

*PostgreSQL*の障害対策について

日本PostgreSQLユーザ会

稲葉 香理

i-kaori@postgresql.jp

本日のおはなし

データベースを利用したシステム
(特に最近多いWebの3層構造システム)
では、データベースの保護が
重要なテーマです。

今回は、PostgreSQLを保護するための
障害対策についておはなしします。

データベースの障害対策

- 障害時に直前の状態まで戻す復旧策
 - バックアップ/リストア
- 障害があっても稼働し続ける高可用性対策
 - 冗長化
- 障害そのものの発生確率の軽減
 - 日々の運用

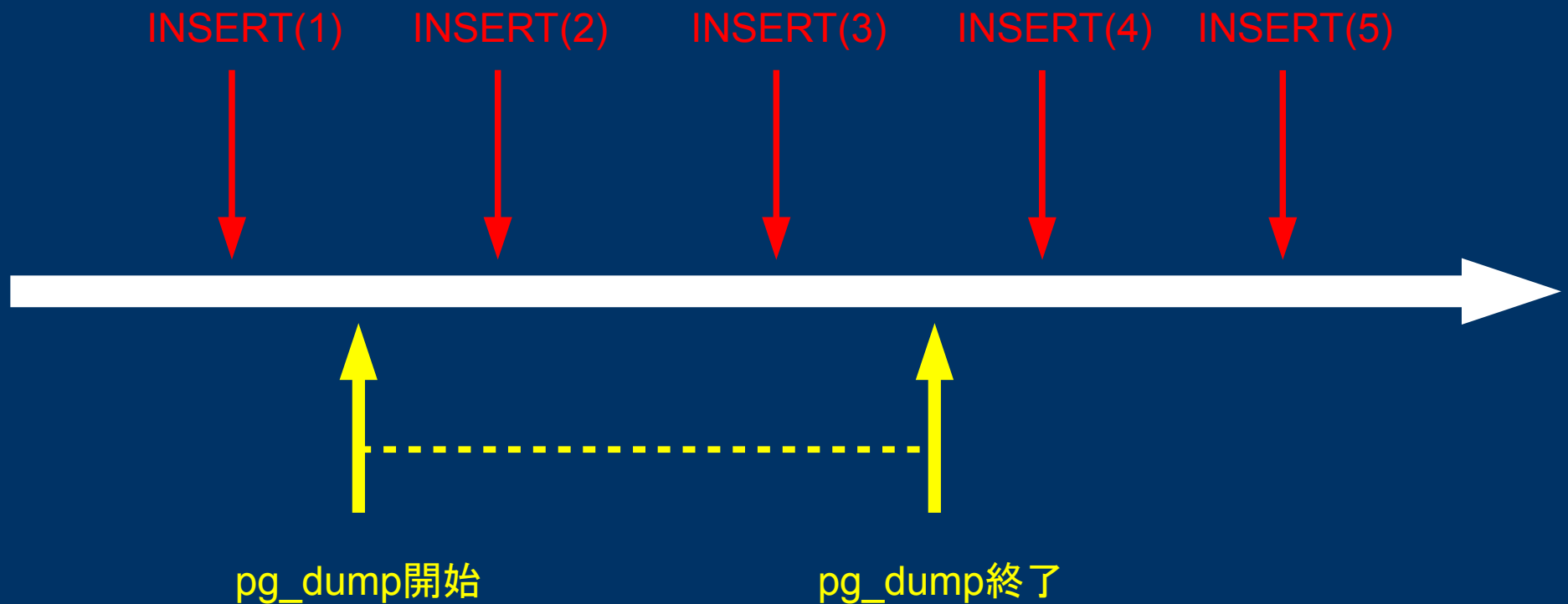
PostgreSQLにおけるバックアップ

- 論理データ
 - pg_dump / pg_restore コマンド
 - 取得開始時点で整合性がある
- アーカイブログ
 - 1: PITR (Point in Time Recovery)
 - 2: ウォームスタンバイ
 - 直前の状態まで復旧
- レプリケーション
 - pgpool-II, Slony-I などのサードパーティー
 - 高可用性の実現

