PostgreSQLレプリケーション徹底紹介

2012年10月18日

NTTデータ/JPUG 藤井雅雄

db tech showcase 2012

NTT DATA

謝辞

本資料は、NTT OSSセンタ様の以前の講演資料をベースにしております。ご提供承諾いただきありがとうございます。

目次

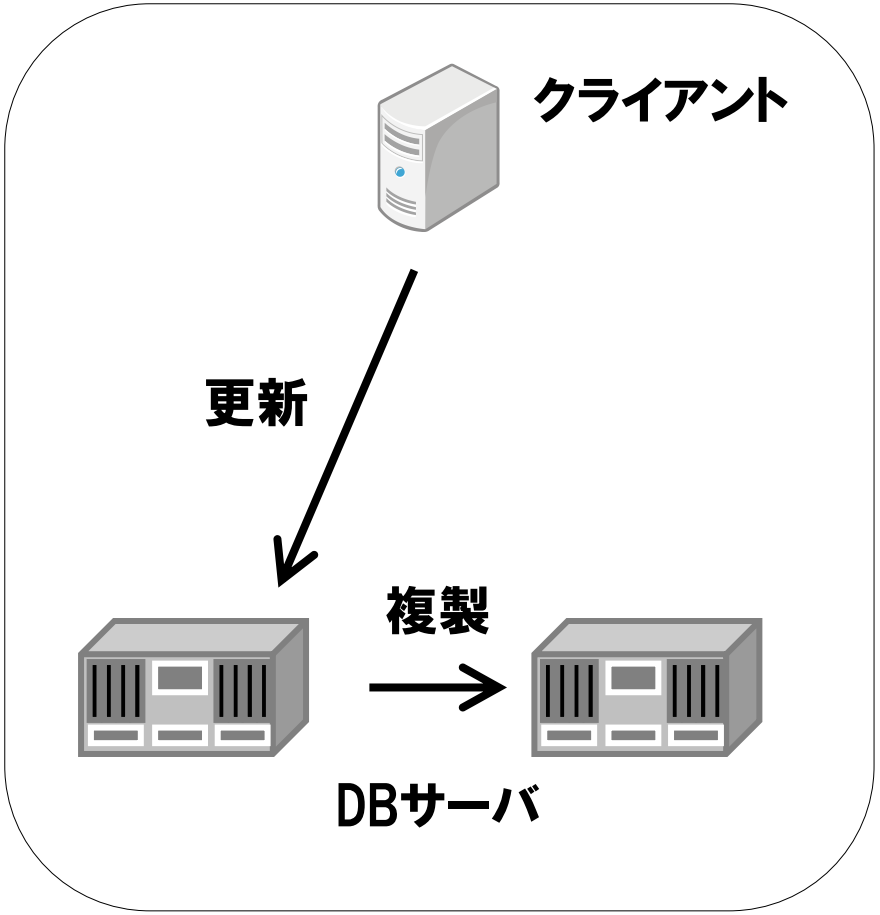
- レプリケーションとは？
- PostgreSQLレプリケーション
  1. 特徴
  2. 同期/非同期
  3. 利用事例

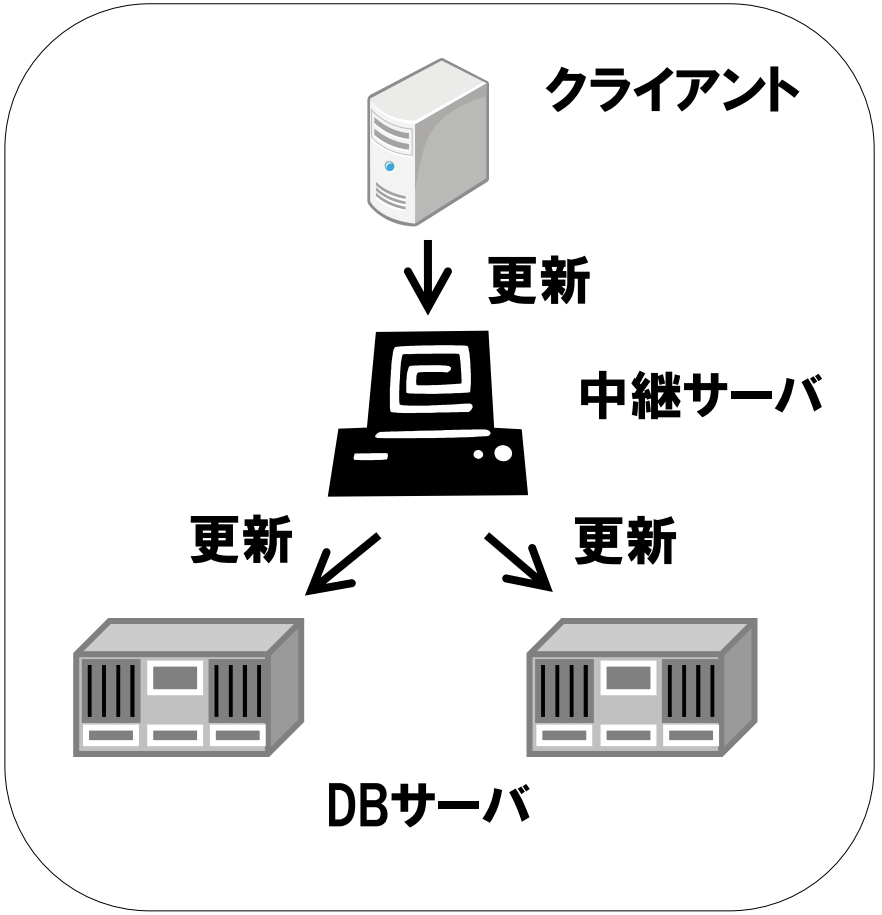
レプリケーションとは？

レプリケーションとは？

NTT DATA

複数のサーバにデータベースを自動的に複製する機能





なぜレプリケーションが必要か？

NTT DATA

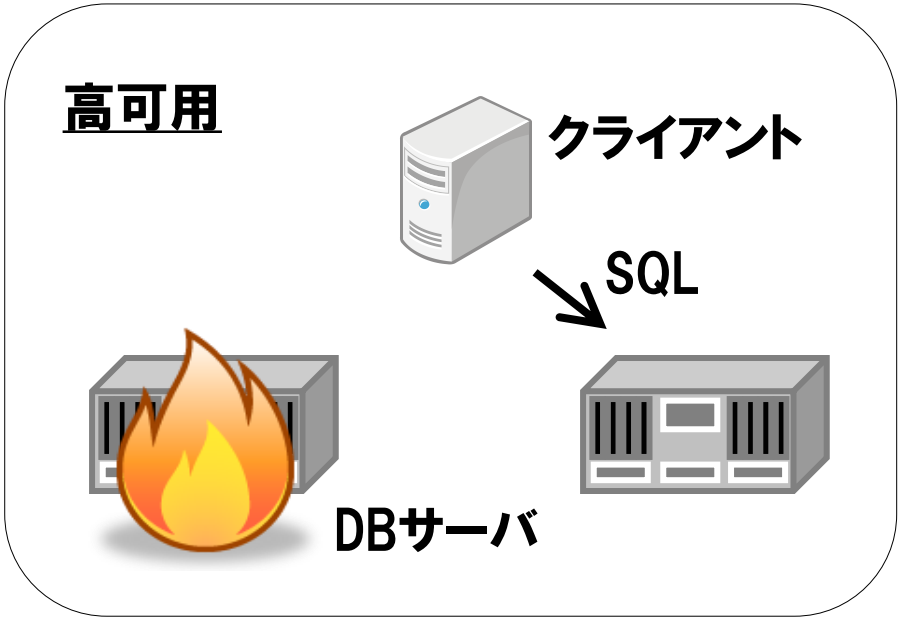
高可用性

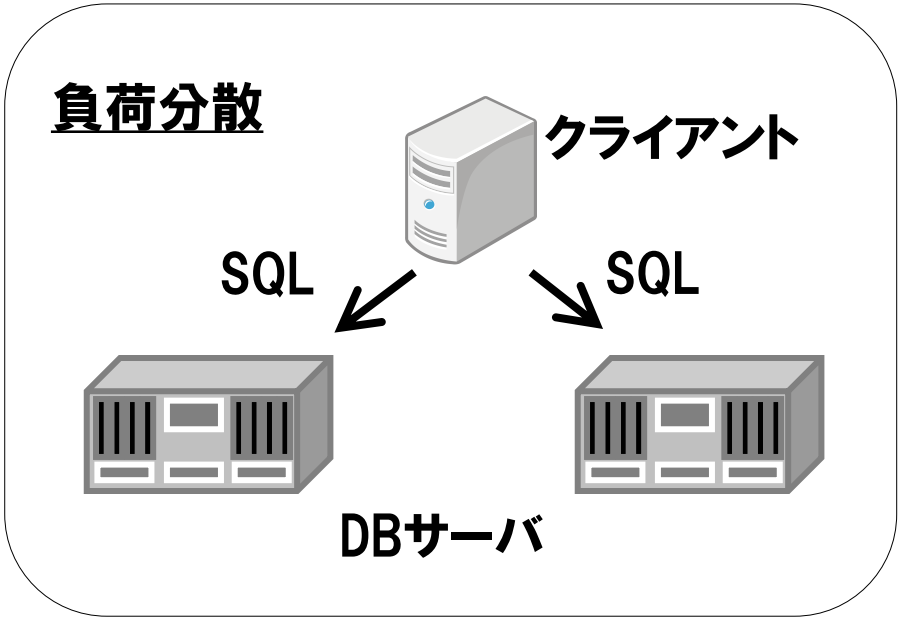
1台が故障しても、別サーバが処理を引き継げる
システム全体としてDBサービスが停止するのを回避できる

負荷分散

SQL実行の負荷を複数のサーバに分散できる
負荷が一箇所に集中しないので、システム全体として性能向上できる

24時間365日システムを安定運用するのに必要!





