

PostgreSQL 9.0 とクラスタリングソ リューション最新動向セミナー

2010年 7/14, 7/28, 8/4, 8/18

SRA OSS, Inc. 日本支社

高塚 遙 / harukat@sraoss.co.jp

はじめに

- 講演者について
- 本セッションの構成
 - PostgreSQL 9.0 のご紹介
 - PostgreSQL クラスタソリューションのご紹介
 - 高可用性クラスタについて
 - 負荷分散クラスタについて

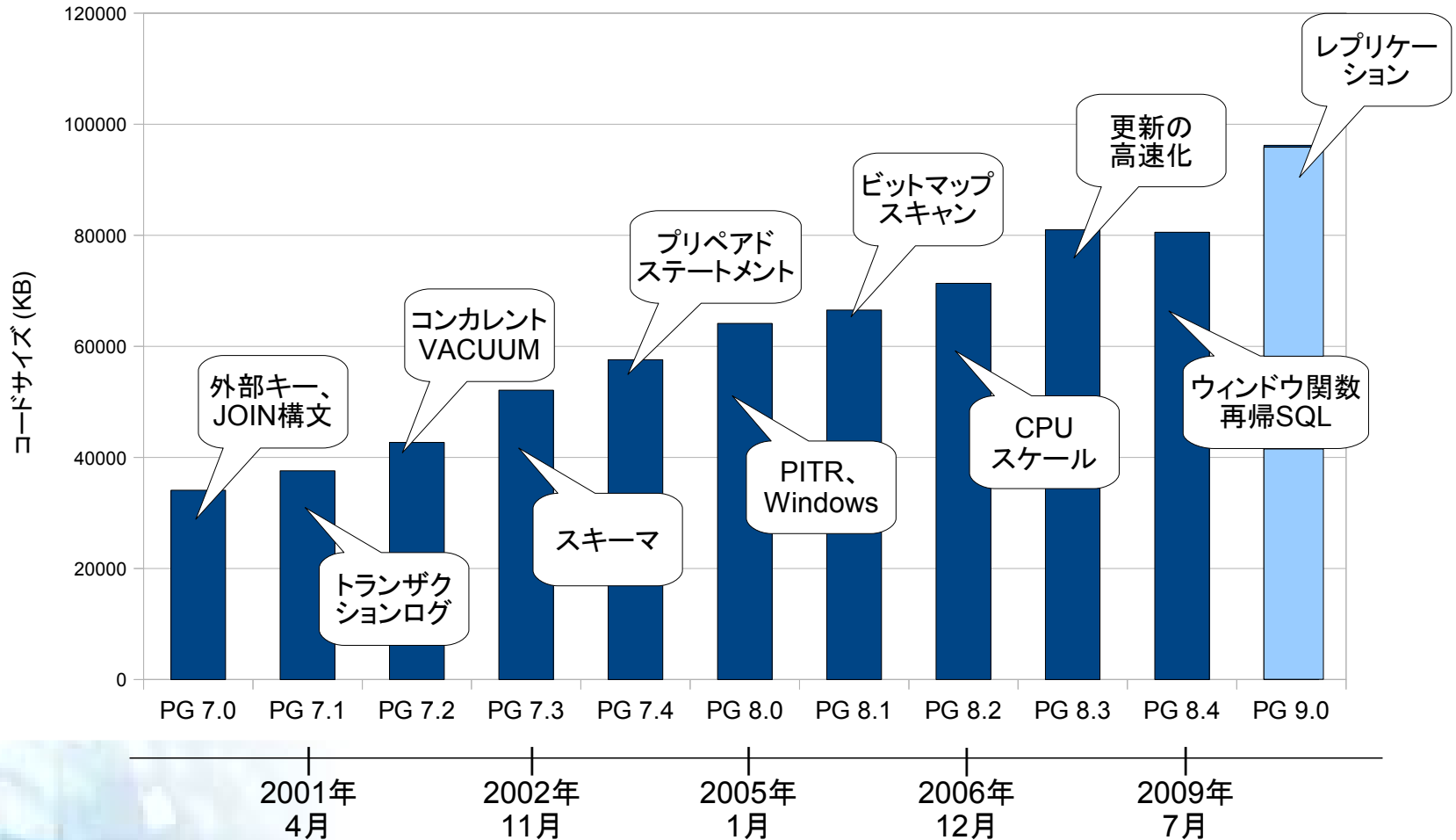
PostgreSQL 9.0 について

PGCON 2010
developpers meeting
Arc the Hotel, Ottawa



PostgreSQL 9.0 までの軌跡

PostgreSQL のコードサイズ



PostgreSQL 9.0 の拡張

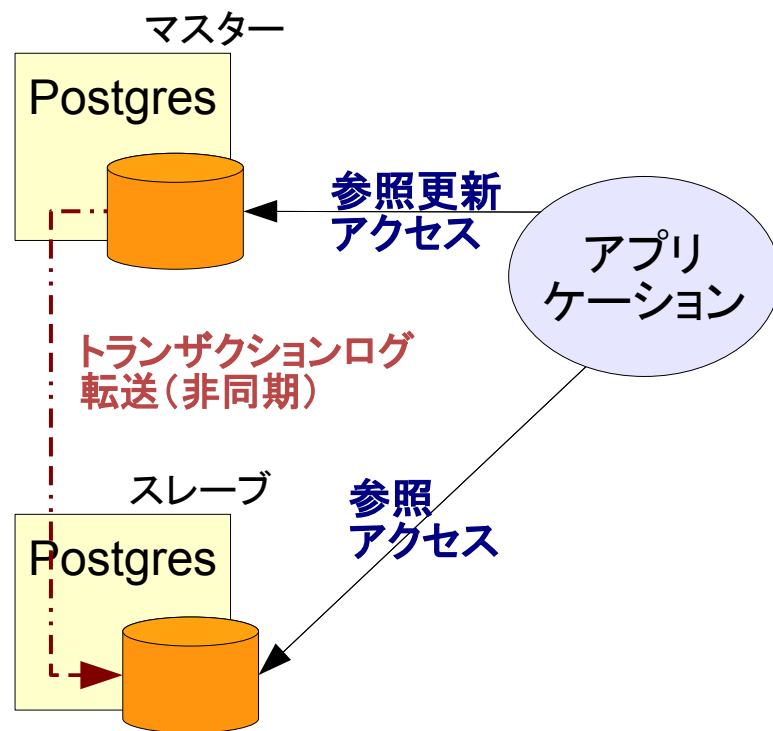


- レプリケーション標準装備
- Windows 64bit 対応
- 遅延可能なユニーク制約 / 「排他制約」追加
- VACUUM FULL のリニューアル
- LISTEN/NOTIFY のリニューアル
- 手続き言語(ストアドプロシージャ)関連を拡張
- オブジェクト権限設定を拡張
- 上書きアップグレードツール(追加ツールの拡張)

レプリケーション標準装備

HS/SR (ホットスタンバイ/ストリーミングレプリケーション)

- シングルマスター
 - マルチスレーブ
 - カスケードは非対応
- トランザクションログ転送
 - block転送 (DRBD)、行転送 (Slony-I) より軽い
 - データベースクラスタ単位でのレプリケーション
 - WALファイル単位でなく、処理単位で転送する
- スレーブで参照アクセス



Windows 64bit 対応

- これまでは 32bitビルドのみ対応
 - 64bit版 Windows では WOW64 で動作
 - shared_buffers や work_mem に 2GB～4GB を超えるサイズを設定できなかった
- Visual Studio 2008 でビルド
 - PostgreSQL 8.3 までは Visual Studio 2005 が推奨ビルド環境（9.0 も 2005 でビルド可）
 - MinGW、cygwin では依然 32bit のみ



遅延可能なユニーク制約

- 以下のようなことができる(これまでできなかった)

```
db=# \d test
```

```
Table "public.test"
```

```
Column | Type      | Modifiers
```

```
-----+-----+-----
```

```
id      | integer   | not null
```

```
v       | text      |
```

```
Indexes:
```

```
"test_pkey" PRIMARY KEY, btree (id) DEFERRABLE
```

```
db=# BEGIN;
```

```
db=# SET constraints all DEFERRED;
```

