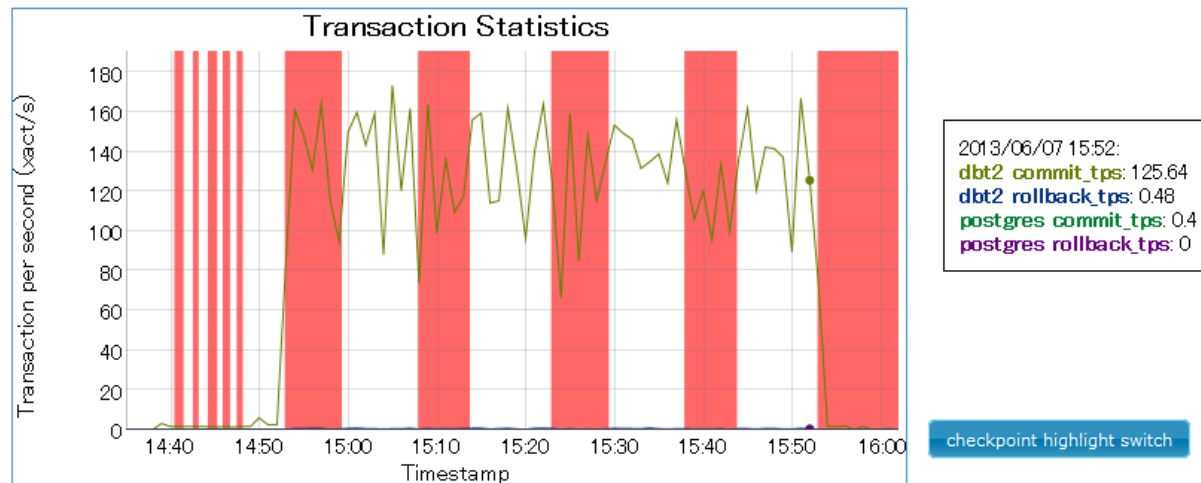
```
max_connections = 300
shared_buffers = 2458MB
work_mem = 1MB
maintenance_work_mem = 64MB
fsync = on
wal_sync_method = fdatasync
full_page_writes = on
wal_buffers = -1
archive_mode = on
```

```
checkpoint_segments = 300
checkpoint_timeout = 15min
checkpoint_completion_target = 0.7
random_page_cost = 2.0
effective_cache_size = 9GB
default_statistics_target = 10
log_destination = 'syslog'
autovacuum = on
```

pg_stats_reporter で可視化した結果 1/5

Transaction Statistics

i



• トランザクション量の変化

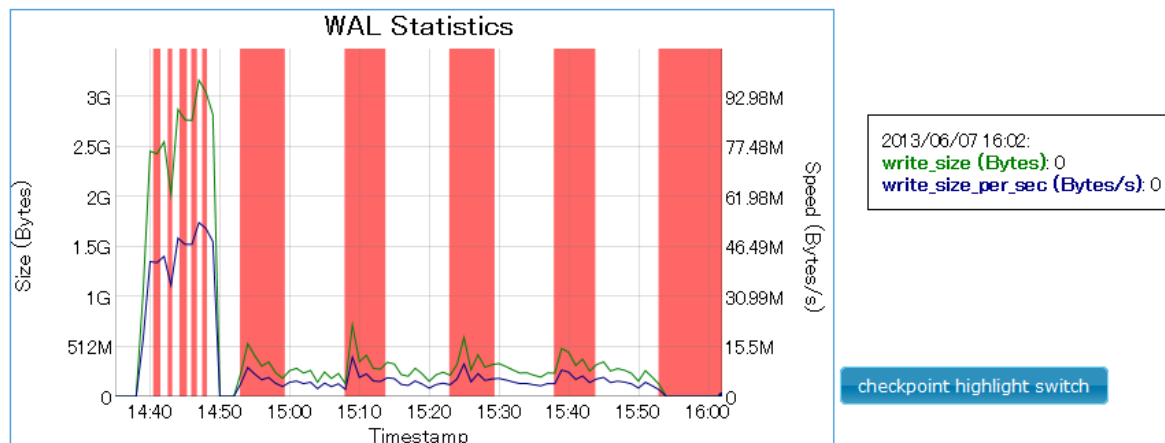
- 一定の間隔で接続数のムラが発生する or PostgreSQL が捌けるトランザクション数にばらつきがある
- CHECKPOINT 実行時は性能が低下
- CHECKPOINT は、時間契機で発生
 - checkpoint_timeout = 15min, checkpoint_segments = 300に設定