PostgreSQLレプリケーションの歴史

NTT DATA

当初コミュニティはPostgreSQL本体にレプリケーション機能を組み込まない方針だったが、ユーザの声を受けて、9.0から本体にレプリケーション機能を搭載!
以降、レプリケーションは着実に進化中

レプリケーションツールが乱立!

Bucardo *GridSQL* *Londiste* *Mammoth*
pgpool-II *PGCluster*
PL/Proxy *PostgresForest*
Postgres-R *Postgres-XC*
rubyrep *Sequoia* *Slony-I* *syncreplicator*

2007 2008 2009 2010 2011 2012

9.0 (2010/9リリース)
- 非同期レプリケーション

9.1 (2011/9)
- 同期レプリケーション
- レプリケーション監視機能強化
- 物理バックアップ取得ツール

9.2 (2012/9)
- カスケードレプリケーション
- スタンバイからの物理バックアップ取得
- 同期モードの拡張

PostgreSQLレプリケーションの特徴

シングルマスタ/マルチスタンバイ構成

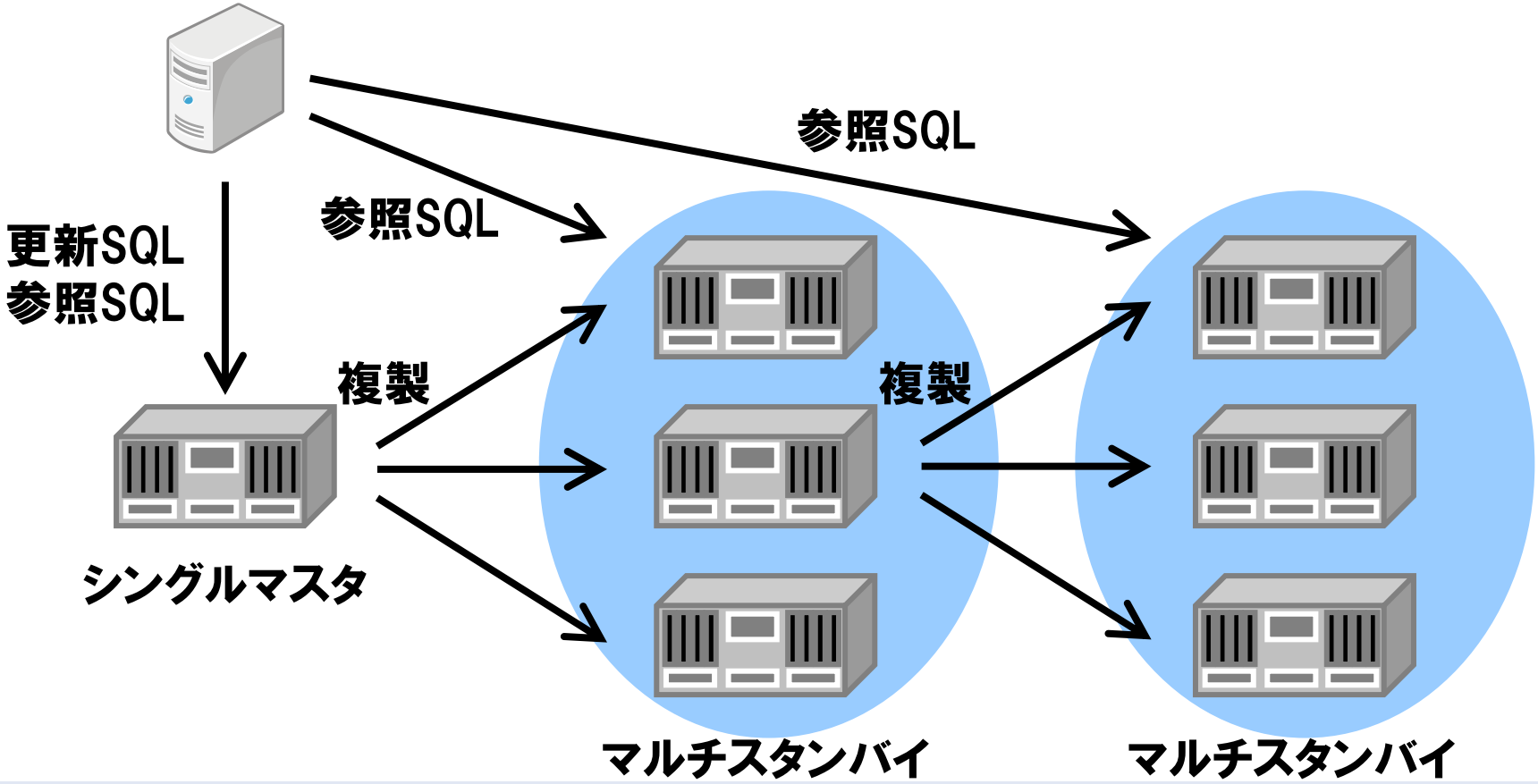
NTT DATA

マスタ1台からスタンバイ複数台へのレプリケーション

カスケードレプリケーション

マスタは更新と参照SQL両方、スタンバイは参照SQLのみ実行可能

➔参照系処理のスケールアウトに利用可能



スタンバイで実行可能/不可能なSQL

NTT DATA

実行可能

クエリ・アクセス

- SELECT
- PREPARE, EXECUTE
- カーソル操作

オンライン・バックアップ

- （論理）pg_dump
- （物理）pg_basebackup

実行不可能

データ操作言語 (DML)

- INSERT, UPDATE, DELETE
- SELECT FOR UPDATE

データ定義言語 (DDL)

