大規模監視サーバでのPostgreSQL

2020/2/12

株式会社SRA
運用 サービスマネージャ
菊池幸博

■はじめに

本日の内容

自己紹介
こんなことを話します
背景
年代ごとの監視規模
困っていたこと
対応したこと
気が付いたこと
まとめ
質疑応答
クロージング

■自己紹介

- Yukihiro Kikuchi / 菊池 幸博
- 株式会社　SRA　NETS1 IT3

Work

・某製造系の社内クラウド運営　サービスマネージャ

Pre Work

・某携帯音楽配信サイトのインフラ担当

・某メーカ系インフラ担当

・開発系SE

こんなことを話します

2015年某製造系会社のプライベートクラウドと機器の監視を担当した際、
当時はZabbix+MySQLという一般的な構成でした。

急速に拡大するインフラを監視するため、
お客様のご意見やご要望を取り入れながら2016年、2018年の
2段階で監視システムを見直ししてきました。
その中で気が付いたこと、感じたことを共有させて頂きます。

背景

- 急速な監視対象の拡大
  某製造業様のプライベートクラウドを一新し、
  ESXi145台に4,000台のVMを収容

- 東京以外のDCにDR環境を構築
  関西圏にDCを構築

- 全国の事業所 (今回は対象外)
  500か所の事業所とそれに付随するネットワーク機器

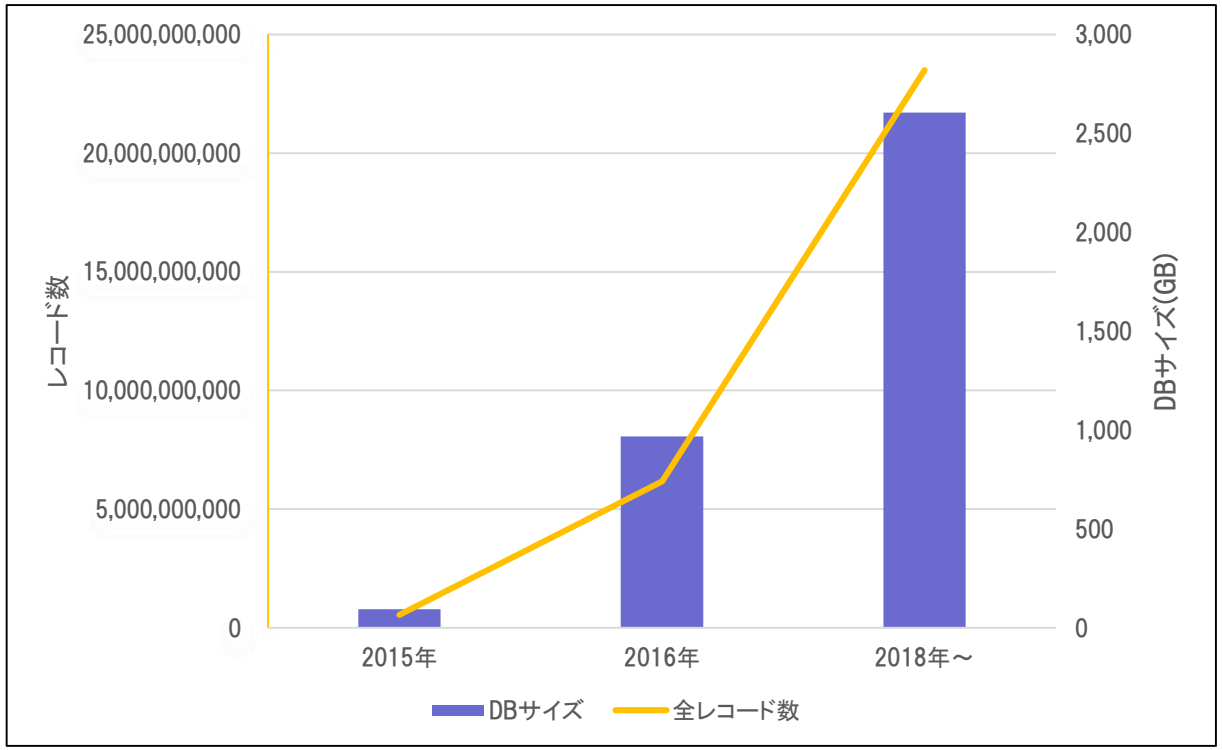
年代ごとの監視規模

年々増えていく監視対象に応じて、監視システムも必要なリソースが増えていきます。

	2015年	2016年	2018年～
ホスト数	672	1467	1711
アイテム数	65,615	77,593	317,989
トリガー数	8,213	18,477	67,430
1秒当たりの監視項目数	858.21	764.31	960.66
DBサイズ	96,256	991,232	2,667,110.4
全レコード数	553,369,496	6,169,518,645	23,492,504,222
DB種類	MySQL5.7.21	postgresql9.6	PostgreSQL10.4