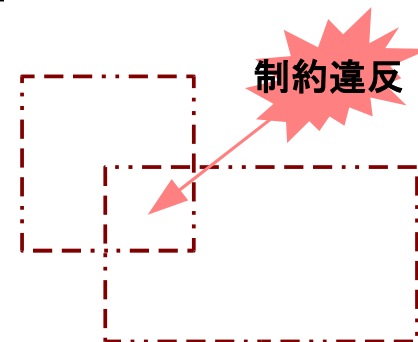
```
db=# UPDATE test SET id = id + 1 ;
```

```
db=# END;
```

id	~
1 → 2	~
2 → 3	~
3 → 4	~

排他制約 EXCLUDE

- 「ユニーク」概念を拡張したものの
 - 任意の演算子で排他制約を
 - gist インデックスのみ対応
 - 応用は色々だが既存データ型ではすぐ使えるものが少ない
 - 組み込み型では BOX などの幾何データ型に適用可能

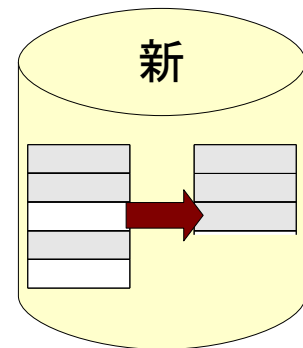
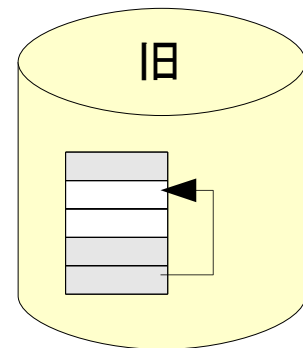


```
CREATE TABLE real_estate_registration (  
  regist_id      int8,  
  land           geometry,  
  term           period,  
  owner         text,  
  EXCLUDE USING gist  
    (term WITH &&, land WITH &&));
```

「重なっているか？」
の演算子／「＝」なら
UNIQUE 制約と同じ

VACUUM FULL リニューアル

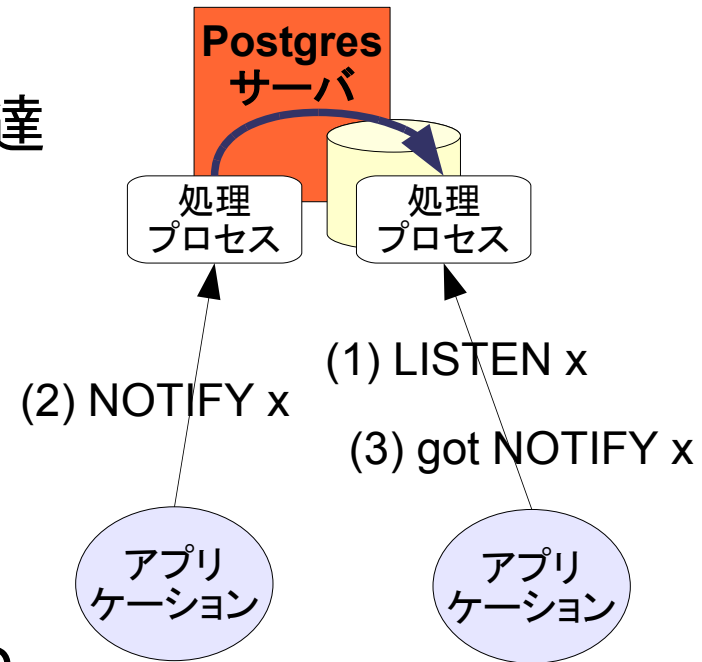
- 高速動作／実施後の REINDEX も不要に
 - 内部的にテーブルを作り直す動作になる
 - そもそも VACUUM FULL は
 - 不要領域を削除してデータ物理サイズを縮小する
 - 通常VACUUMをよく行い、使わずに済むのが王道
 - これまでの VACUUM FULL は
 - 代替にCLUSTERコマンドを使う
 - 代替にダンプ／リストアを使う
 - パーティショニングをしてテーブルごと削除
- といった技法がしばしば使われるほどに、
非効率な方式となっている



動作イメージ

LISTEN/NOTIFY のリニューアル

- 接続セッション間での通信手段
 - PostgreSQL独自の機能
 - 「ある件について通知が発生」を伝達
 - 各種接続ドライバで対応
- 高速化
 - テーブルで実装していたものを、メモリ上のキューで実装
- 任意の文字列メッセージを渡せる



関数／手続き言語の拡張(1)

- DO文を使って匿名関数を作成

```
db=> DO LANGUAGE plpgsql
db-> $$ BEGIN FOR i IN 1..3 LOOP RAISE NOTICE 'hello';
db->           END LOOP; END; $$;
NOTICE:  hello
NOTICE:  hello
NOTICE:  hello
```

- いきなり記述できる
- バッチ処理記述など

- 関数呼び出しにおいて自由な引数順序

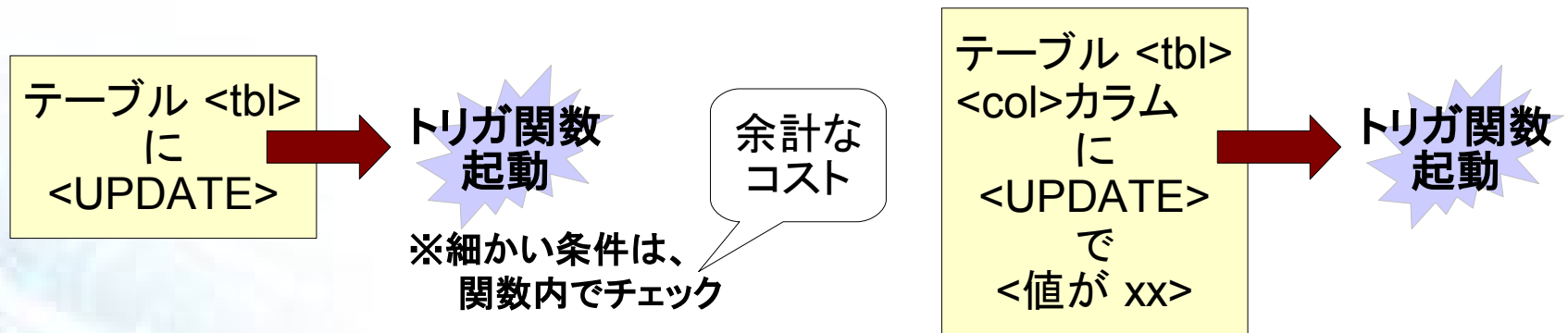
```
db=> SELECT f(p3 := 12, p1 := 7);
```

これまでは末尾の引数にしかデフォルト値が
利用できなかった

```
f(p1 int DEFAULT 0,
  p2 int DEFAULT 0,
  p3 int DEFAULT 0)
```

関数／手続き言語の拡張(2)

- カラム単位のトリガ
 - 特定カラムの変更に対して駆動するトリガ定義
- 条件付きトリガ
 - 処理対象行の値が特定条件のとき駆動するトリガ定義



- PL/Perl が大幅に拡張、PL/Python も拡張

オブジェクト権限設定の拡張

- ラージオブジェクトにアクセス権限設定
 - これまで欠けていた機能
 - デフォルトでは他のユーザが所有するラージオブジェクトにアクセスできないようになる **(非互換注意)**

```
db=# GRANT SELECT, UPDATE ON LARGE OBJECT 16577 TO foo;  
GRANT
```

- スキーマごとにまとめて権限の変更
 - 「このスキーマに属するテーブル全てに一度に権限設定」ができるようになります

```
db=# GRANT SELECT ON ALL TABLES IN SCHEMA scm1 TO role1;  
GRANT
```

追加ツール（contrib）の拡張

- 上書きアップグレードツール pg_upgrade 新登場

- 8.3.x、8.4.x からのアップグレードに使用可能
- 内部データ書き換えを行う

以前は pg_migrator と呼ばれていた

- contrib/pgbench の拡張

- マルチスレッド対応
- シェルコマンド実行機能

マルチスレッド対応でないと、クライアントマシン側が先に限界となって、性能計測仕切れないケース

- contrib/hstore の拡張

- ハッシュデータ型（キーバリューの組）
- キー長、バリュー長の制限が撤廃された
- GROUP BY、DISTINCT が利用可能に

以前は、キーも値も、65535バイトが最大長

PostgreSQL 9.0 いつリリース？

- 7/12 ~ 7/13 に beta3 が公開
 - 僅かながら beta2 に機能追加もされている
- その後に rc (release candidate) を 1つ 2つ出す
- 正式リリースは早くとも 8月

PostgreSQL 9.1 の展望

- 同期レプリケーション
- MERGE
- JSON型、各種Range型
- マテリアライズドビュー
- パラレル pg_dump
- テーブルパーティショニング構文
- SQL/MED（外部表参照）
- SE-PostgreSQL
- 書き込み再帰SQL
- プリペアドステートメントのプラン改善
- インデックスのみスキャン