- CREATE, DROP, ALTER

一時テーブル

メンテナンス・コマンド

- VACUUM, ANALYZE

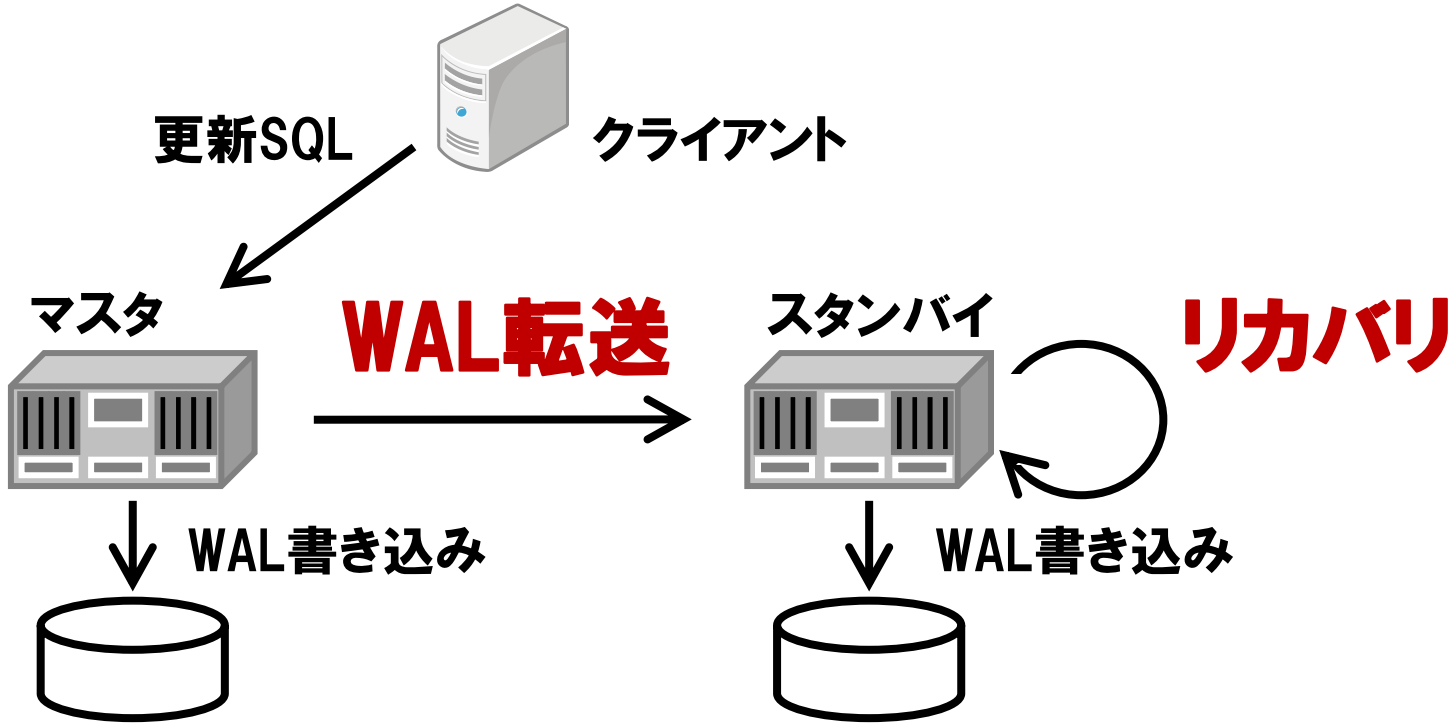
※マスタからメンテナンスの実行結果が複製されるため、スタンバイでは実行不要

ログシッピング

マスタからスタンバイにトランザクションログ（WAL）を転送

スタンバイはリカバリモード

転送されたWALをリカバリすることで、スタンバイはデータベースを複製



ログシッピングによる制約

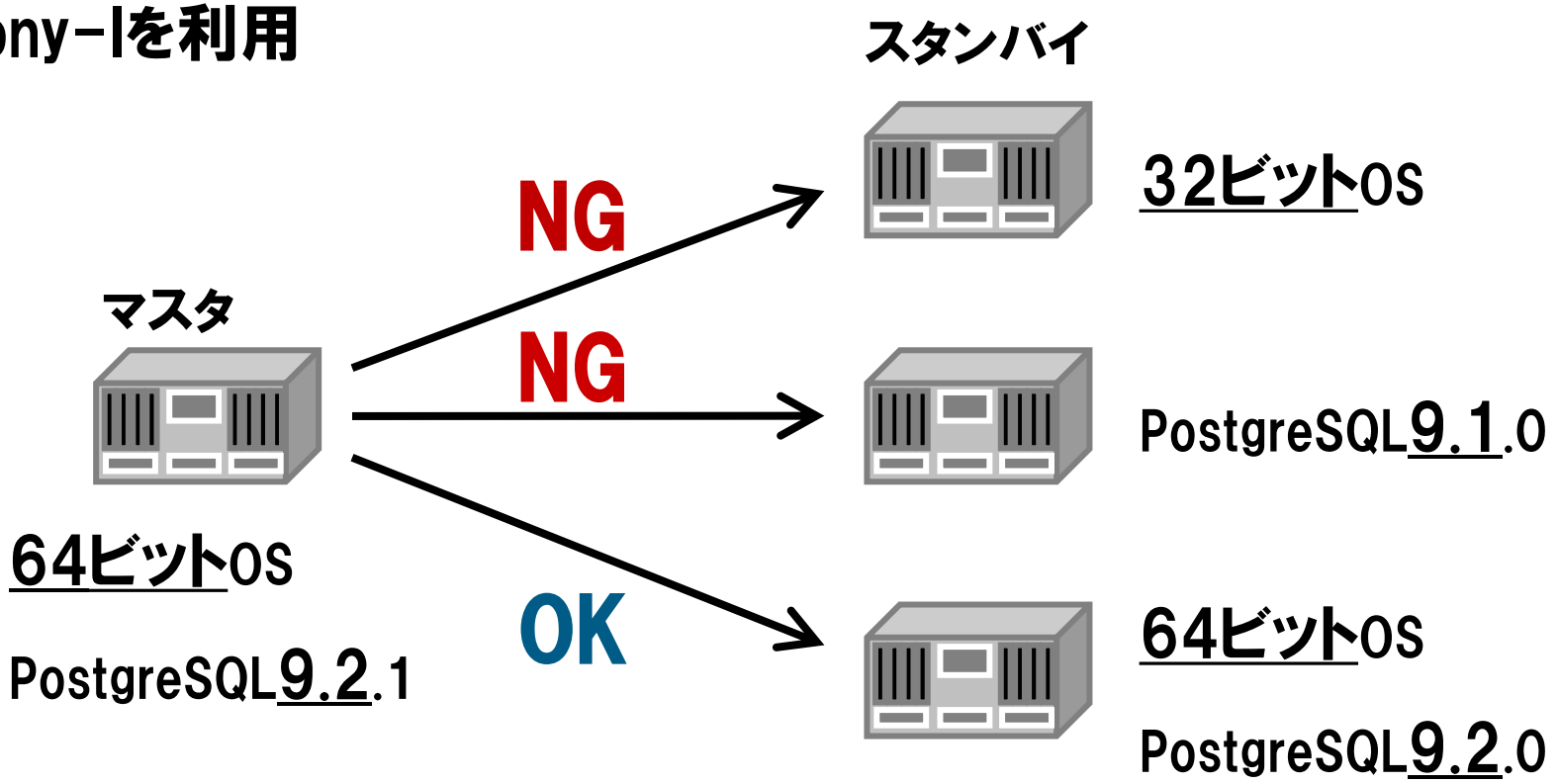
NTT DATA

マスタとスタンバイでは以下2点が同じでなければならない

① ハードウェアとOSのアーキテクチャ

② PostgreSQLのメジャーバージョン

➔ 異なる環境間のレプリケーション（ローリングアップグレード）には Slony-Iを利用



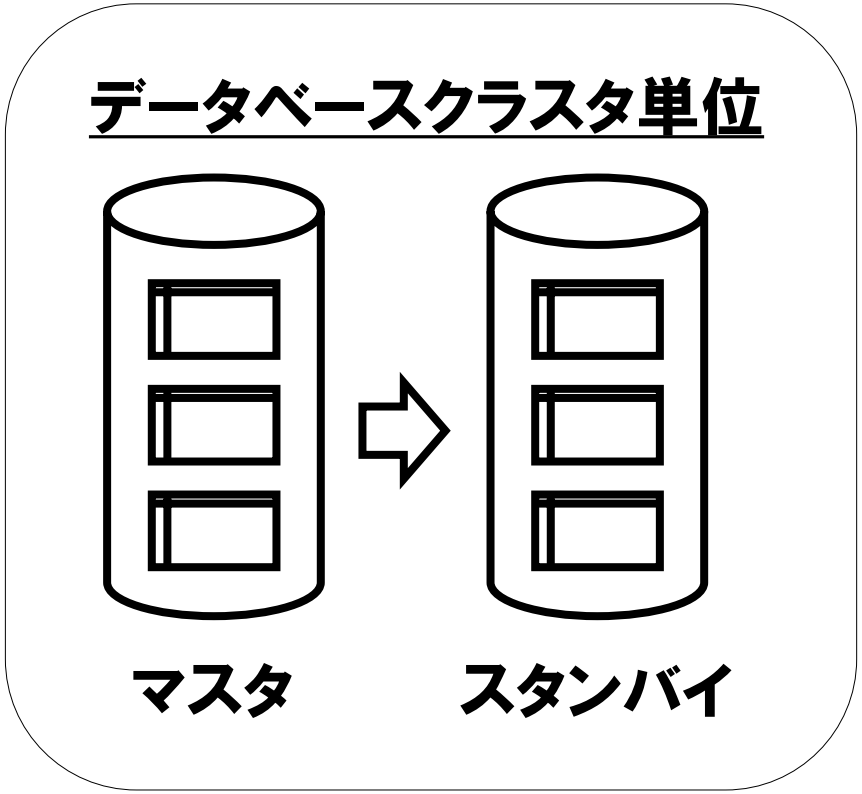
データベースクラスタ単位のレプリケーション

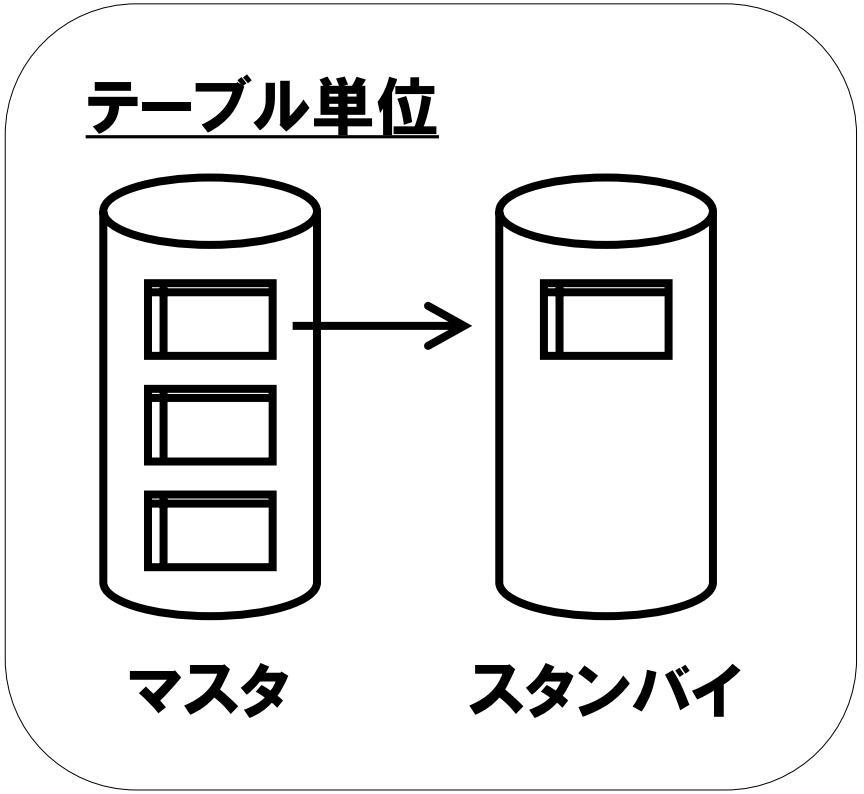
NTT DATA

すべてのデータベースオブジェクトが基本的にレプリケーション対象

- テーブル単位のレプリケーション指定は不可

➔テーブル単位のレプリケーションにはSlony-Iを利用





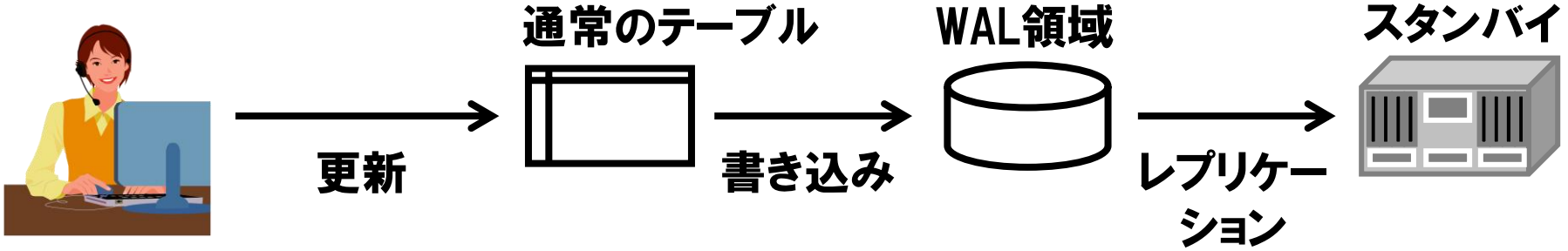
(参考) UNLOGGED TABLE

NTT DATA

WALを書かないテーブルを作成可能!