年代ごとの監視規模

2018年の監視対象はレコード総数230億、2.56TB
2015年から比較するとレコード数42.45倍、DBサイズ27.71倍



困っていたこと

2015年当時、お客様より要望（苦情）が寄せられました。

・グラフが切れてしまう

データが取得できない時間帯があり、グラフが切れ切れになってしまう

・もっと長期間データを保管したい

年度単位でデータを見たいので最低1年半はデータを残したい

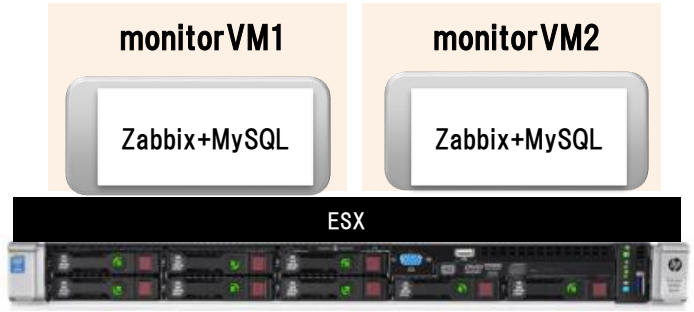
・レスポンスをよくして欲しい

グラフを表示すると「コーラを買いに行って立ち話しても表示されない」

2015年監視サーバの構成

Zabbix/DBサーバ（VM）
　プライベートクラウド上に以下のスペックで構築
　　CPU 16core
　　Mem 32G
　　OS　CentOS6.7
　　FS　ext4

構成：オールインワン（一般的なZabbixサーバの構成）を2台
　　zabbix proxyを別DCに置いて相互監視



S/W
　DB　MySQL-5.7
　監視サーバ　Zabbix-3.0

レスポンス悪化の原因

●DBのレスポンス悪化

①インストールしたままのmy.conf
②小さ過ぎるログファイルサイズ
③他の重いサーバと同じサーバ上で動いていた

●グラフが切れたりデータが取得できない

①zabbix_server.confもほとんど手つかずで、最適化されていなかった。
②DBレスポンスの悪化
（監視データ取得とDB書き込みからの戻りのタイミングずれ）

対応したこと

DBをPostgreSQLにするのは良いアイディアか？

もともとPostgreSQLに慣れており、運用もトラブルもほぼ対応できることから
DBをMySQLからPostgreSQLに変更を決定

以下のサイトがとても参考になりました。
「Zabbixのデータベース　ベンチマークレポート
PostgreSQL VS MySQL」
Yoshiharu Mori　SRAOSS Inc. Japan
https://www.sraoss.co.jp/event_seminar/2013/20130829_Zabbix_PG_vs_MySQL.pdf

「PostgreSQLとMySQLは同等の性能が出せる」とあり、テーブルパティショニングは
PostgreSQL9.6以降なら実現可能

監視サーバのレスポンス改善

目標

- ●応答速度向上

  「コーラを一口飲んだらもう表示されてる」

- ●運用の容易化

  ファイル容量の肥大化は予想できていたので、ファイルシステムの容量追加が誰でも容易に出来る事

- ●データを確実に保持する

  数年間データを保管するため、データが無くならないようにする

2016年監視サーバ

DBとZabbixのチューニングで問題は解決！

2016年時点で試みたこと

- PostgreSQLの採用
- テーブルパティショニングの採用
- PostgreSQL9.6の最適化
- zabbix_server.confの最適化
- zabbix proxyをaws上に設置（外形監視のため）