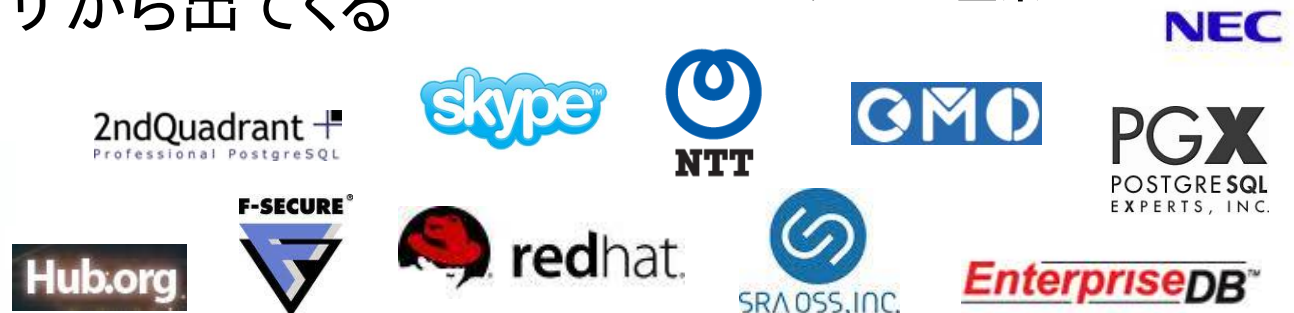
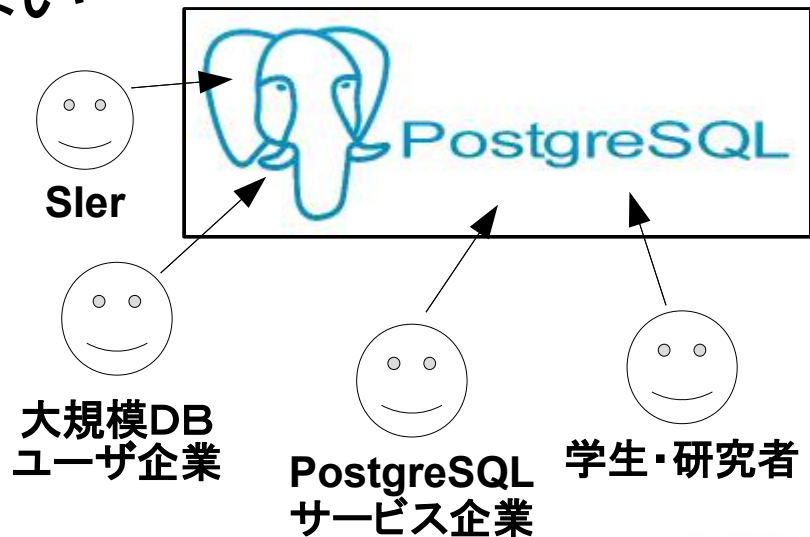


PgCon 2010
Developer Meeting より

PostgreSQL の展望

- PostgreSQL開発は止まらない

- 特定オーナー企業は無い
(支援企業も一定ではない)
- 世界に多数ある関連サービス企業が主力
- 時に研究実証に使われる
- 大規模ユーザから出てくる新機能



休憩



猫 Tokyo/Japan

PostgreSQL の クラスタソリューション一挙紹介

- 高可用性クラスタ
- 負荷分散クラスタ

目的あつての
クラスタリング

高可用性 クラスタ について



カルタゴの武装した象
Parco de Mostri di Momarzo

データベース高可用性クラスタ

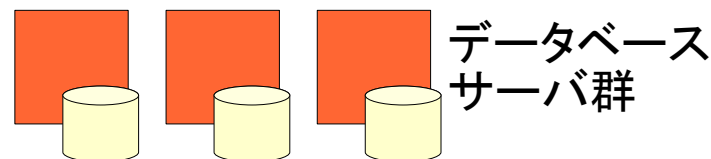
- 可用性の追求
 - サービス停止時間を短く
 - 避けえない HW/SW 障害
 - 復旧時間の短縮
 - ↓
 - クラスタにする
 - 並行で別サーバを稼働させればよい
 - Single Point of Failure
 - DBMS は データを持つ