- レプリケーション対象にならない
- 更新性能が著しく向上
- クラッシュ時にテーブルが空になる

➔信頼性より性能を必要とするテーブルに有効



レプリケーション構成への移行が容易

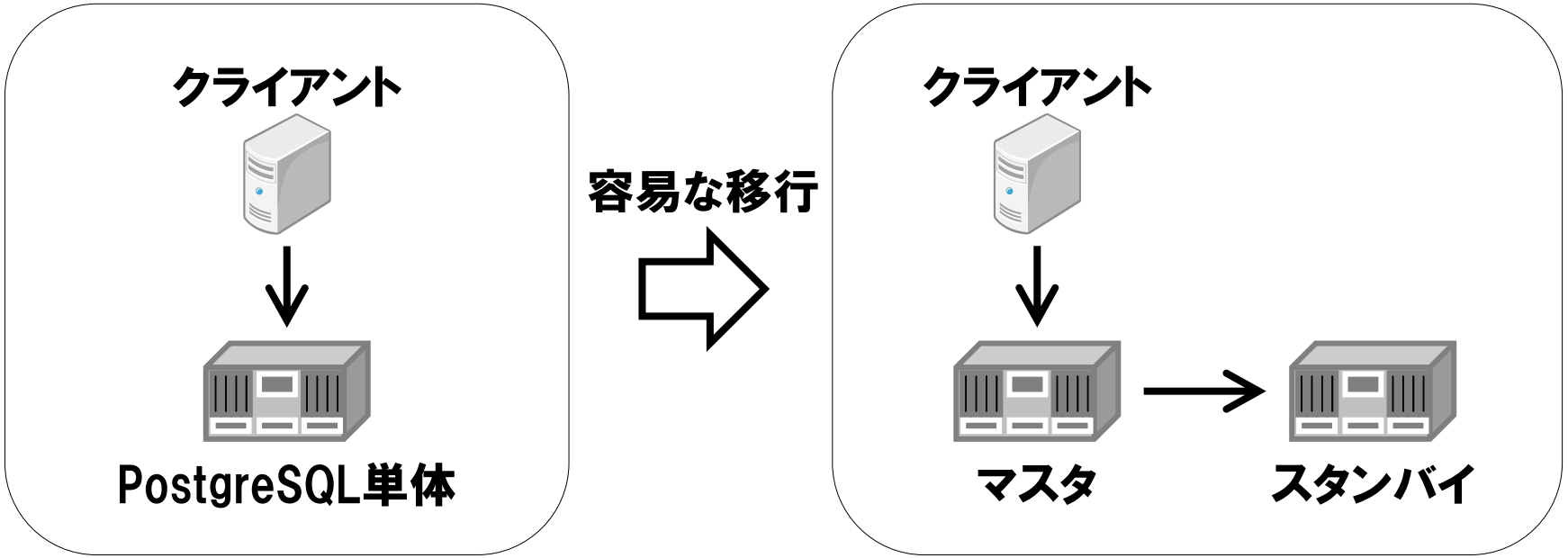
NTT DATA

テーブル定義の変更不要

- 例）テーブルにプライマリキーを定義する必要がない

SQLの書き換え不要

- 例）実行するまで結果が確定しないSQLを矛盾なく実行可能
- PostgreSQLがサポートするすべてのSQLをマスタで実行可能

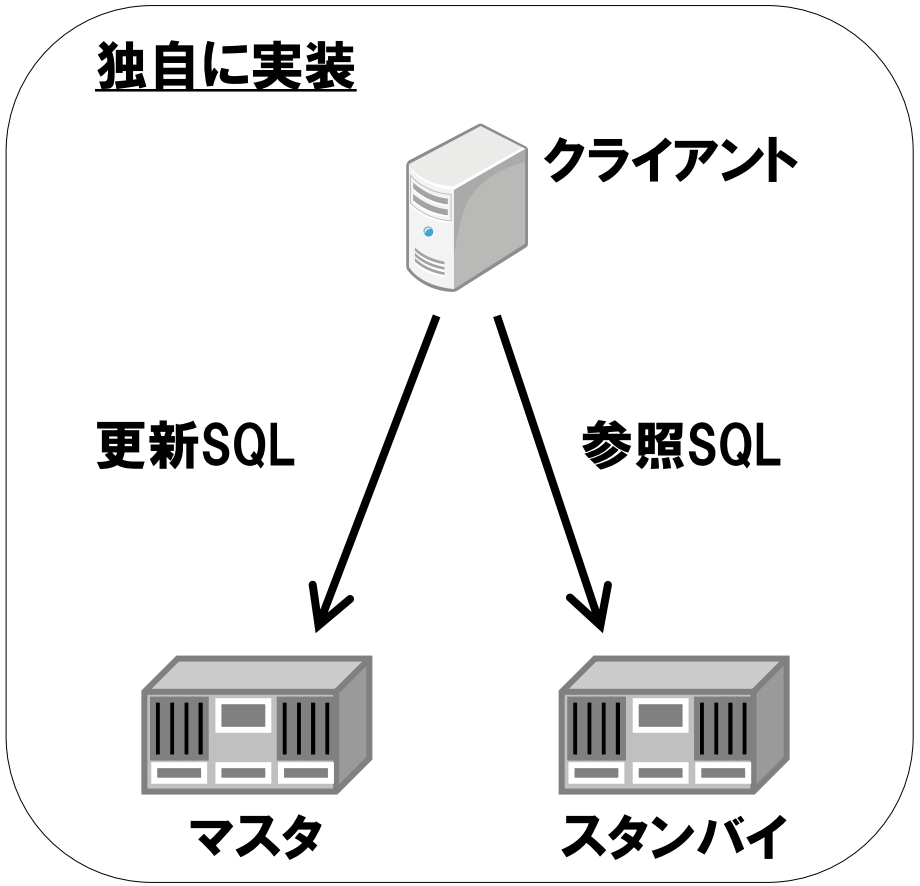


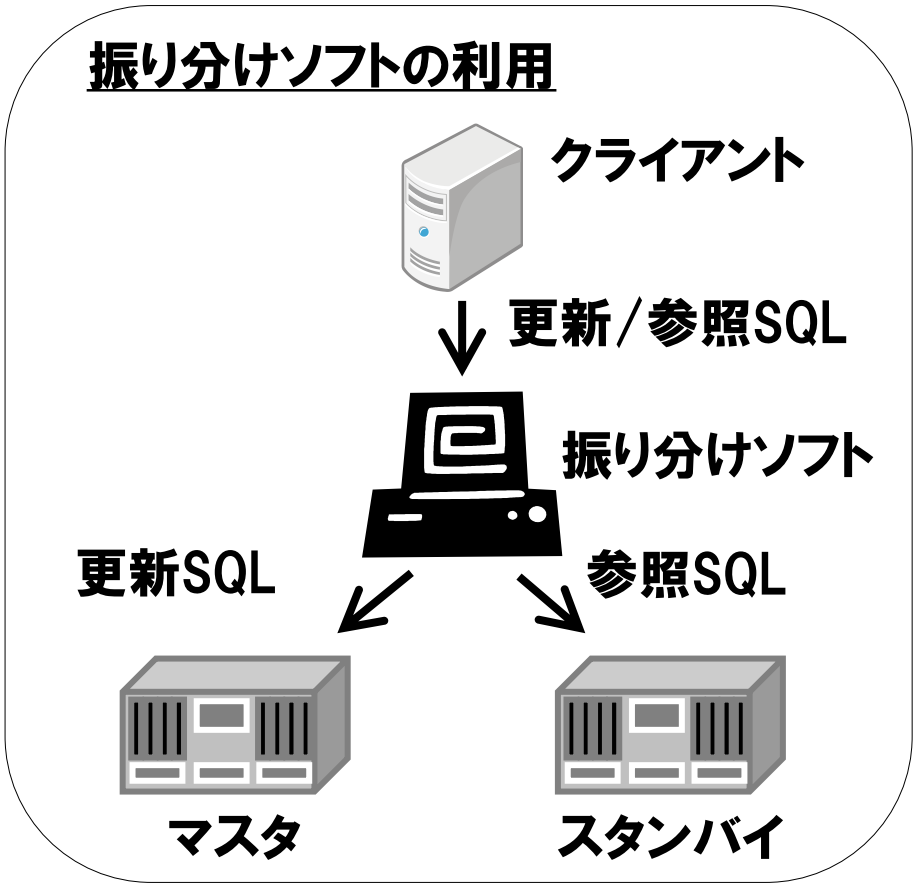
SQLの振り分け

NTT DATA

PostgreSQLはSQLの振り分け機能を提供しない

- クライアント側で振り分けを独自実装
- 振り分けを行うソフトウェア（pgpool-II）を利用





（参考）pgpool-IIによるSQLの振り分け

NTT DATA

スタンバイで実行できない
参照SQLはマスタに振り分け

トランザクション内の参照SQLも
スタンバイに振り分け

BEGIN	→	マスタ
SELECT	→	スタンバイ
SELECT（一時テーブル）	→	マスタ
SELECT FOR UPDATE	→	マスタ
UPDATE	→	マスタ
SELECT	→	マスタ
COMMIT	→	マスタ