2016年監視サーバの構成

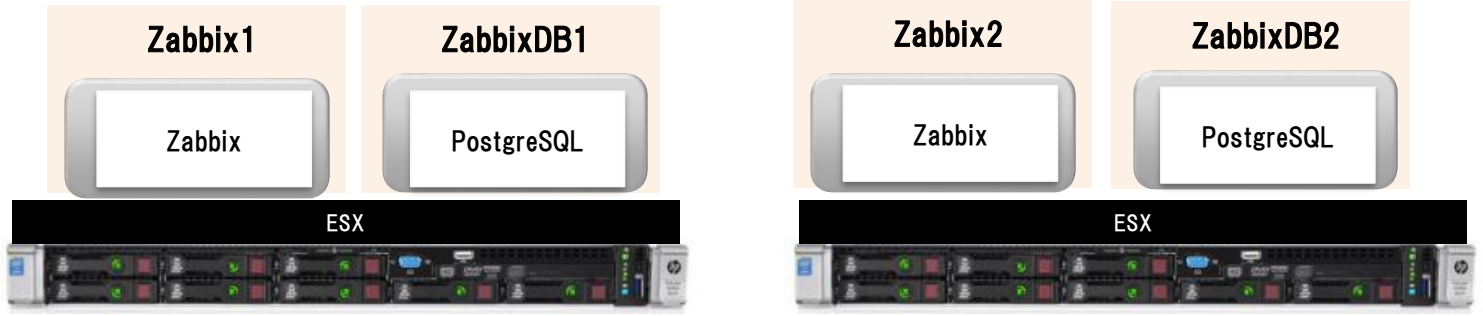
Zabbix ServerとDBサーバを分離
2つのセットをロケーションの異なるDCに配置

Zabbixサーバ

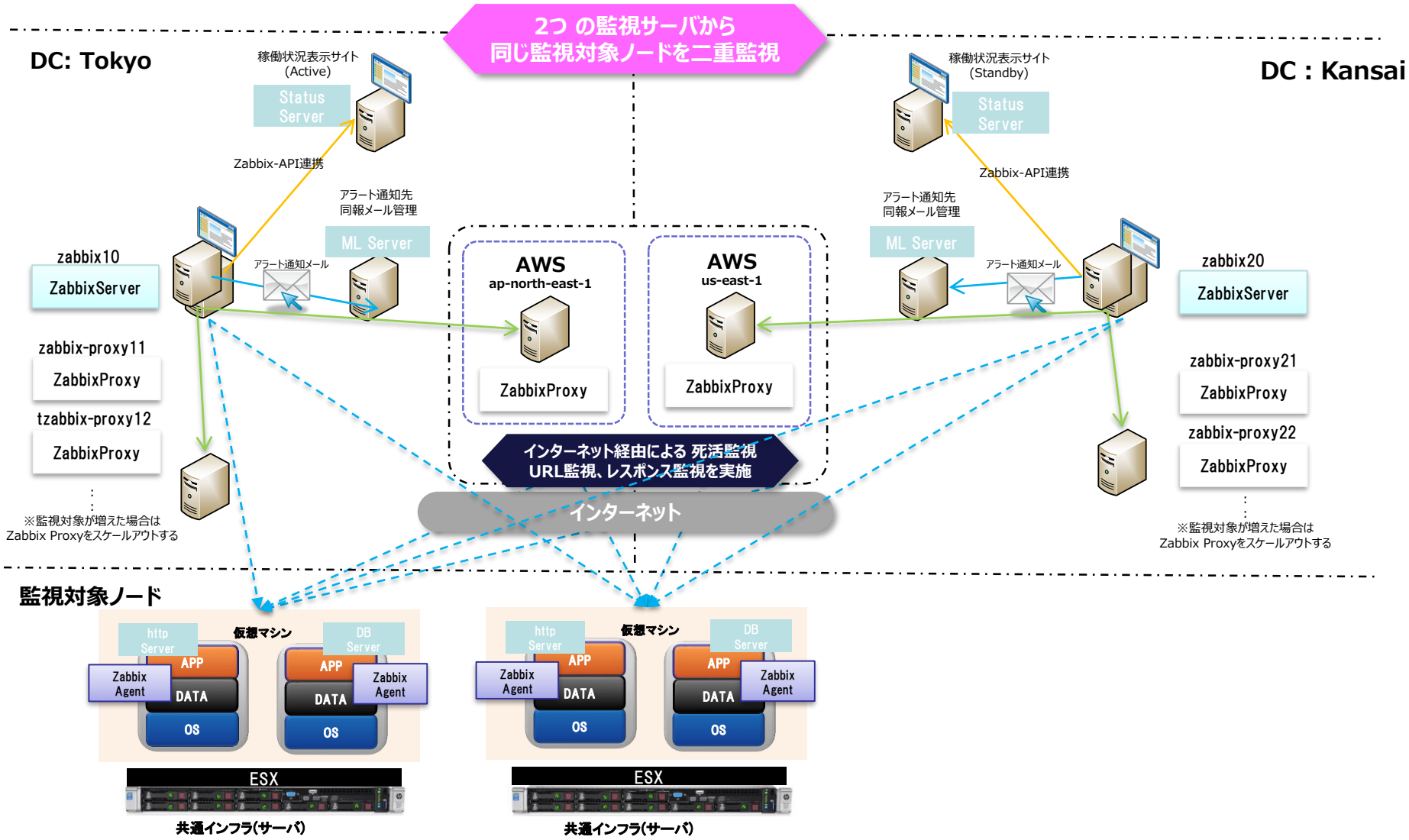
CPU 16core
Mem 64G
OS　CentOS7.x
FS　XFS　(容量50BG)
Zabbix Zabbix-3.4