pg_stats_reporter で可視化した結果 2/5

Instance Activity

WAL Statistics

write_total	write_speed
46423.542	8.792



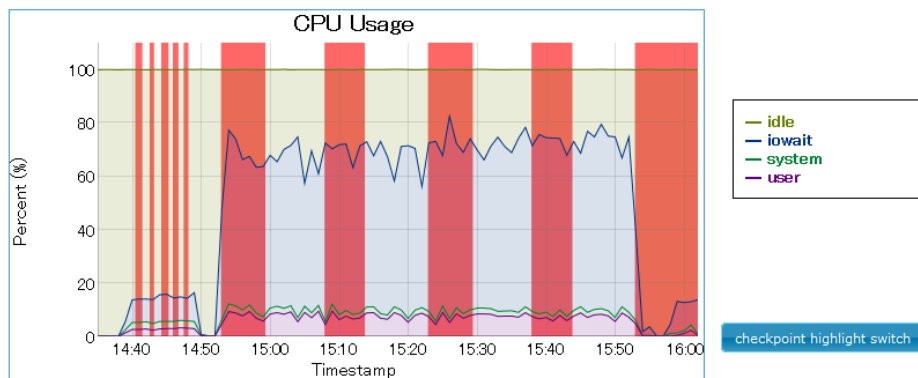
• WALの出力量

- データロードから、ベンチマーク終了まで4.6GBのWALを出力
- ロード時に最大のWAL出力量（54MB/s）が発生
- ベンチマーク時の最大WAL出力量は 12MB/s
- CHECKPOINT開始時は、full page writeの影響でWAL量が増える

pg_stats_reporter で可視化した結果 3/5

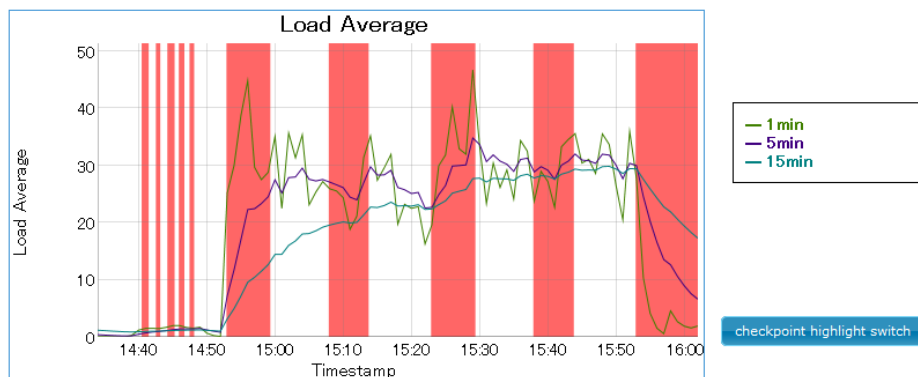
CPU Usage

6



Load Average

6



• CPU使用率

- iowaitが支配的、idleも多い
- CHECKPOINT最後の処理での Load Average が高い
 - リレーションファイルの fsync() 連打が原因

pg_stats_reporter で可視化した結果 4/5

SQL

Notable Tables

Heavily Updated Tables

database	schema	table	INSERT	UPDATE	DELETE	total	HOT (%)
dbt2	public	order_line	2224269	1804324	0	4028593	88.400
dbt2	public	stock	0	2224269	0	2224269	99.900
dbt2	public	orders	212147	189860	0	402007	99.900
dbt2	public	new_order	212147	0	189674	401821	0.000
dbt2	public	customer	0	391546	0	391546	99.900
dbt2	public	district	0	218071	0	218071	98.400
dbt2	public	history	201685	0	0	201685	0.000
dbt2	public	warehouse	0	201685	0	201685	99.400
dbt2	public	item	0	0	0	0	0.000

1/1 10

Heavily Accessed Tables

database	schema	table	seq_scan	seq_tup_read	tup_per_seq	hit (%)
dbt2	public	order_line	1	99315562	99315562.000	85.600
dbt2	public	orders	1	10417143	10417143.000	86.000
dbt2	public	history	1	10401739	10401739.000	65.400
dbt2	public	customer	1	10375737	10375737.000	63.400
dbt2	public	new_order	1	3191401	3191401.000	97.700
dbt2	public	item	1	100000	100000.000	99.900
dbt2	public	district	1	3400	3400.000	100.000