測定環境

- ハードウェア
 - CPU AMD Opteron 285 2.6GHz × 2
 - メモリ 5GB
 - HDD 72GB Ultra320 SCSI × 2
- OS
 - CentOS5.3(32bit)
- PostgreSQL
 - 8.4
- データベース
 - pgbenchで作成(-s 1000, 約15GB)

pg_dump によるバックアップ



バックアップにふくまれるのは、INSERT(1) まで

pg_dump の特徴

- メリット
 - 手順が簡単
 - 事前設定不要
- デメリット
 - 最後取得時点までしか復旧できない
 - 復旧時間がかかる

お勧めの使い方

1日前に戻ればよい、復旧にある程度時間がかかってもよい時
自動的に1日1回バックアップを行う

pg_dump / pg_restore 手順

- バックアップ (約5分)

```
$ pg_dumpall > all.backup
```

- 通常のリストア (約25分)

```
$ pg_restore -d bench all.backup
```

- 並列リストア (約17分)

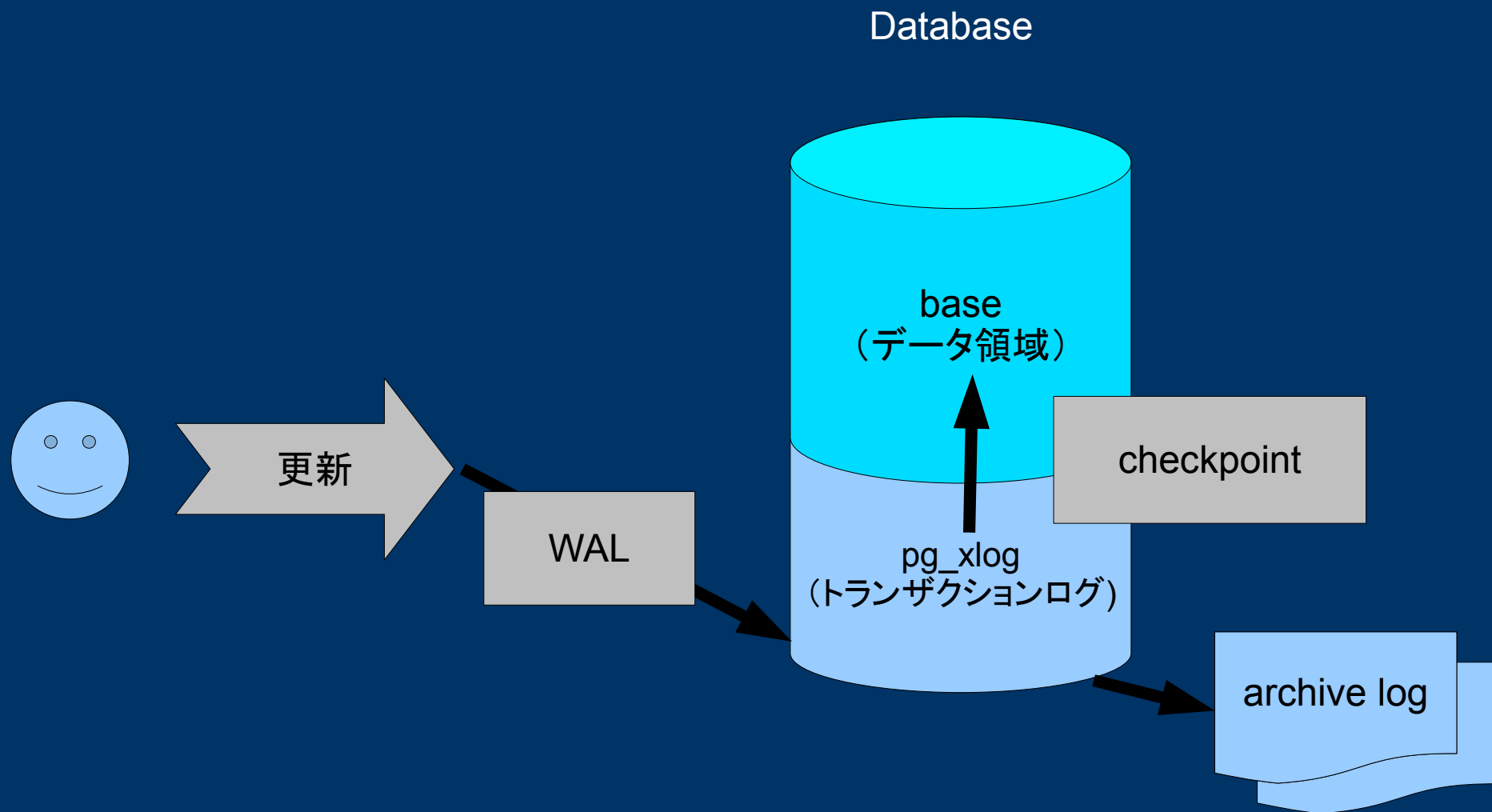
```
$ pg_restore -j2 -d test all.backup
```