スタンバイの遅れが閾値を
超えていたら、マスタに振り分け

更新SQL後の参照SQLは
マスタに振り分け。
更新SQLの実行結果を、
参照SQLがすぐに参照する
必要があるため

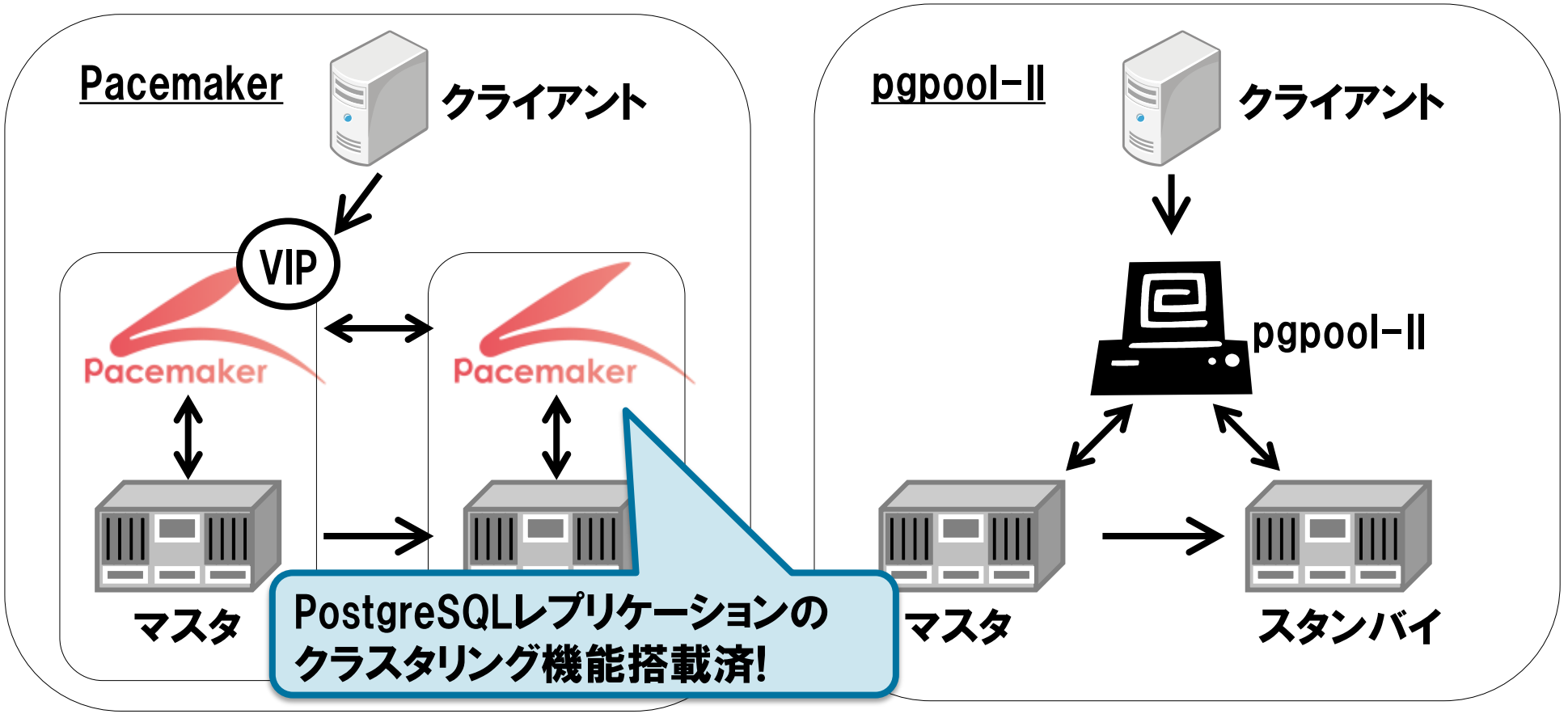
更新SQLはマスタに振り分け

フェイルオーバ

NTT DATA

PostgreSQLは自動的なフェイルオーバ機能を提供しない

- スタンバイはいつでもマスタに昇格可能 (pg_ctl promote)
- 自動的な故障検知とフェイルオーバにはクラスタソフトと要連携



レプリケーションの監視

NTT DATA

```
=# SELECT * FROM pg_stat_replication;
-[ RECORD 1 ]----+------------------------------
procpid          | 26531
usesysid         | 10
usename          | postgres
application_name | tokyo
client_addr      | 192.168.1.2
client_hostname  |
client_port      | 39654
backend_start    | 2012-02-01 18:54:49.429459+09
state            | streaming
sent_location    | 0/406E7CC
write_location   | 0/406E7CC
flush_location   | 0/406E7CC
replay_location  | 0/406E1B0
sync_priority    | 1
sync_state       | sync
```

レプリケーション接続情報
スタンバイのIPアドレス、ポート番号、
レプリケーションに使用するユーザ名、
レプリケーションの開始日時など

レプリケーションの進捗
マスタはどこまでWALを送信したか?
スタンバイがどこまでWALを
書き込み/フラッシュ/リカバリしたか?

レプリケーションの状態
どの同期モードで動作中か?
スタンバイはマスタに追いつき中か?済か?