1/1 10

• テーブルの更新状況

- 全体的にHOTが上手く動作している
- order_line テーブルと stock テーブルにアクセスが多い
- キャッシュはヒット率が高いが・・・（怪しい）

pg_stats_reporter で可視化した結果 5/5

Long Transactions				
pid	client address	when to start	duration (sec)	query
15200	172.20.144.58	2013-06-07 14:38:38	621.675	VACUUM FREEZE VERBOSE ANALYZE;
4562	172.20.144.58	2013-06-07 15:57:43	106.851	SELECT now(),* FROM pgstattuple('customer');
32342		2013-06-07 15:38:26	93.753	autovacuum: ANALYZE public.new_order
4643	172.20.144.58	2013-06-07 16:00:34	85.920	SELECT now(),* FROM pgstattuple('stock');
4619	172.20.144.58	2013-06-07 15:59:36	54.508	SELECT now(),* FROM pgstattuple('order_line');
19966	172.20.144.58	2013-06-07 15:03:34	10.968	SELECT count(distinct s_i_id) FROM order_line, stock, district WHERE d_id = 4 AND d_w_id = 28 AND d_id = ol_d_id AND d_w_id = ol_w_id AND ol_i_id = s_i_id AND ol_w_id = s_w_id AND s_quantity < 10 AND ol_o_id BETWEEN 2987 AND 3006
19879	172.20.144.58	2013-06-07 15:01:50	10.091	COMMIT
20256	172.20.144.58	2013-06-07 15:09:30	10.024	SELECT count(distinct s_i_id) FROM order_line, stock, district WHERE d_id = 1 AND d_w_id = 225 AND d_id = ol_d_id AND d_w_id = ol_w_id AND ol_i_id = s_i_id AND ol_w_id = s_w_id AND s_quantity < 16 AND ol_o_id BETWEEN 2992 AND 3011
19877	172.20.144.58	2013-06-07 15:05:41	9.457	COMMIT
20251	172.20.144.58	2013-06-07 15:10:01	9.176	SELECT count(distinct s_i_id) FROM order_line, stock, district WHERE d_id = 9 AND d_w_id = 299 AND d_id = ol_d_id AND d_w_id = ol_w_id AND ol_i_id = s_i_id AND ol_w_id = s_w_id AND s_quantity < 11 AND ol_o_id BETWEEN 2995 AND 3014
19832	172.20.144.58	2013-06-07 15:16:01	9.017	COMMIT
20208	172.20.144.58	2013-06-07 15:07:07	8.266	SELECT count(distinct s_i_id) FROM order_line, stock, district WHERE d_id = 9 AND d_w_id = 205 AND d_id = ol_d_id AND d_w_id = ol_w_id AND ol_i_id = s_i_id AND ol_w_id = s_w_id AND s_quantity < 13 AND ol_o_id BETWEEN 2990 AND 3009
20209	172.20.144.58	2013-06-07 15:45:47	8.234	SELECT o_c_id FROM orders WHERE o_id = 2149 AND o_w_id = 6 AND o_d_id = 10
19830	172.20.144.58	2013-06-07 14:58:57	8.166	COMMIT
20170	172.20.144.58	2013-06-07 14:57:22	8.106	COMMIT
20078	172.20.144.58	2013-06-07 15:19:42	7.839	COMMIT
19921	172.20.144.58	2013-06-07 15:40:32	7.615	COMMIT
19919	172.20.144.58	2013-06-07 14:55:17	7.588	COMMIT

クエリ状況

- 複雑な条件句を持つSELECTが遅い
- COMMITに時間がかかるケースも
 - COMMIT時のWAL書き込みに時間がかかっている？
 - CHECKPOINT最後のfsync()でDisk IOが詰まっているのが原因？

性能向上のための TIPS (おまけ)

- **アーカイブの copy コマンドに direct_cp を使う**
 - アーカイブのコピーの際に、不要なファイルキャッシュが大量に生成されるため、性能に大きく影響します
 - direct_cp を用いることで、ファイルキャッシュを汚さずにアーカイブのコピーが行えるため、性能が向上します
 - BSD ライセンスの OSS です
 - <http://directcp.projects.pgfoundry.org/index.html>
- **SSDを使う**
 - DB のボトルネックは主にランダムIO のため、SSD を使うことで性能が飛躍的に向上します。ランダムIO の多いテーブルのみをSSD 上のテーブルスペースに配置するのもいいでしょう
- **大きなRAIDキャッシュを持つRAIDカードを使う**
 - CHECKPOINT 時の リレーションファイルの fsync() が大きなボトルネックとなりがちです。それらを回避するためにRAID キャッシュは最大の物を用意することをお勧めします

まとめ

• pg_statsinfo

- PostgreSQL の統計情報を取得/蓄積するツール
 - BSD ライセンス
 - http://pgstatsinfo.projects.pgfoundry.org/pg_statsinfo-ja.html
- DB 運用に必要な統計情報は、ほぼ取得しています
 - 新たなレポートは SQL で作成できます

• pg_stats_reporter

- pg_statsinfo で取得/蓄積した統計情報の可視化ツール
 - BSD ライセンス
 - http://pgstatsinfo.projects.pgfoundry.org/pg_stats_reporter-ja.html
- jQuery ベースの先進的 IF
 - レポートインデックス機能も便利です
- PHP + javascript で作成しているので改造も容易
 - patch 投稿も容易です！