DBサーバ

CPU 16core
Mem 128G
OS　CentOS7.x
FS　XFS　(容量500BGでスタート)
DB　PostgreSQL9.6



2016年監視システムの構成



更なる要件の追加

諸問題の解決で以下の要件が出てきて
監視システムを再設計することになりました。

- 監視項目の見直し
  適切なテンプレートの適用
- ドキュメント整備
  監視要件のまとめ、ポリシー策定
- 機器の追加で監視対象の増大
  新しくDC設備を増設するため、監視対象が大幅に増える
- 実測データ(history)の長期保存
  当初1年間のトレンドデータ保存が、グラフの細度を上げるため実測データを396日間保管

システムとして見直した点

・検索速度、グラフ表示速度の向上
さすがに一年運用後はDBサイズも大きくなり、速度の低下を感じるようになった。
→レプリケーションDBを参照用に振り分けるため、Pgpool-IIを採用する
　速度向上策としてパラレルクエリを有効化

・ファイルシステムの見直し
運用メンバーが誰でもファイルシステムの追加を行えるようにする。
XFS＋LVMより簡単にしたい
→ZFSを採用する（DBのデータ部分だけ。システムはXFS)

・バックアップ
ストレージの機能でバックアップを効率的に取得

ZFSの採用

・長期間のデータ保管はファイルシステムの増大が予想される
・FSとしても性能は良い方がいい

こんな記事を見つけました。
PostgreSQLを使うならZFS
https://blog.ohgaki.net/postgresql-should-use-zfs

大垣さんのブログにZFSが取り上げられていました。
ふむふむ、こんなに性能が良いなら使うしかない！

XFSとZFSの検証結果

参照のみ

条件：pgbench –c 100 –t 10 –N test

結果：zfsはxfsの7割程度の性能

ファイルシステム	1	2	3	4	5	平均
XFS	3733.642	3917.329	3959.8	3991.586	3648.117	3850.095
	3761.574	3947.876	3990.842	4023.083	3674.877	3879.65
ZFS	2516.522	2672.271	2607.609	2825.425	2560.638	2636.493
	2533.222	2690.459	2625.121	2845.9	2577.166	2654.374

更新系

条件：pgbench –c 100 –t 10 test

結果：zfsはxfsの1.8倍の性能

ファイルシステム	1	2	3	4	5	平均
XFS	858.7524	933.3922	953.4643	908.4245	920.6569	914.938
	860.2337	935.1084	955.2609	910.0683	922.351	916.6045
ZFS	1608.171	1737.952	1724.887	1706.248	1683.187	1692.089
	1614.84	1745.757	1732.57	1713.771	1690.529	1699.493

Pgpool-IIの検証結果

参照のみ

条件：pgbench –c 100 –t 10 –N test

結果：mondb100だと、最高で4倍、平均で2.86倍の処理

接続先	1	2	3	4	5	平均
MasterDB	3991.627	3647.101	3494.181	4055.452	4077.337	3853.14
SlaveDB	4323.788	4186.186	4341.588	4337.69	4347.679	4307.386
Pgpool-II	11033.02	7864.859	9706.875	12389.37	14185.15	11035.86

更新系

条件：pgbench –c 100 –t 10 test

結果：単体とほぼ変わらず（pgpoolのオーバーヘッドはない模様）

接続先	1	2	3	4	5	平均
MasterDB	1379.758	1319.785	1091.382	1050.13	853.1534	1138.842
SlaveDB	-	-	-	-	-	-
Pgpool-II	1427.882	1282.836	1174.457	1029.257	932.3296	1169.352