- クラスタソフト化に必要な要素は

↓ 処理要求

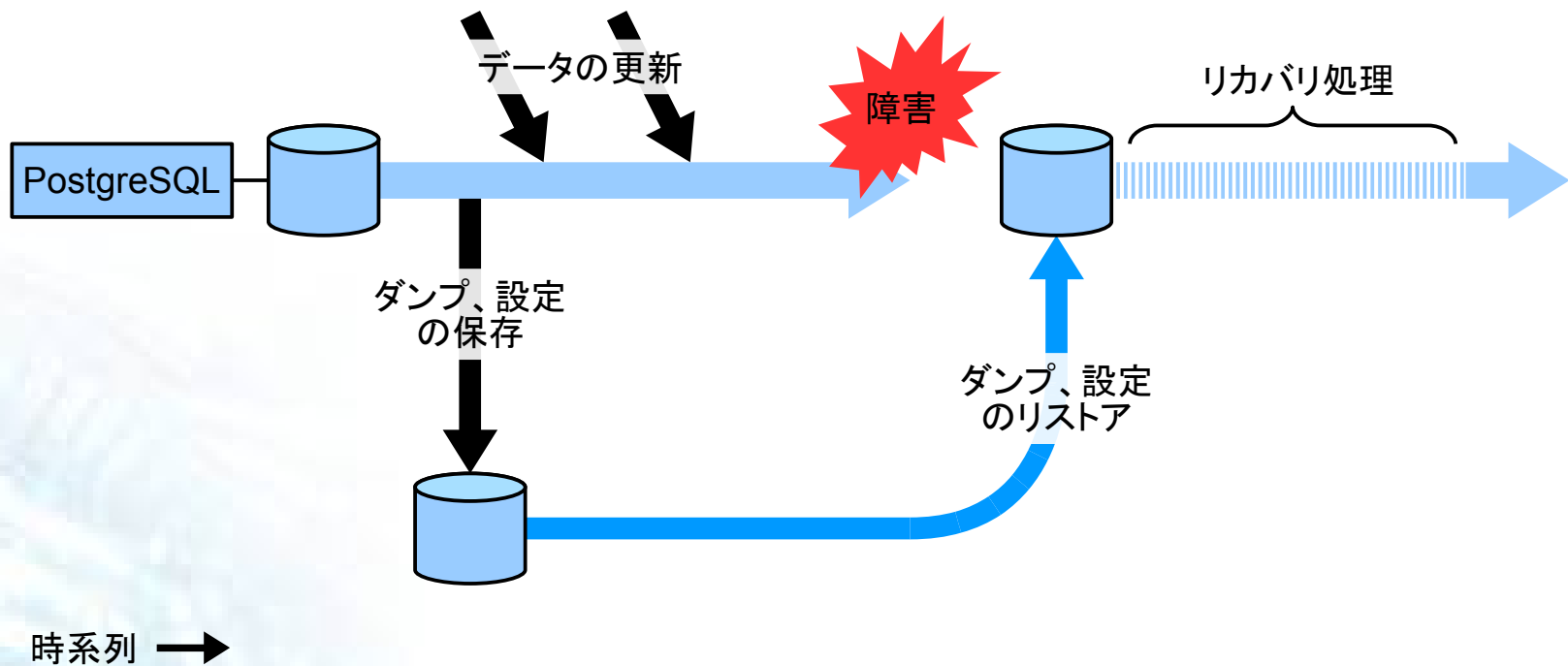
(1) サービスの維持
死活監視、フェイルオーバー

(2) 一元的なデータの維持
共有ディスク、レプリケーション



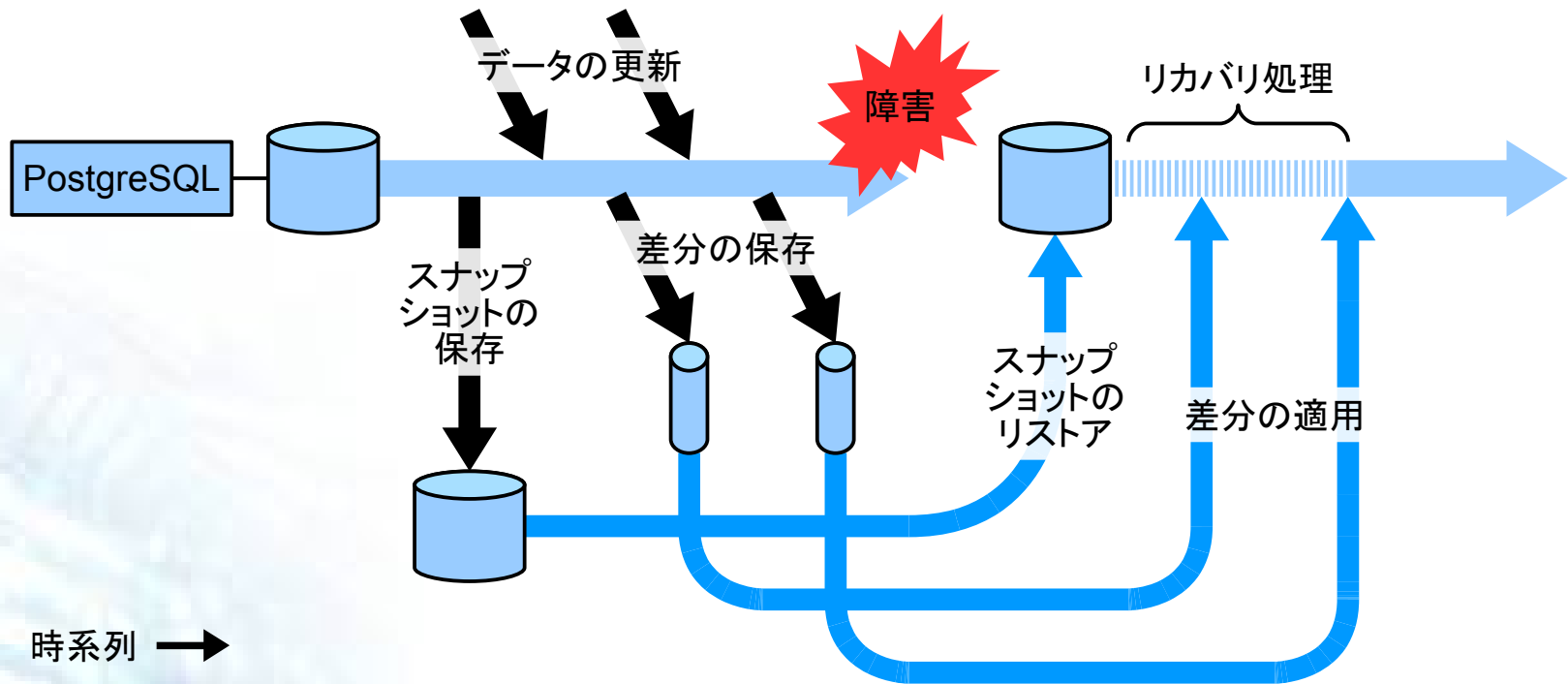
クラスタ以前(1) ダンプによるバックアップリストア

- 障害の発生時にはバックアップ時点まで復旧



クラスタ以前(2) PITR によるバックアップリストア

- データの更新によって発生した差分を保存し、障害の発生時には差分を適用して直近の状態まで復旧



Single Point of Failure

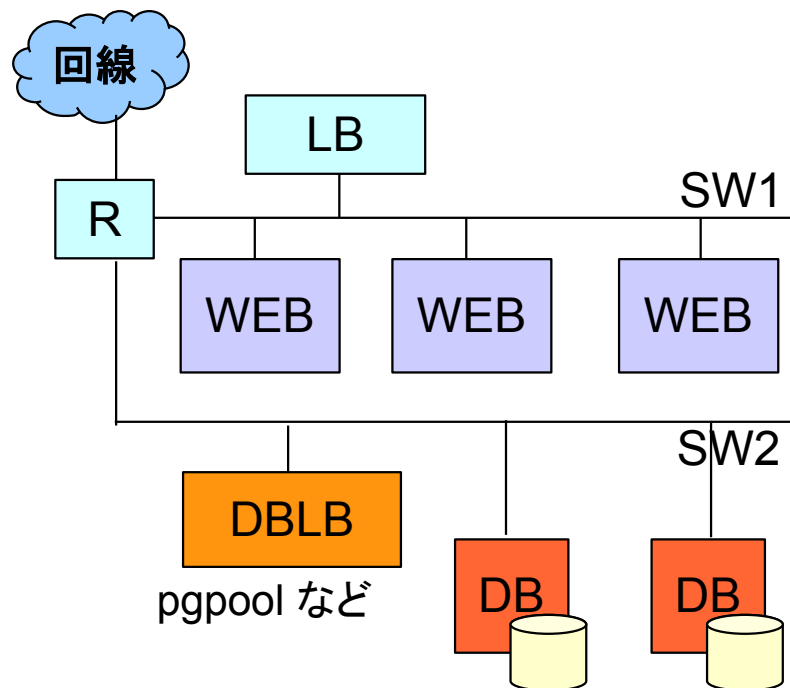
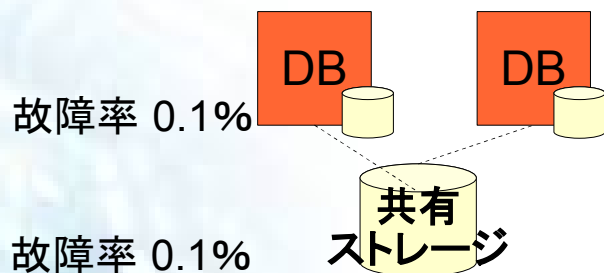
- 全体の可用性は弱点の可用性にひっぱられる
- 現実には全2重化はしてないことが多い

単体可用性 99.9 %



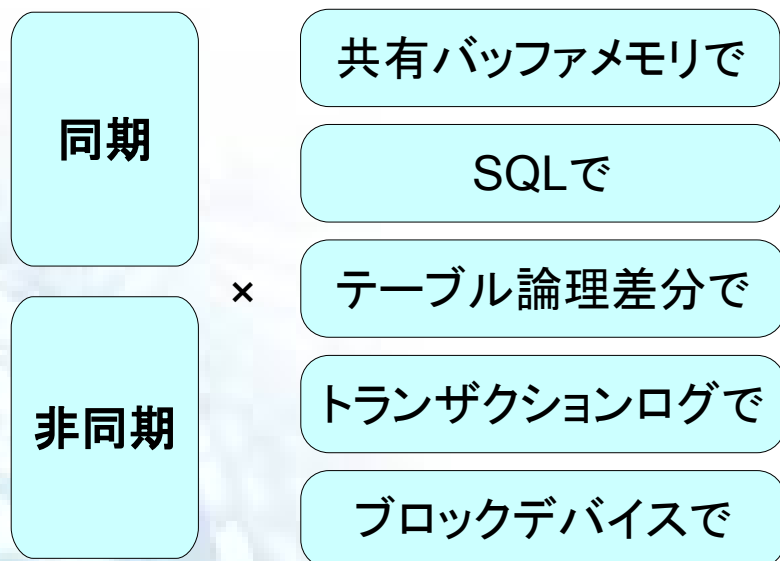
故障率 0.1% (8h/年)

クラスタ可用性 99.8999% あれ？
 $= (1 - 0.001 \times 0.001) \times (1 - 0.001)$



データベース管理システム (DBMS) は データを持つ

- クラスタソフトウェアに
重い課題
 - ノード間データ維持をどう
するか？
- 「データ保全」がサービ
ス可用性よりも重いこと
がある

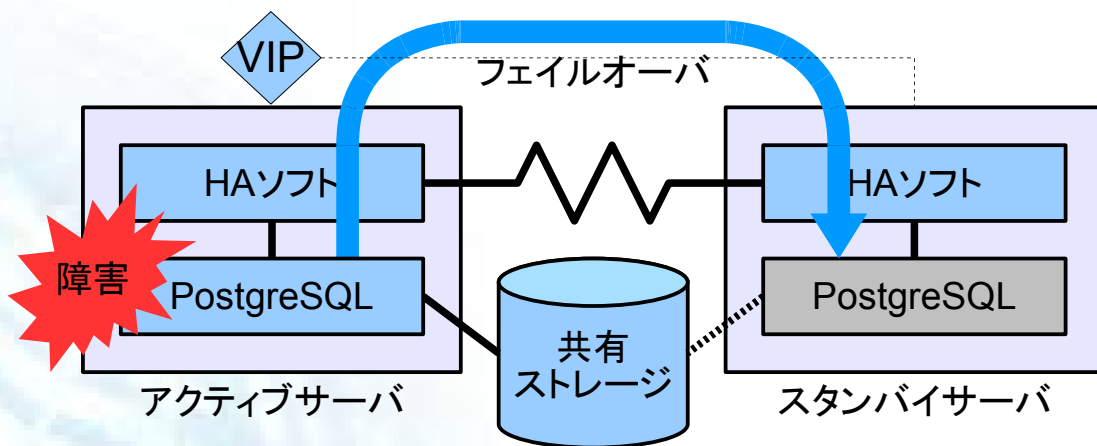


可用性が落ちる
レプリケーションにも
価値がある

PostgreSQL 高可用性クラスタ 構成例

HAクラスタ(共有ディスク)構成

- かつては PostgreSQL の実用的な唯一選択肢
- 相互アクティブスタンバイは可能だが、両アクティブに対応したミドルウェアはない
- SQL制約なし／処理オーバーヘッドなし