その他

NTT DATA

① クラッシュセーフなマスタとスタンバイ

クラッシュしたマスタ/スタンバイを再起動するだけでレプリケーション再開可能

② スタンバイのオンライン追加・削除

マスタ稼働中にスタンバイを追加・削除可能

スケールアウト時にサービスの一時停止が発生しない

③ リカバリの一時停止・再開

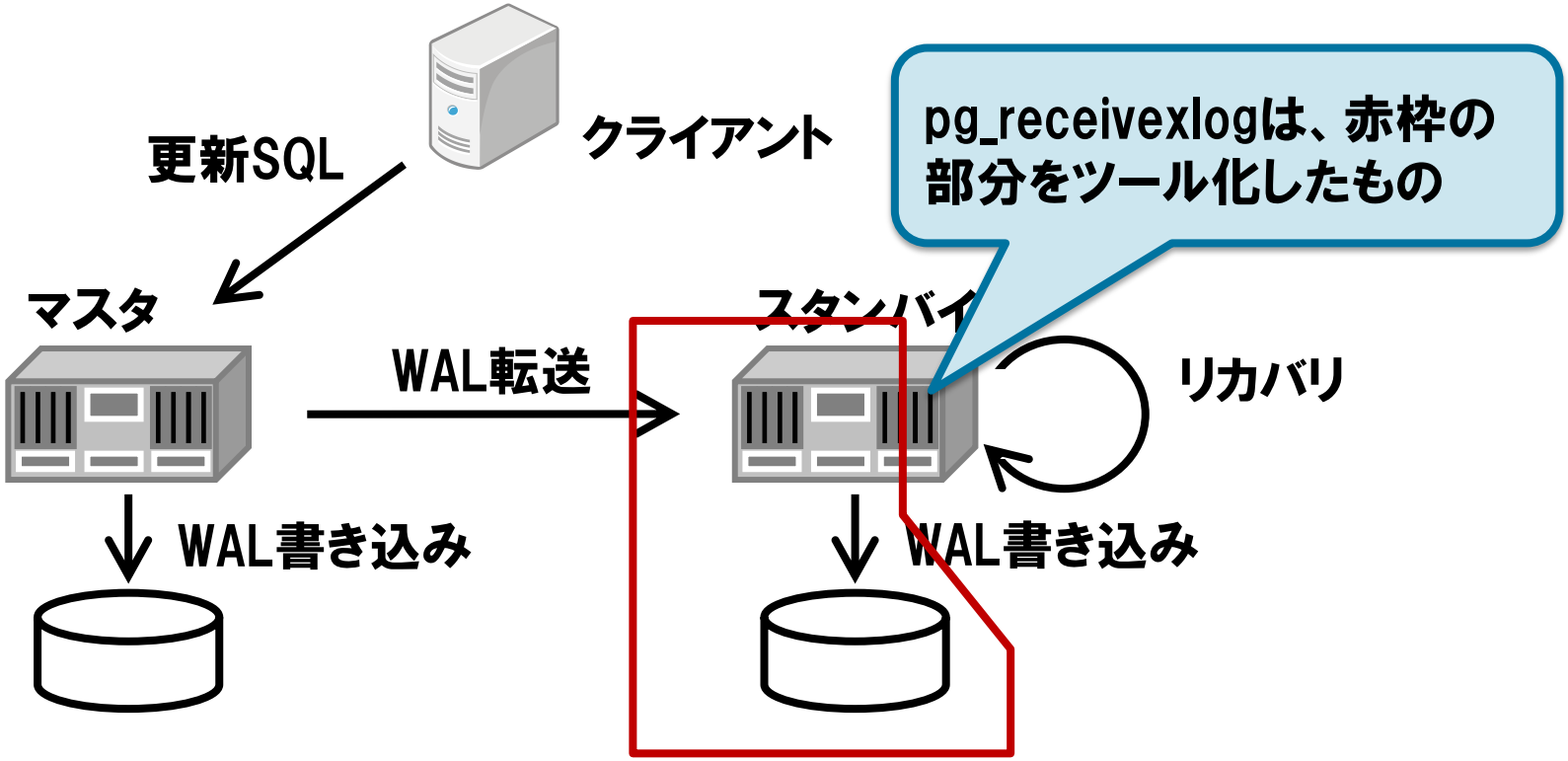
スタンバイ側でリカバリを一時停止し、静止点を作成可能

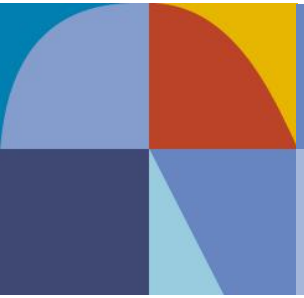
(参考) pg_receivexlog

NTT DATA

WALの受信と書き込みを繰り返すクライアントツール

- オペミス対策でWALの多重化などの用途





PostgreSQLレプリケーションの同期/非同期

レプリケーションモード

NTT DATA

レプリケーションモードを選択可能

- 非同期
- 同期

非同期レプリケーション

NTT DATA

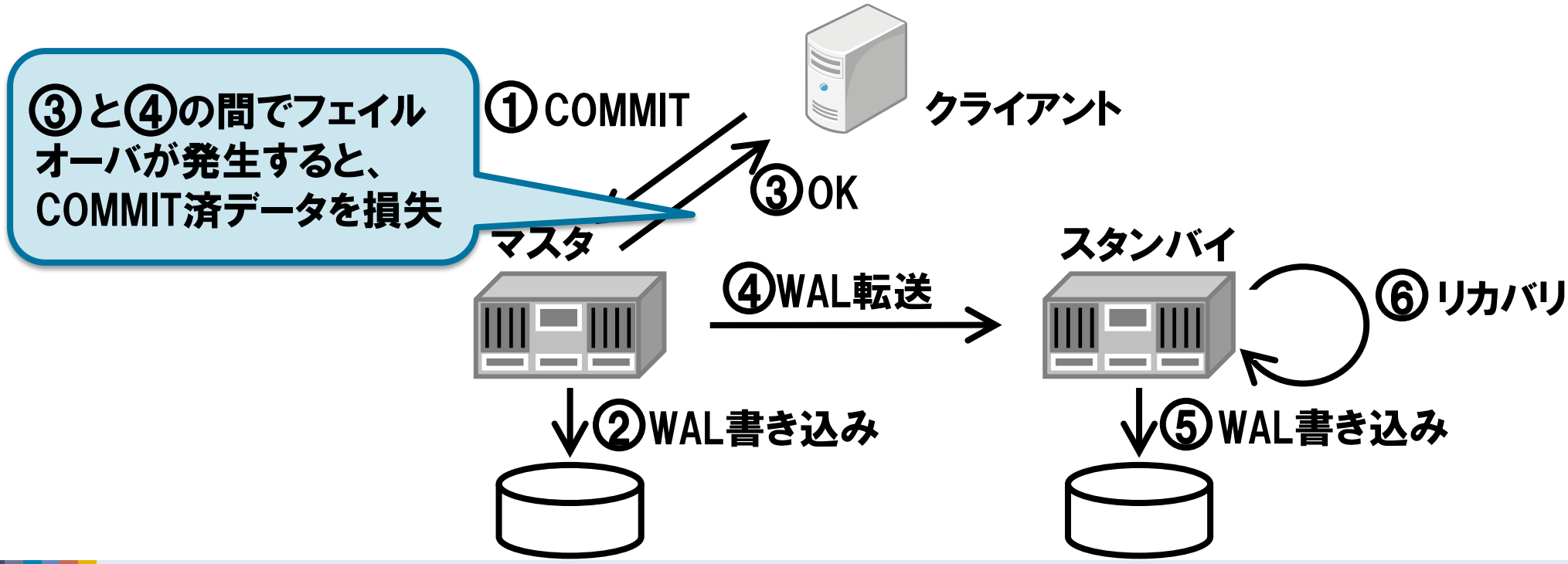
COMMIT時にレプリケーションの完了を待たない

- COMMIT成功時にWALがスタンバイに届いている保証なし

☹ フェイルオーバ時にCOMMIT済データを失う可能性あり

☹ スタンバイの参照SQLで古いデータが見える可能性あり

👍 レプリケーション完了を待たないので比較的高性能!



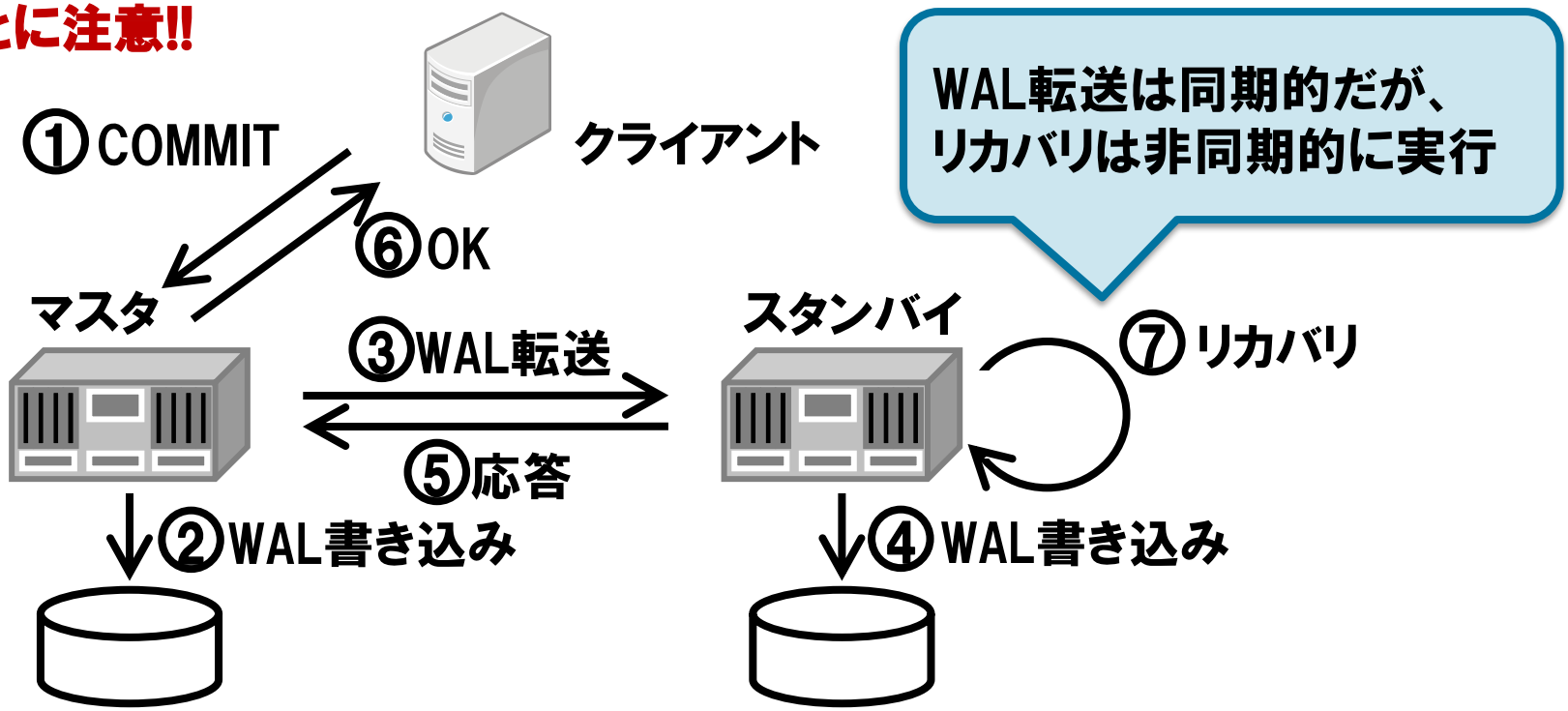
同期レプリケーション

NTT DATA

COMMIT時にレプリケーションの完了を待つ

- COMMIT成功時にWALがマスタ・スタンバイ両方に書き込み済と保証
- フェイルオーバ時にCOMMIT済データを失わない!
- レプリケーション完了を待つので比較的低性能
- スタンバイの参照SQLで古いデータが見える可能性あり

➔「同期レプリケーション＝COMMIT済データをすぐにスタンバイで参照可能」ではないことに注意!!