2018年監視サーバのS/W構成

PostgreSQL-10.4

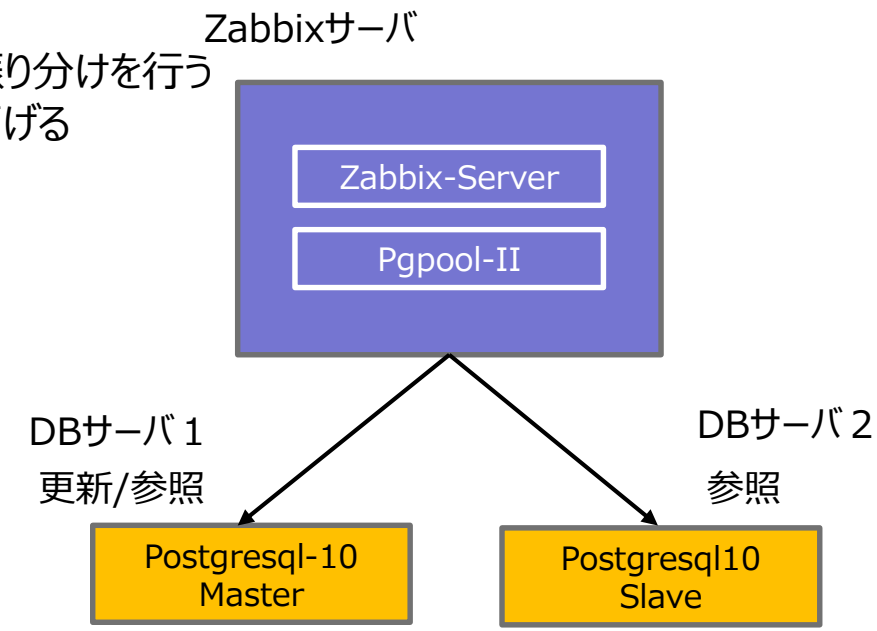
・streaming replicationで非同期にレプリケーションを構成している
・パラレルクエリを使い、複合型のクエリを並行に実行して実行時間を短縮している
・テーブルパーティショニングで巨大化するテーブルを効率良く利用できるようにする

Pgpool-II-3.6

・Zabbix-Serverから出力されるSQLを参照系と更新系でMaster/Slaveへ振り分けを行う
・PostgreSQLとのコネクションプーリングを行い、セッション接続・切断のコストを下げる

ZFS

・性能は予想より出ていないが、運用面で有利
・CPU負荷はXFSと変わらず
　性能追求というよりは運用面での有利さを採用



2018年監視サーバの構成

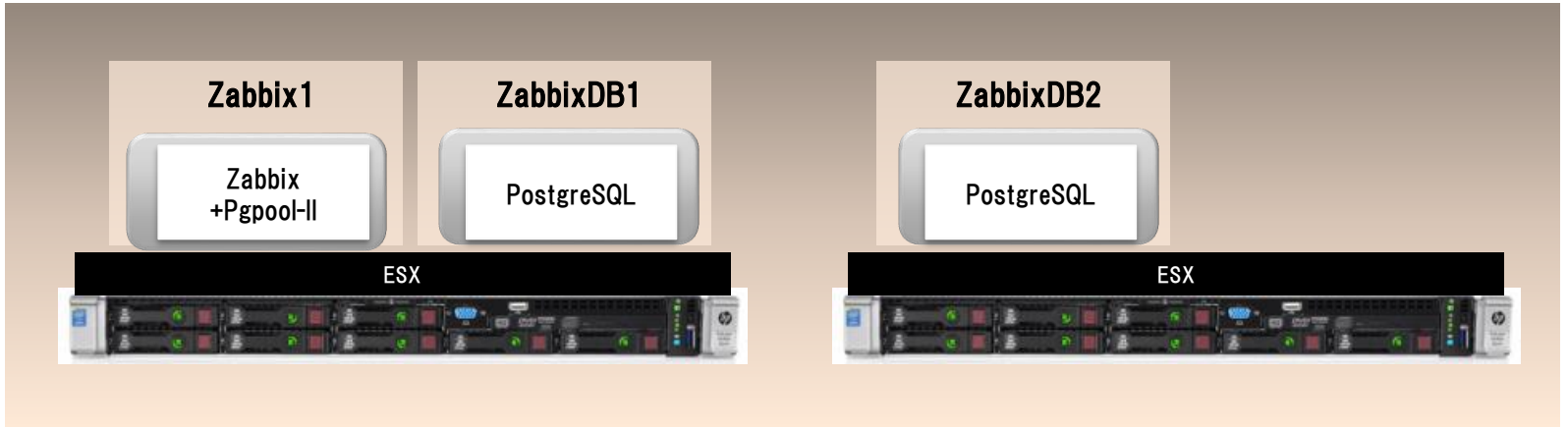
Zabbix ServerとDBサーバを分離
　　ロケーションの異なるところにVMを配置