<HAソフト>

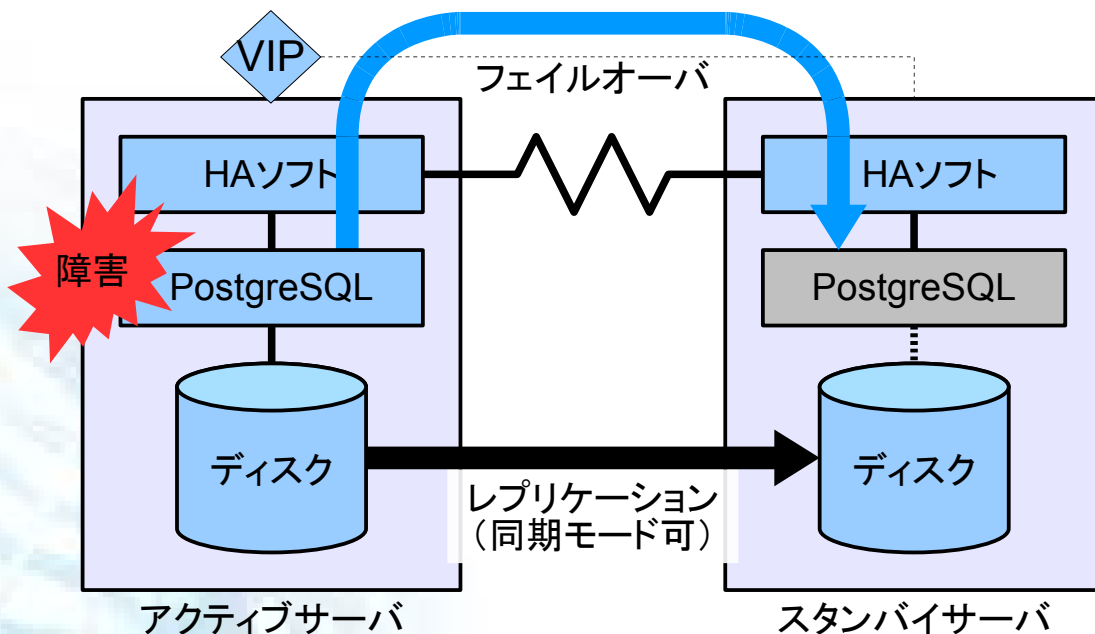
Heartbeat/Pacemaker
 LifeKeeper
 CLUSTERPRO
 RHCS
 Veritas CS
 PRIMECLUSTER
 etc..

PowerGres on Linux + Lifekeeper で
 データベースサーバの業務継続を実現

PowerGres
 on Linux **HA**

HAクラスタ(DRBD)構成

- ネットワークRAIDによるレプリケーションを使う
 - DRBD、商用HAソフト付属のレプリケーションツール
- SQL制約なし、処理オーバーヘッドあり、

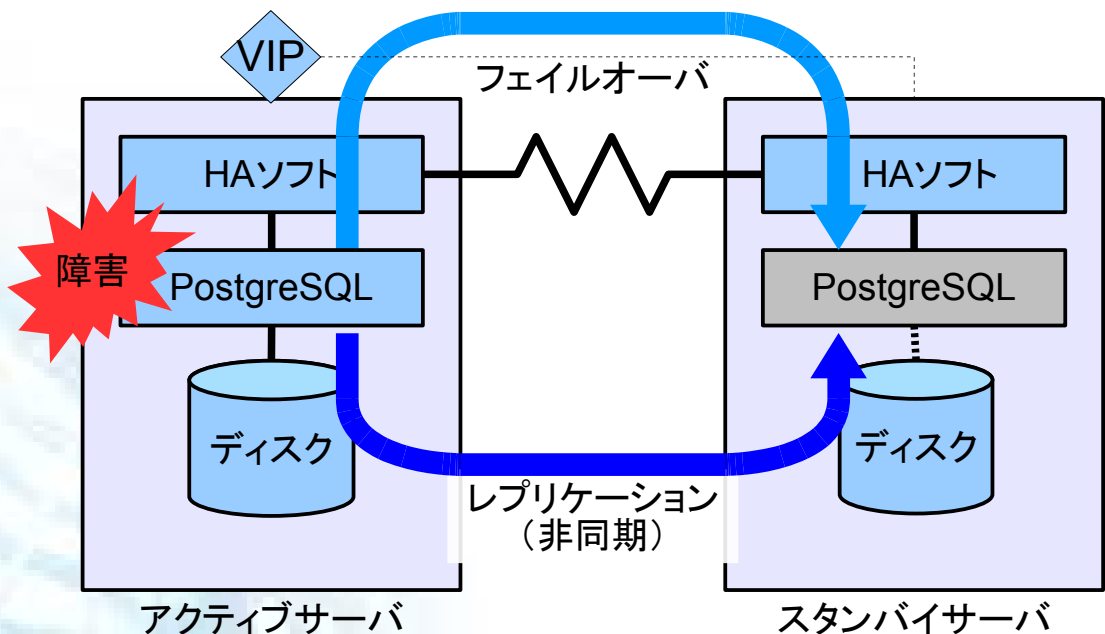


<HAソフト>
 Heartbeat/Pacemaker
 LifeKeeper
 CLUSTERPRO
 RHCS
 Veritas CS
 PRIMECLUSTER
 etc..



HAクラスタ(その他)構成

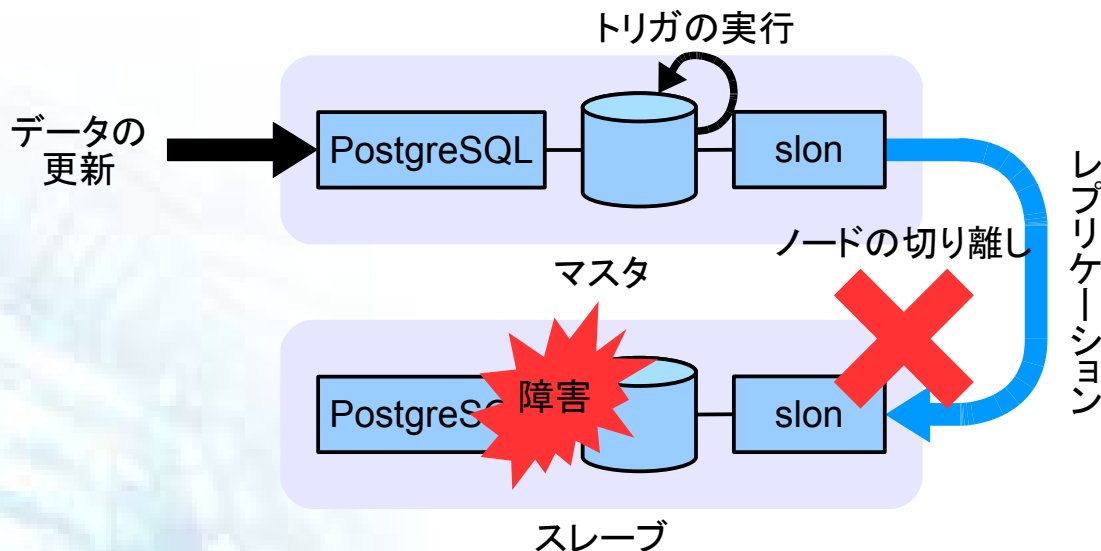
- 各種レプリケータを使う
- HAソフトとの連携について、作りこみが必要
- SQL制約あるものも、処理オーバーヘッドあり



<各種レプリケータ>
 ウォームスタンバイ
 9.0 HS/SR
 Slony-I
 Bucardo
 Londiste

Slony-I について

- 1つのマスタのみがデータの更新を受け付け、トリガによって差分を保存し、slonが複数のスレーブに差分を転送してレプリケーション
- 指示をうけて、スレーブのマスタ昇格ができる