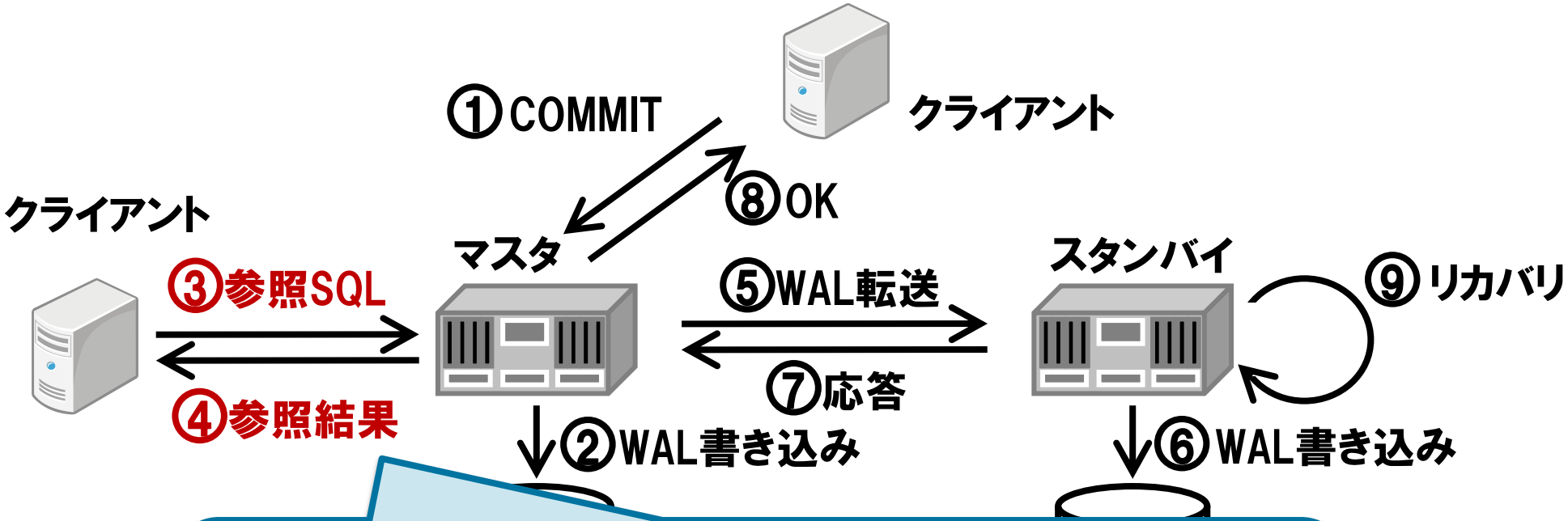


同期レプリケーション

NTT DATA

応答が届くまでトランザクションの結果は別トランザクションから参照不可

- 参照を許すと、未COMMITのデータが参照される可能性がある



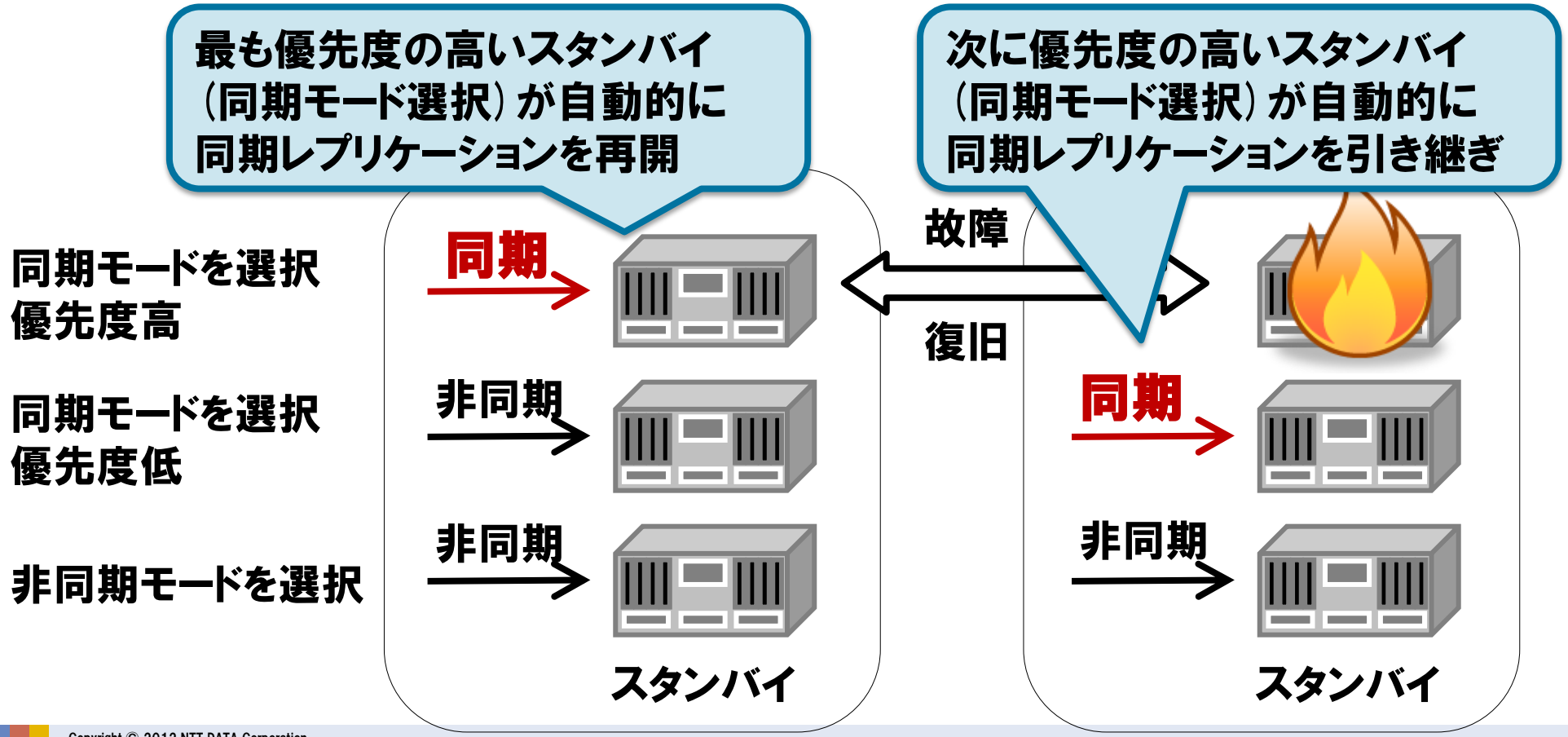
応答待ちのトランザクションの結果を④が参照した場合
④と⑤の間でフェイルオーバが発生すると、
応答待ちのトランザクションの結果はスタンバイに存在しない
④では未COMMITのデータを参照したことになる

スタンバイごとのレプリケーションモード選択

NTT DATA

スタンバイごとにレプリケーションモードを選択可能

- 同期レプリケーションを実行できるスタンバイは同時に1台のみ
- 同期レプリケーションの実行優先度をスタンバイに設定可能
- カスケードレプリケーションは非同期モードのみ選択可能

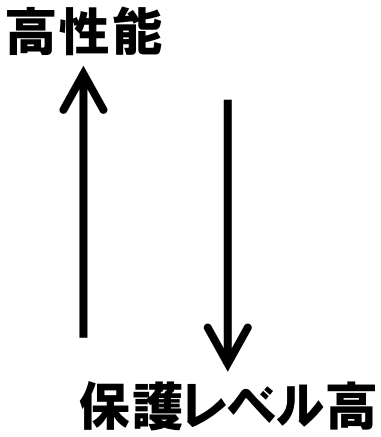


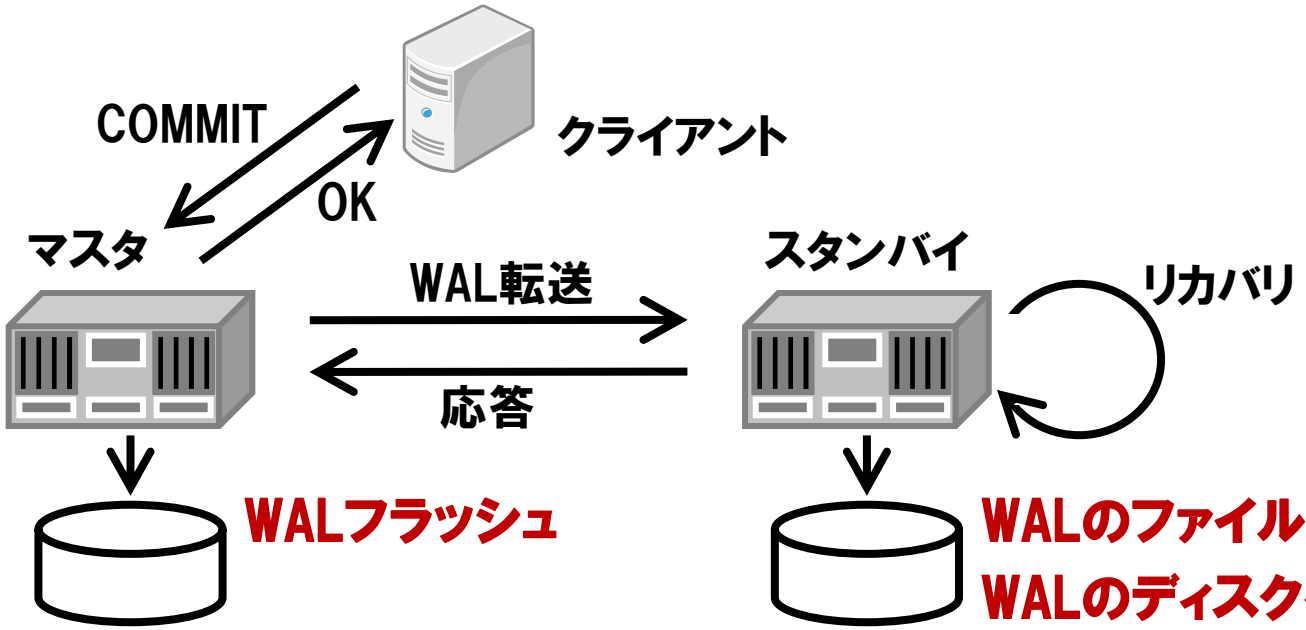
トランザクションごとのデータ保護レベル選択

NTT DATA

同期レプリケーションでは、トランザクションごとにデータ保護レベルを選択可能

データ保護レベル	マスタ	スタンバイ	
	WALフラッシュ	WAL書き込み	WALフラッシュ
off	待たない	待たない	待たない
local	待つ	待たない	待たない
remote_write	待つ	待つ	待たない
on	待つ	待つ	待つ