Zabbixサーバ

CPU 16core
Mem 128GB
OS　CentOS7.x
FS　ZFS
Zabbix Zabbix-3.4

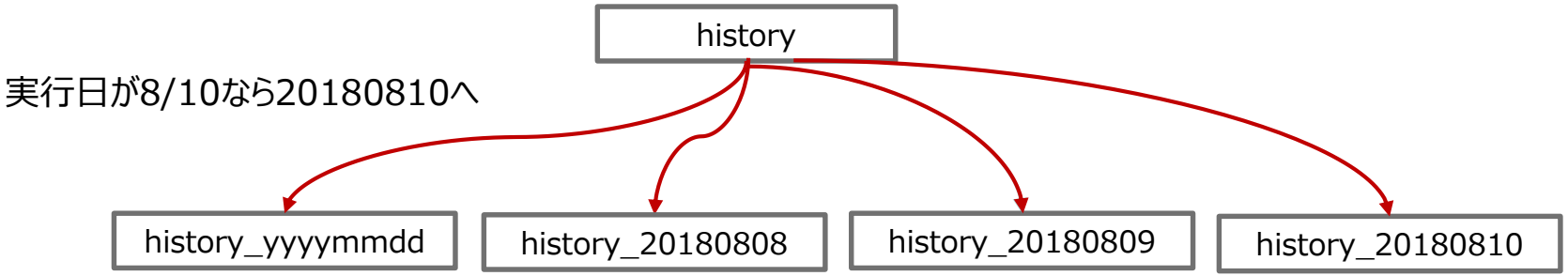
DBサーバ（レプリケーションも同様）

CPU 32core
Mem 128GB→256GB
OS　CentOS7.x
FS　ZFS　（容量500BGでスタート）
DB　PostgreSQL10.4



PostgreSQLのテーブルパーティショニング

MySQL、Oracleと同様に宣言型で実装可能となった(Ver10より)



アクセスはhistoryに対して行う。
historyを親テーブルとして、その下に日付が付いたテーブルを所属させる
親テーブルにはPrimary Keyやindexが付けられないため、
子テーブルにPrimary Key,indexを付与する
毎日2週間分の子テーブルを作成するが、「IF EXIST」で存在しない場合のみ作成する

<親テーブル定義>

```
CREATE TABLE public.history_uint (
    itemid bigint NOT NULL,
    clock integer DEFAULT 0 NOT NULL,
    value numeric(20,0) DEFAULT '0'::numeric NOT NULL,
    ns integer DEFAULT 0 NOT NULL
)
PARTITION BY RANGE (clock);


ALTER TABLE public.history_uint OWNER TO zabbix;
```

<パーティショニング定義>

子テーブル定義：

```
CREATE TABLE public.history_uint_20180625 PARTITION OF
public.history_uint
FOR VALUES FROM (1529852400) TO (1529938800);
ALTER TABLE public.history_uint_20180625 OWNER TO zabbix;
ALTER TABLE ONLY public.history_uint_20180625 ALTER COLUMN clock
SET DEFAULT 0;
ALTER TABLE ONLY public.history_uint_20180625 ALTER COLUMN ns
SET DEFAULT 0;
```

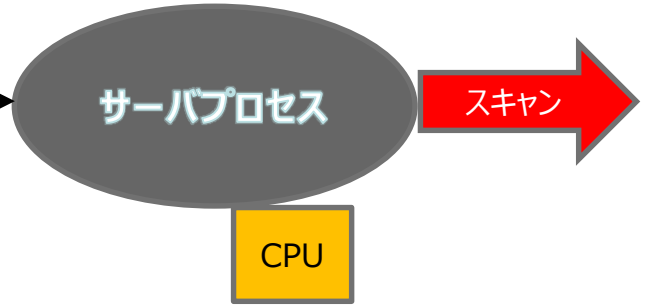
インデックス定義：

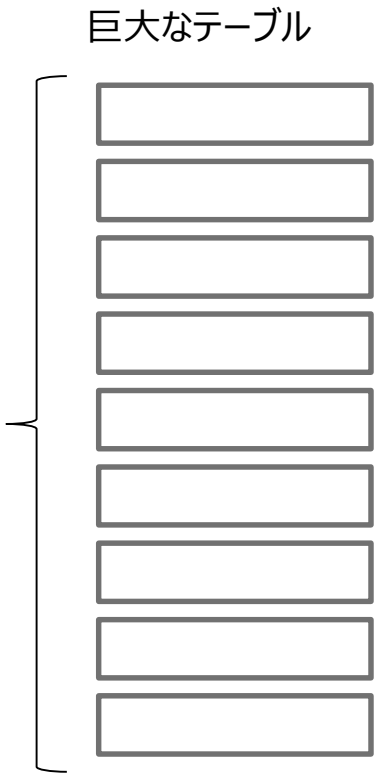
```
CREATE INDEX ihistory_uint_20180625 ON
public.history_uint_20180625 USING btree (itemid, clock);
```