ポイント

バックアップよりリストアに時間がかかる！

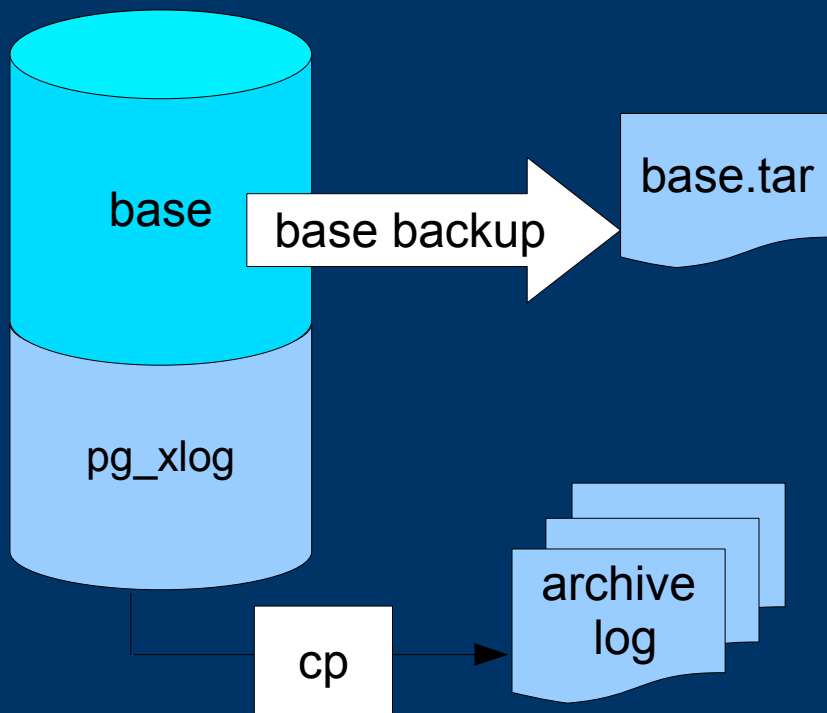
マルチCPU,マルチコアなら並列リストアの効果を期待

アーカイブログの仕組み

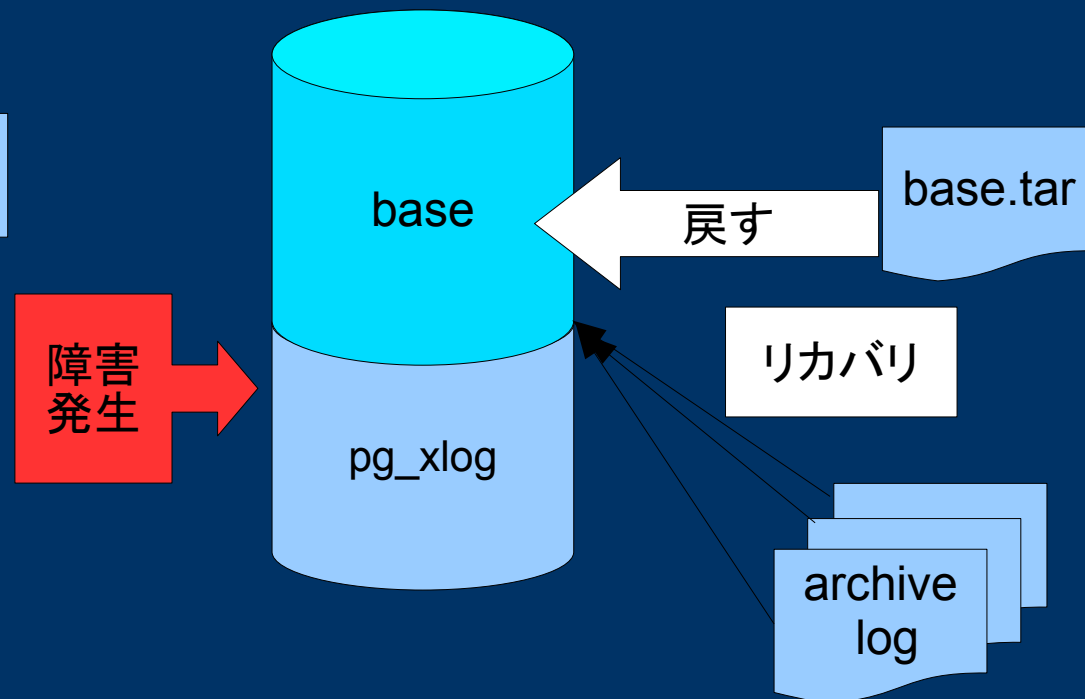


PITRの仕組み

ログをアーカイブ化



ログを適用



障害発生

PITRの特徴

- メリット
 - 直前まで復旧可能
- デメリット
 - 初期設定が必要
 - アーカイブログの保管と管理

お勧めの使い方

障害発生時の直前の状態まで戻したいケース、
ある程度の復旧時間をもてる時に

PITR手順

- 設定(postgresql.conf)

```
archive_mode = on
```

```
archive_command = 'cp %p /data2/archive/%f'
```

好きなコマンドでアーカイブの指定

- ベースバックアップ

```
$ psql -c "SELECT pg_start_backup('091031')" template1
```

```
$ tar cf base.backup $PGDATA
```

```
$ psql -c "SELECT pg_stop_backup('091031')" template1
```

データベースに宣言してから、物理的バックアップ

- トランザクションを流す

```
$ pgbench -c 100 -t 100 bench
```

PITR手順

- リカバリファイル作成(recovery.conf)
restore_command = 'cp /data2/archive/%f %p'
- ベースバックアップ復旧
\$ tar xf base.backup
- アーカイブログ復旧
\$ pg_ctl -D /data start