大量書き込みでは
転送遅延が生じる

<同類ソフト>

Bucardo
マルチマスター可

Londiste
PgQでやや軽いか

VMware HA + プロセス監視

- 既に仮想化インフラが整備されているとき
- プロセス監視、再起動の仕組みが要る

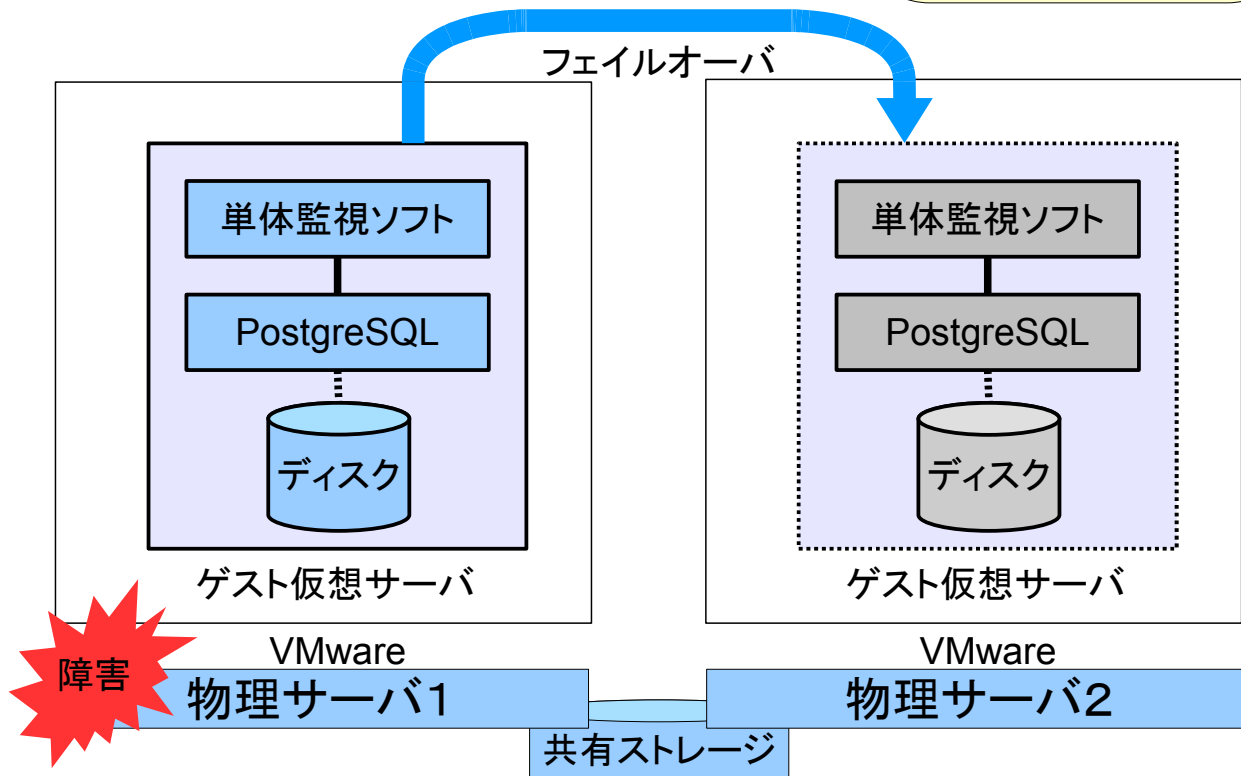
ftServer等も
得失は似ている

<単体監視ソフト>
CLUSTERPRO SSS
monit

各種HAソフトの単体
むけ使いまわしも可

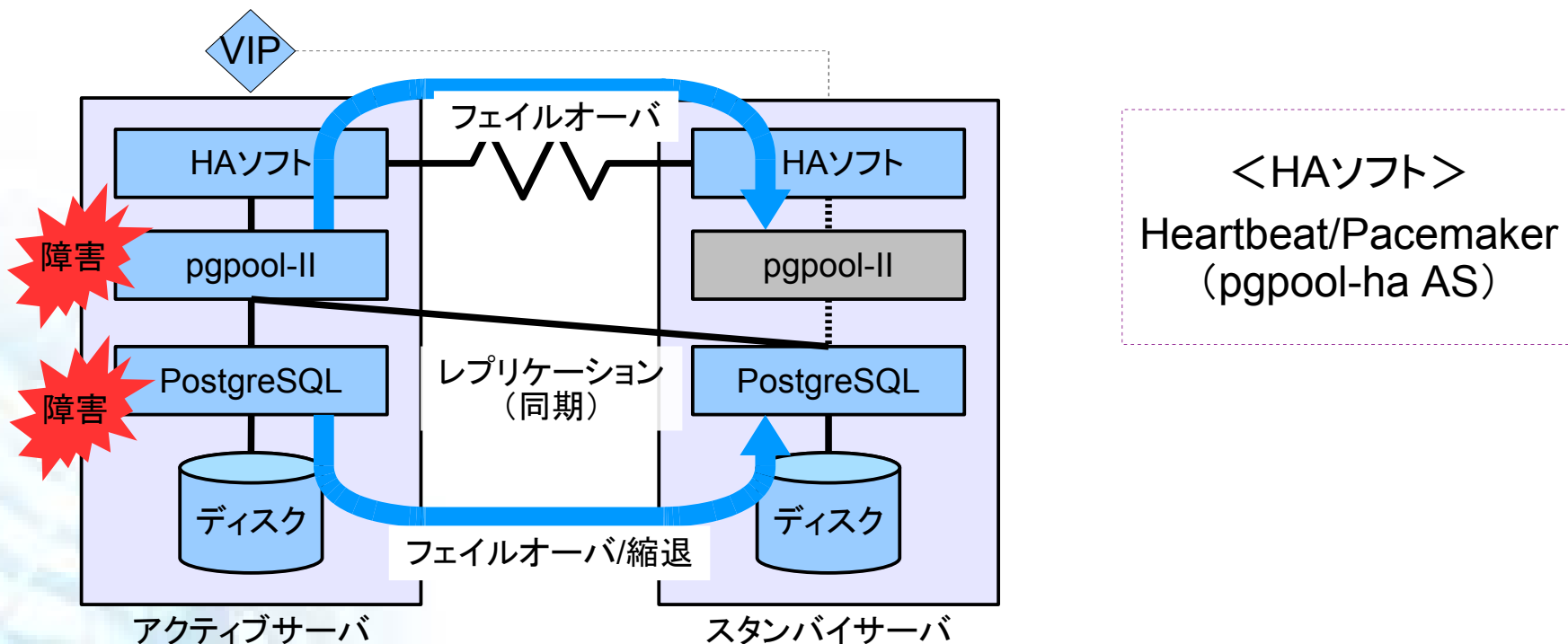
<仮想化ソフト(HA)>

各種HAソフトの上に
仮想化ソフトを載せる
パターンがいくつか可



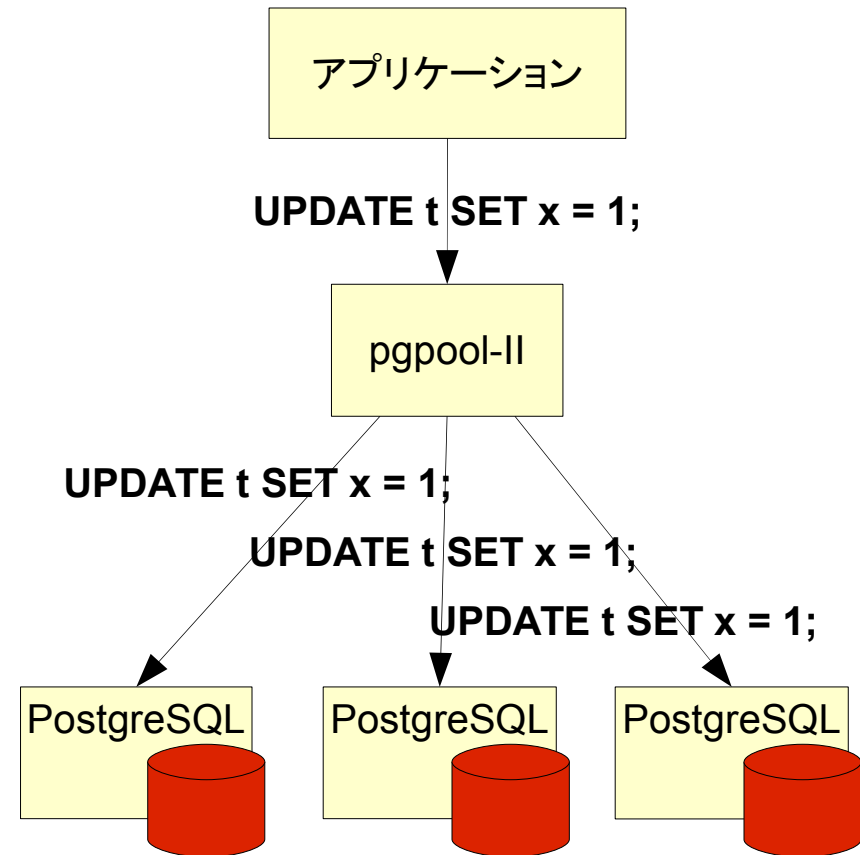
pgpool-HA 構成

- pgpool-II の読み取り負荷分散が併用できる
- SQL制限あり、書き込み処理オーバーヘッドあり



pgpool-II について

- PostgreSQLのプロキシサーバ
 - レプリケーション
 - フェイルオーバー
 - オンラインリカバリ
 - スクリプトを設定して、他のレプリケータと連携
 - 参照負荷分散
 - パラレルクエリー
 - クエリーキャッシュ