パラレルクエリ

◆PostgreSQLの機能で9.6から強化された
クエリの応答をより速くするために、複数のCPUを活用するクエリプランを生成する。

従来の動作…
CPU一つが頑張ってテーブルを読んでいる

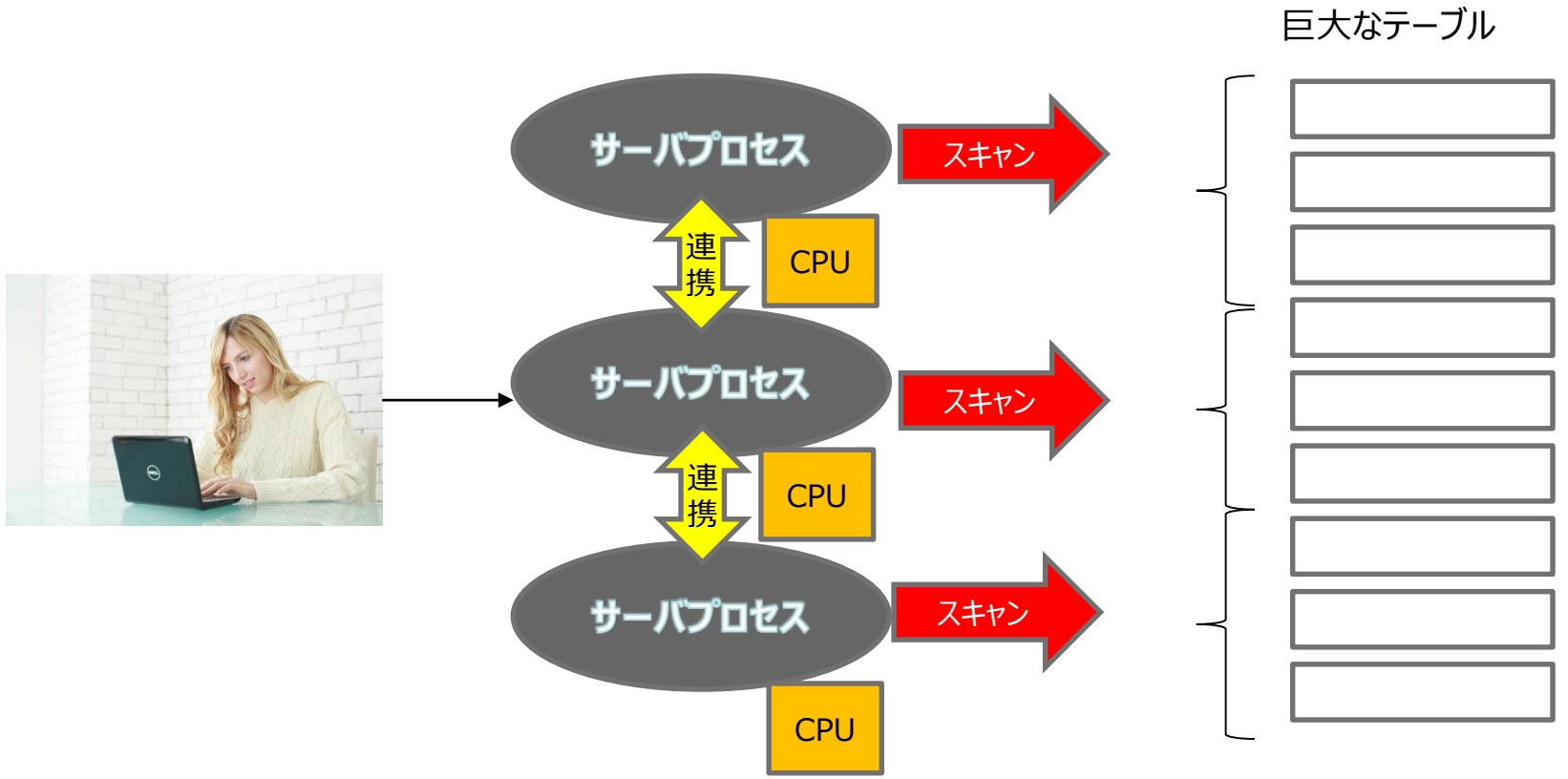




パラレルクエリ

今回の動作…

連携して平行動作するので、処理時間が短縮される！



postgresql.conf

Postgresqlの動作決めるパラメタを定義するファイル

パラメタ	意味	設定値
max_connections	最大接続数	500
shared_buffers	総メモリの25%	8G
effective_cache_size	総メモリの75%	24G
work_mem	標準16M	16M
maintenance_work_mem	Vacuumバッファ	2G
min_wal_size	チェックポイントとwalの	4G
max_wal_size	コントロールを行う	8G
checkpoint_completion_target	OSがフラッシュする時間	0.9
wal_buffers	未書き込みデータ領域	16M
default_statistics_target	統計対象数	500
max_parallel_workers_per_gather	パラレルクエリ平行度(CPU数)	16
synchronous_commit	トランザクションとwalの同期レベル	off
fsync	同期処理	off