ベースバックアップ復旧（数分くらい）、アーカイブログ復旧（1分30秒）。手順が増えるのと人手を介すので、実際には15分はかかる？

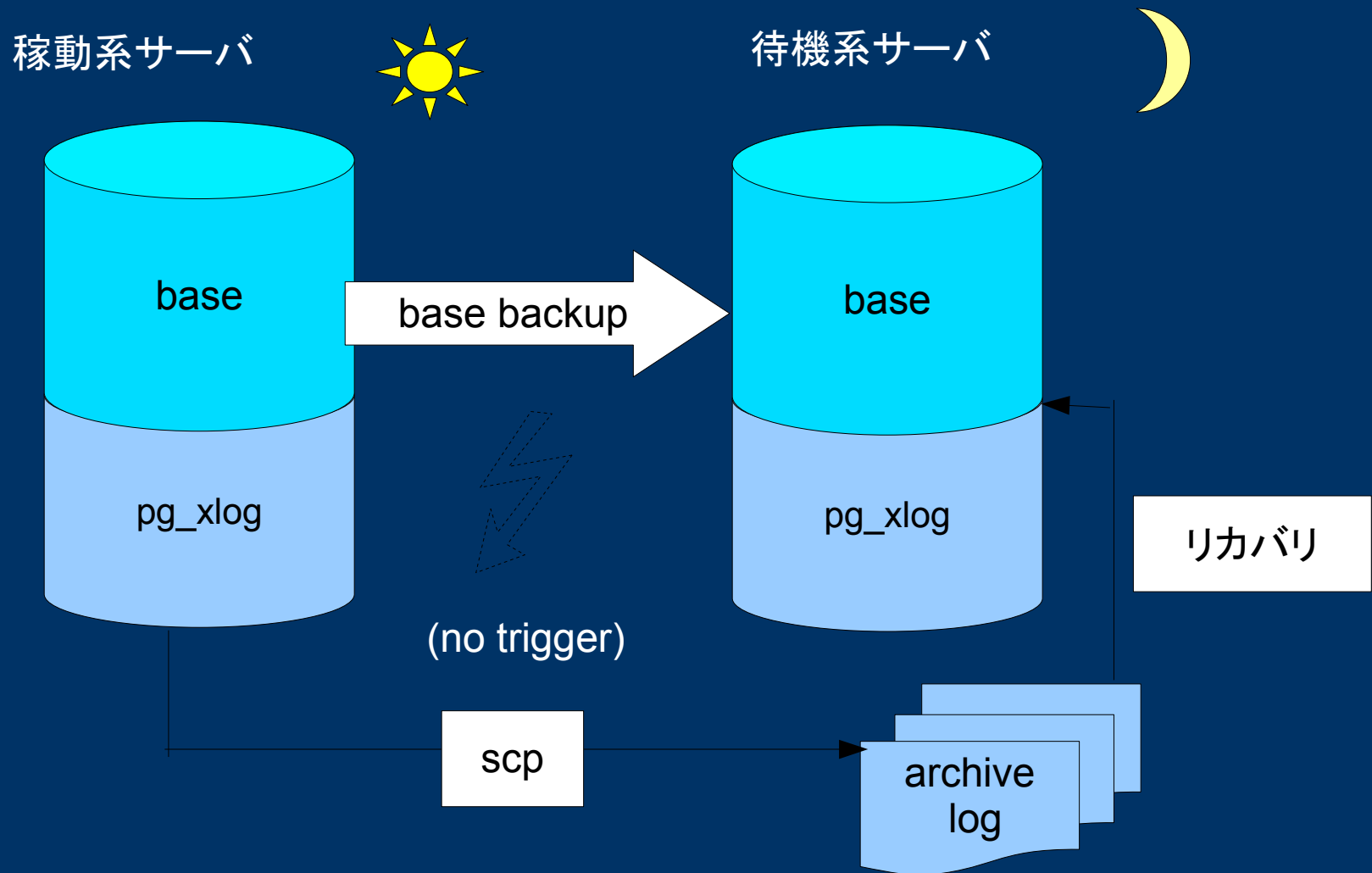
Warm Standby の特徴

- メリット
 - 復旧時間を最大限に短縮
- デメリット
 - 設定が煩雑

お勧めの使い方

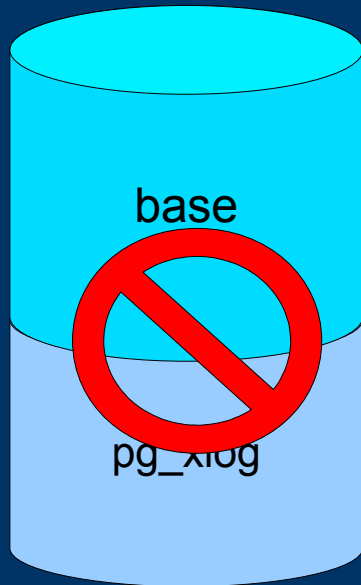
障害復旧時間を最小限におさえたく、
同程度スペックサーバを2台用意できるなら

Warm Standby の仕組み

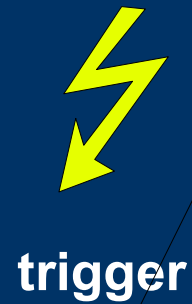
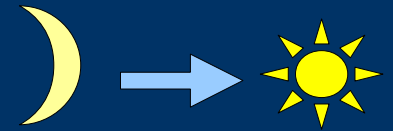
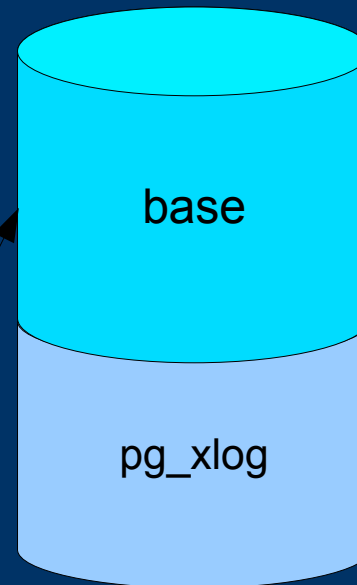


Warm Standby の仕組み

稼動系サーバ



待機系サーバ



Warm Standby 手順

- (主系)設定(postgresql.conf)
archive_mode = on
archive_command = 'scp %p server2:/data/archive/%f'
archive_timeout = 30
- (待機系)アーカイブログ保管ディレクトリ作成
\$ mkdir /data/archive
- (主系)サーバ起動
- (主系)ベースバックアップ取得
PITRと同じ手順。ただし、待機系に保管

Warm Standby 手順

- (待機系)リカバリファイル作成
(recovery.conf)

```
restore_command = 'pg_standby -l -t  
/tmp/pgsql.trigger /data/archive %f %p %r'
```

- (待機系)起動

まとめ

	pg_dump	PITR	Warm Standby
復旧できるところ	・最後のバックアップ取得時間	・直前まで	・直前まで
復旧時間	・かかる	・ややかかる	・かからない
設定の手間	・簡単	・やや簡単	・少し大変