Feature	Shared Disk Failover	File System Replication	Hot/Warm Standby Using PITR	Trigger-Based Master-Slave Replication	Statement-Based Replication Middleware	Asynchronous Multimaster Replication	Synchronous Multimaster Replication
Most Common Implementation	NAS	DRBD	PITR	Slony	pgpool-II	Bucardo	
Communication Method	shared disk	disk blocks	WAL	table rows	SQL	table rows	table rows and row locks
No special hardware required		•	•	•	•	•	•
Allows multiple master servers					•	•	•
No master server overhead	•		•		•		
No waiting for multiple servers	•		•	•		•	
Master failure will never lose data	•	•			•		•
Slaves accept read-only queries			Hot only	•	•	•	•
Per-table granularity				•		•	•
No conflict resolution necessary	•	•	•	•			•

PostgreSQL 9.0 マニュアルより

負荷分散 クラスタ について



Balance à tabac (wikipedia commons)

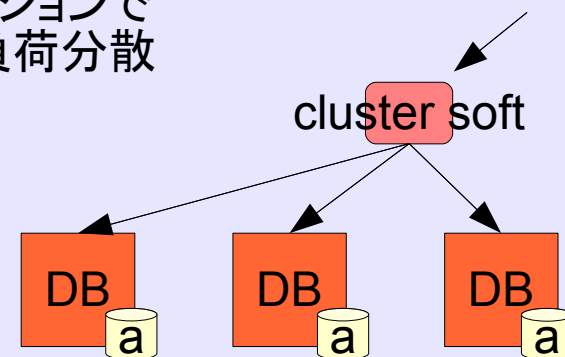
性能アップをめざしてスケールアウト

- RDBMSはデータを持つ
 - スケールアウトしにくい
 - スケールアップは有効
 - PostgreSQL: 8~16CPU
 - シェアードナッシングも適所では有効
 - OLAP分野むけソフト多数
 - アプリ巻き込みも有力