2016版から2018版への移行データ

対象テーブル：

名前	行数
history	9億3千万件(933702608)
history_str	2千百万件(21247310)
history_log	1千万件(10140221)
history_text	22万件(219808)
history_uint	42億件(4200191632)一番巨大化するテーブル
trends	3千万件(29684975)　90日過ぎるとこちらへ集約
trends_uint	1億5千万件(152478047)90日過ぎるとこちらへ集約
	上記は移行前2017年10月3日現在の値(90日データ保持設定)

しばらく運用して気が付いたこと

- PostgreSQL

  大き過ぎるDBは運用コストがかかる

  - Zabbix proxyは数日データ保存。WALはslaveに一日保存。

    2.5TBのbasebackupには半日かかります。
    WALデータをキープしておかないとbasebackupが終わったてもDBが復元できません
    DB停止中はzabbix proxyで監視データを保持します。

  - ZabbixのDBとして使用する場合は、300GB程度の容量が使いやすい

    大き過ぎるDBは復旧に時間がかかり、運用が出来なくなる。
    また、メニュー移動時に表示が遅くなる

・Pgpool-II

通常Masterを参照/更新、Slaveを参照で使用するのは吉

DBのメンテナンスはコスト増になることも

- LBモードはもろ刃の剣

ZabbixはDBステータスOKなのに、書き込みでエラーになる。

→SlaveだけOKの場合もZabbixから見るとDBはOK

- DB停止を伴う操作はPgpool-IIも再起動を行う必要がある

Pgpool-IIがDBがダウンしていると思い続けてしまう

→SlaveからMasterへの昇格をしない設定のため

Zabbix

・テーブル定義改善

最初はレスポンスも良く好調でしたが、徐々にレスポンスが悪くなってきました。
発行されているSQLを調べたところ、indexが設定されていないSQL文が発行されていて、
DBサーバの負荷を増大させ、レスポンスの悪化を招いていました。

→新たにINDEXを追加

```
CREATE INDEX ihistory_uint_20180625 ON public.history_uint_20180625 USING btree (clock, itemid);
```

・グラフ表示をGrafanaに変更

Zabbixに多くの人がアクセスするようになり、ユーザ管理コスト増大、DBの負荷も増えたため、
Grafanaでグラフを提供し表示内容を最適化した。

まとめ

・ZabbixはDBの性能に依存している
適切なサイジング(300GB程度)と、必要ならSQL文の解析が有効

・DBサイズが大きくなるとautovacuumは停止させる方が良い場合がある
truncateでテーブル削除する場合、処理が停止する事が多々あった。

・fsync=off
監視サーバの場合は速度優先で良い

・housekeeperの停止
不要データを削除する便利な機能ですが、データはtruncateで消すため
停止しておきます。

・checkpointのチューニング
Zabbixは基本的に書き込み中心なので、環境に合わせたチューニングが必要。

まとめ

・グラフの提供はGrafana
Zabbixのユーザと切り離して必要な情報を提供できるので、ユーザ管理や
サーバ負荷の低減に有利

・pgpool-II
使用を取りやめました。 負荷分散モードで快適な面もありましたが
master DBがNGになってもZabbixからはDBがOKのように見えてしまい、
大量のDBエラーが発生することになった。運用メンバーへのレクチャーなど
運用面でのコストがかかるため。

参考記事

Zabbixのデータベース　ベンチマークレポート
PostgreSQL VS MySQL

Yoshiharu Mori　SRAOSS Inc. Japan
https://www.sraoss.co.jp/event_seminar/2013/20130829_Zabbix_PG_vs_MySQL.pdf

PostgreSQLを使うならZFS
https://blog.ohgaki.net/postgresql-should-use-zfs
大垣さんのブログ

上記の記事は大変参考になりました。
また、URLのご紹介を快諾頂き感謝いたします。

質疑応答

質問等あればお願い致します。

クロージング

ご清聴ありがとうございました。