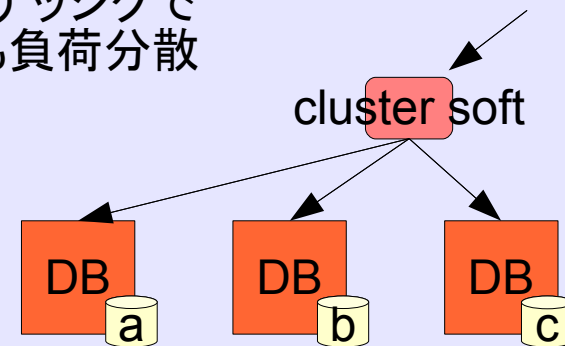
Key-Value型DB なら
スケールアウトする？

⇒ Key-Value 用途だけなら
RDBMSだって速いしスケールする

レプリケーションで
読み取り負荷分散



シェアードナッシングで
書き込みも負荷分散



可用性も！ 負荷分散も！

• CAP定理

▪ Consistency

- 一貫性
- データ一致！ 遅延差異なし！

▪ Availability

- 可用性
- 待たせない！ も含む

▪ Partition-Tolerance

- 分割透過性
- 分散して動く！ 部分でも動く

3つを全ては満たせない！ →

- 「レプリケーション+負荷分散型」なら「高可用性も」「負荷分散も」となりそうだが、易しくない

- 形のうえでできても、たいしてそれは「ノードを足すほどに『遅い』」

- 銀の弾丸は無い

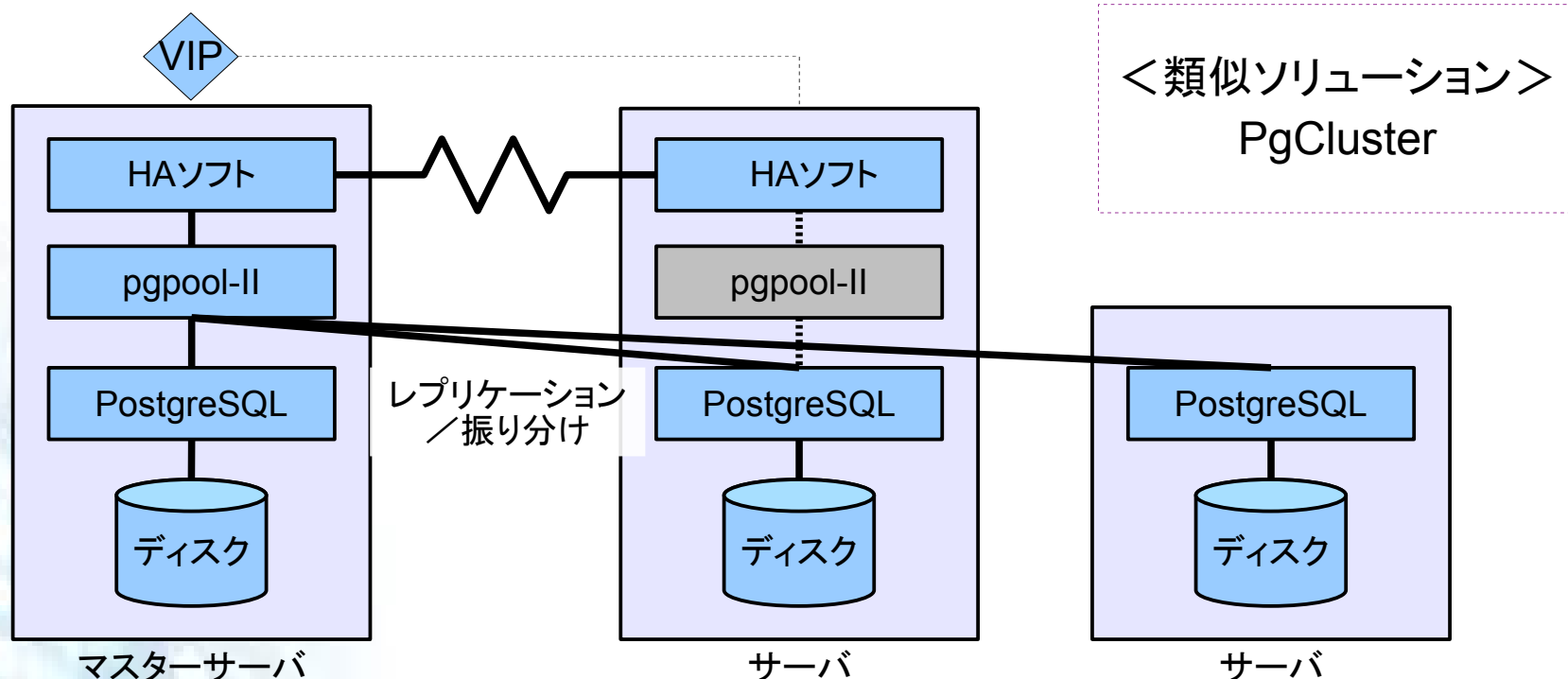
- 特定条件にマッチしたソリューションはあっても、万能構成は無い

システム全体としてあきらめどころを見つける

PostgreSQL 負荷分散クラスタ 構成例

pgpool-II 読み取り負荷分散

- SQL制限あり、書き込み処理オーバーヘッドあり
- アプリケーションから透過的に使える



pgpool-II 2.3.x の世界

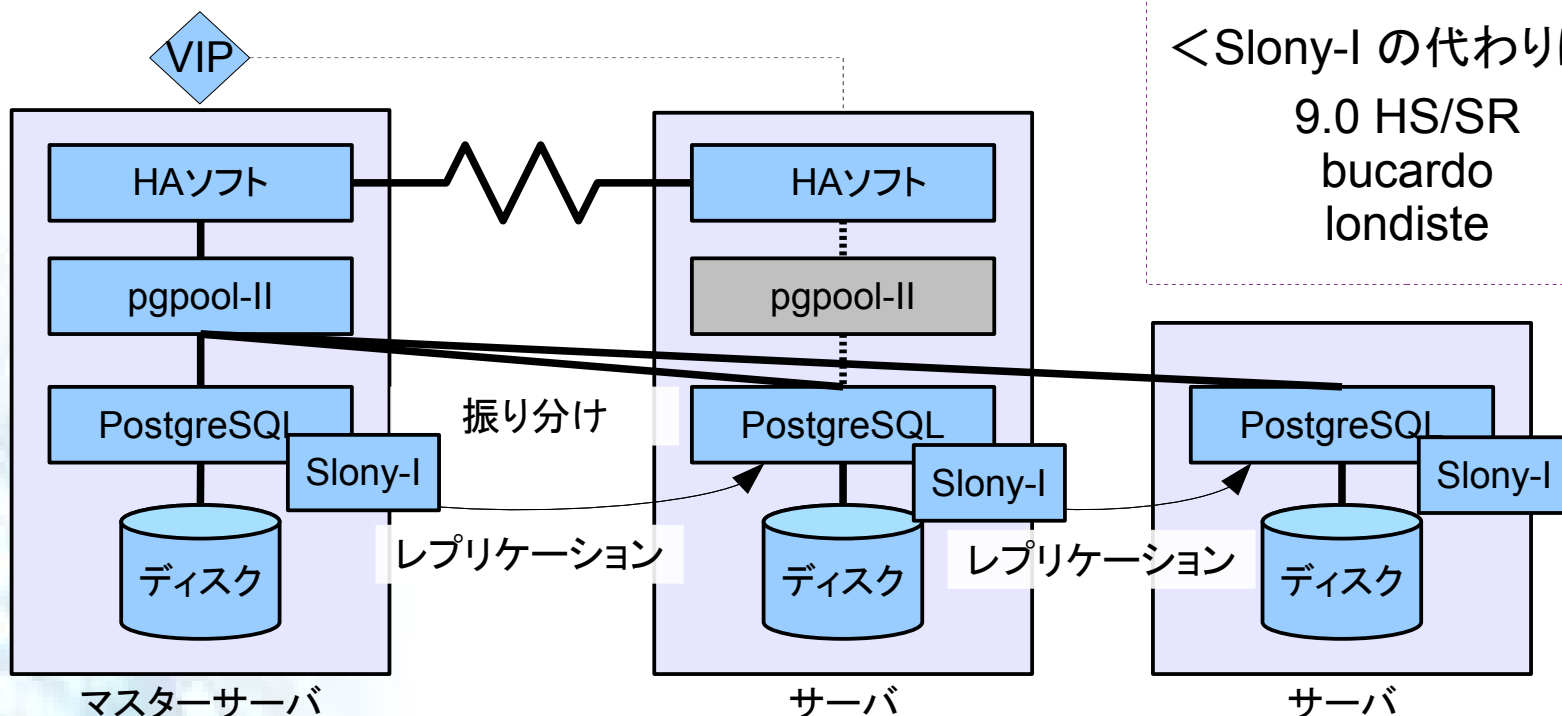
- タイムスタンプデータの挿入ができる

```
INSERT INTO t (ts)
VALUES (CURRENT_TIMESTAMP);
```

- ノードごとにサーバ時刻が異なるので従来はNG
- マスターノードの時刻を使うように自動書き換え
- そのほか
 - フェイルオーバーする／しないの制御
 - pgpool-II 停止時に状態を記憶

pgpool-II 読み取り負荷分散 - Slony-I 等による非同期版 -

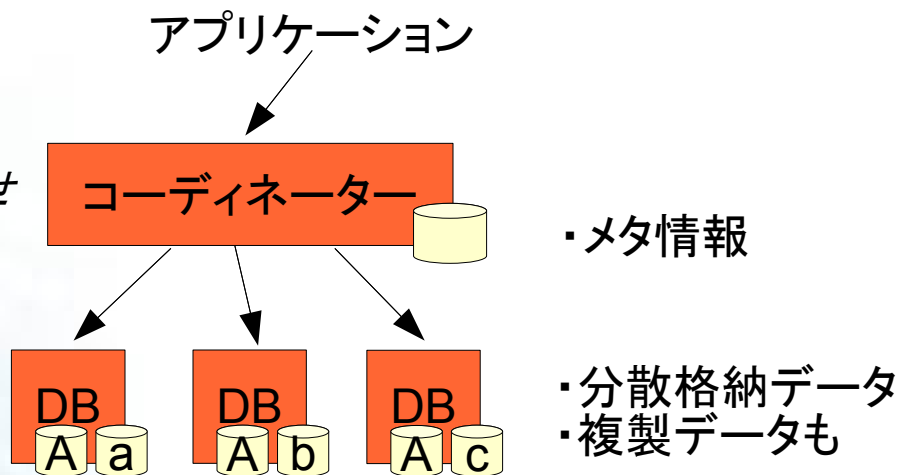
- pgpool-II のマスタースレーブモード
- スレーブサーバのデータは遅れる



シェアードナッシング + 分散クラスタ

- 巨大テーブルを水平分割して各ノードに格納
- 透過的なアクセスを可能にする分散クラスタソフトウェア
- 読み取り中心、OLAP では実用多数
 - (単純な)書き込み主体も OK、複雑で不規則な書き込みは？

格納場所の管理
分散並列問い合わせ
結果の併合、結合



<分散クラスタ>

GridSQL

pgpool-II

Postgres-XC

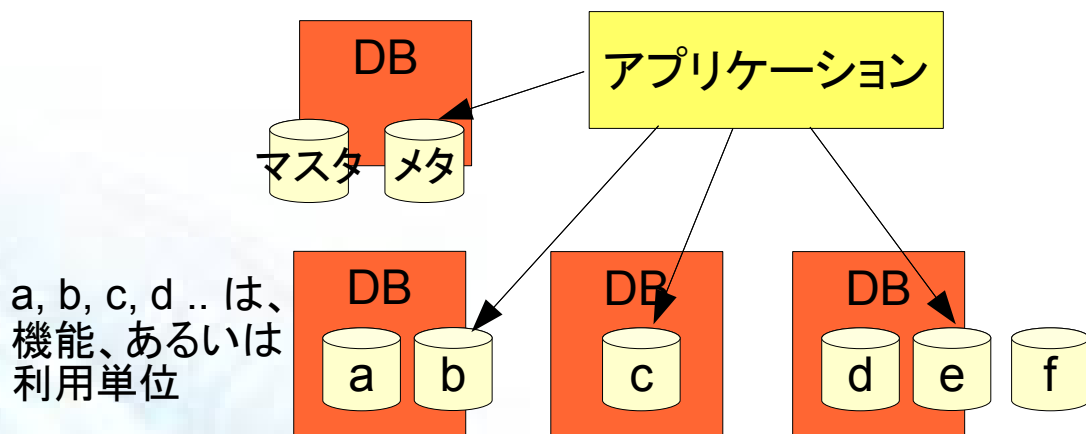
GreenPlum

postgresForest

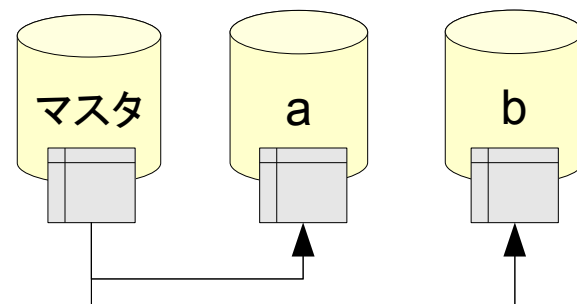
etc..

アプリケーション実装として… 機能分割、利用単位分割

- サービスの機能ごとに分ける
- 利用単位 (ASPなら契約顧客ごと等) に分ける



JOINしたい共通データを
テーブル単位でレプリケーションする



Slony-I など利用

DB分割しやすい
設計か？

分散クラスタソフトを試してうまく書き込み
性能をスケールできないなら手堅い方法

ご質問

本稿に表れる製品名・企業名・ロゴマーク等は各社の登録商標または商標です。
本稿では TMマーク等は明記していません。