WALのファイルキャッシュへの書き込み
WALのディスクへのフラッシュ

非同期レプリケーション継続不可時の挙動

NTT DATA

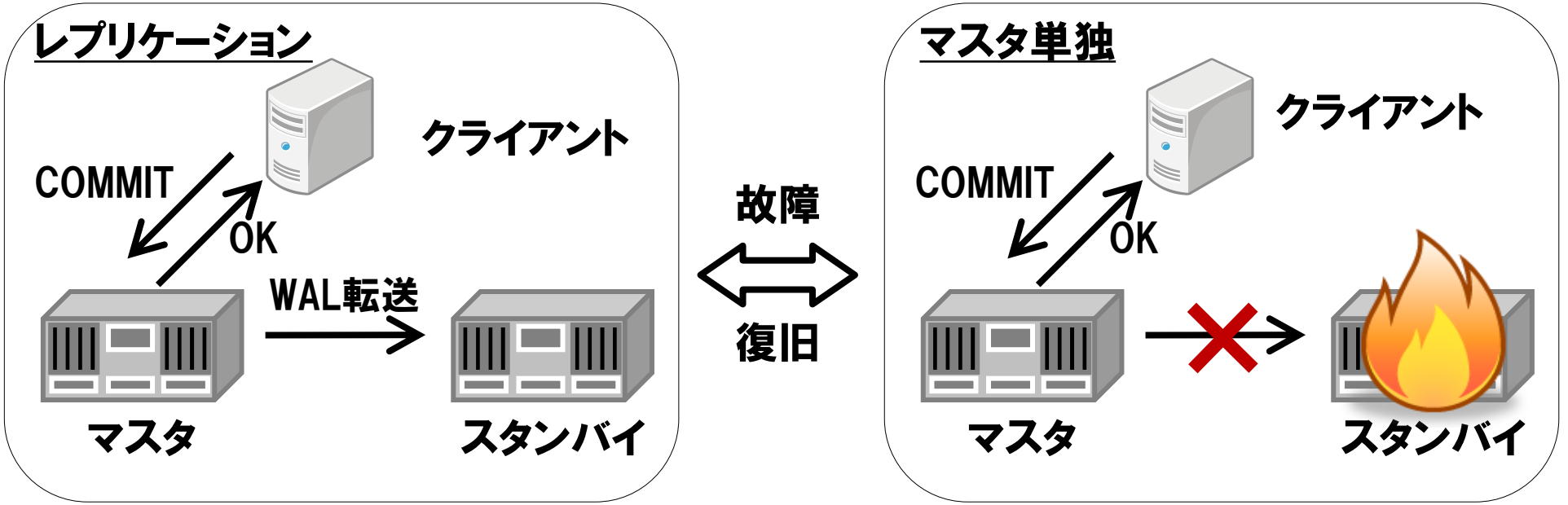
故障により単独稼働するマスタの挙動に注意!

- マスタ故障 (フェイルオーバ) → 新マスタ単独稼働
- スタンバイ故障
- ネットワーク故障

→ 既存マスタ単独稼働

マスタ単独でトランザクションを処理

- 故障によりトランザクションが停止することはない



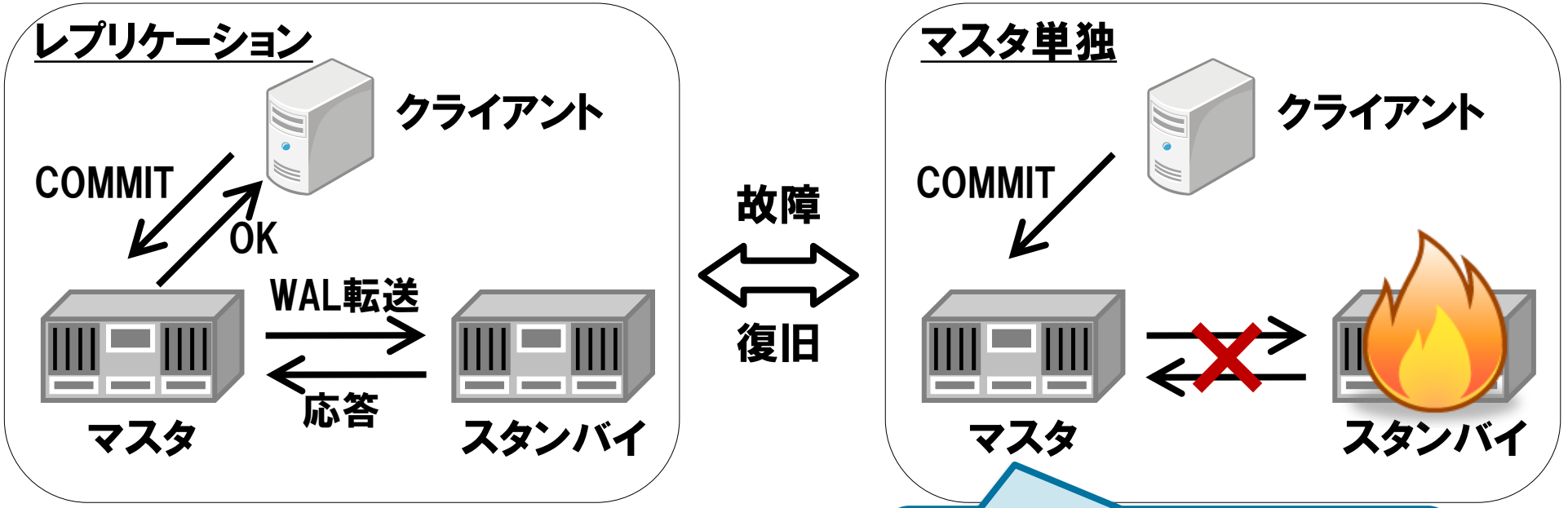
同期レプリケーション継続不可時の挙動

NTT DATA

COMMIT時に（スタンバイからの届くことのない）応答を待ち続ける

- 故障によりトランザクションは停止する!
- 復旧により応答が届くようになったら、トランザクション再開

➔ 復旧にかかる時間だけシステム停止

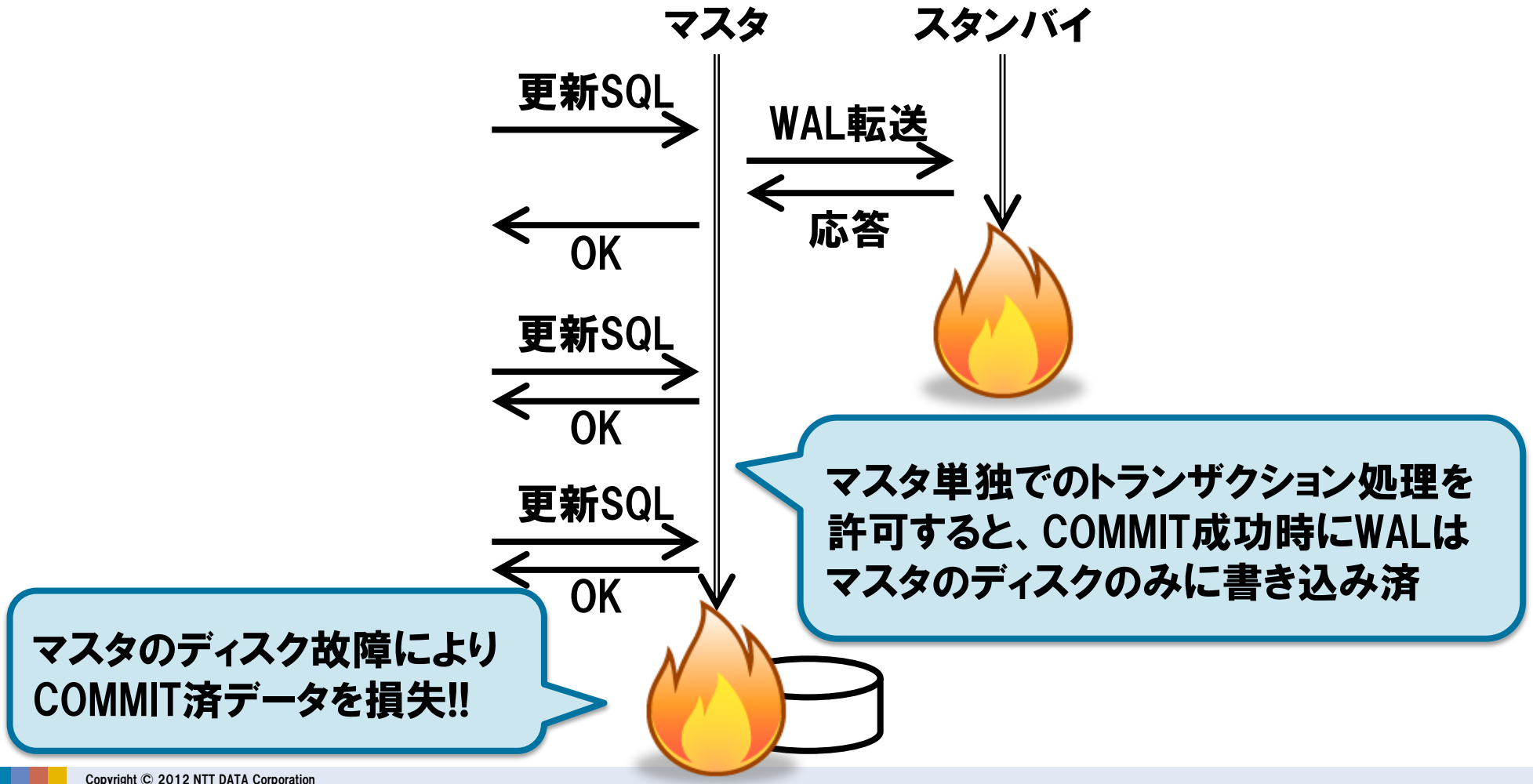


同期レプリケーション継続不可時の挙動

NTT DATA

トランザクション停止は、データ保護を最優先するため

- COMMIT成功時に確実にWALが複数箇所に書き込み済と保証

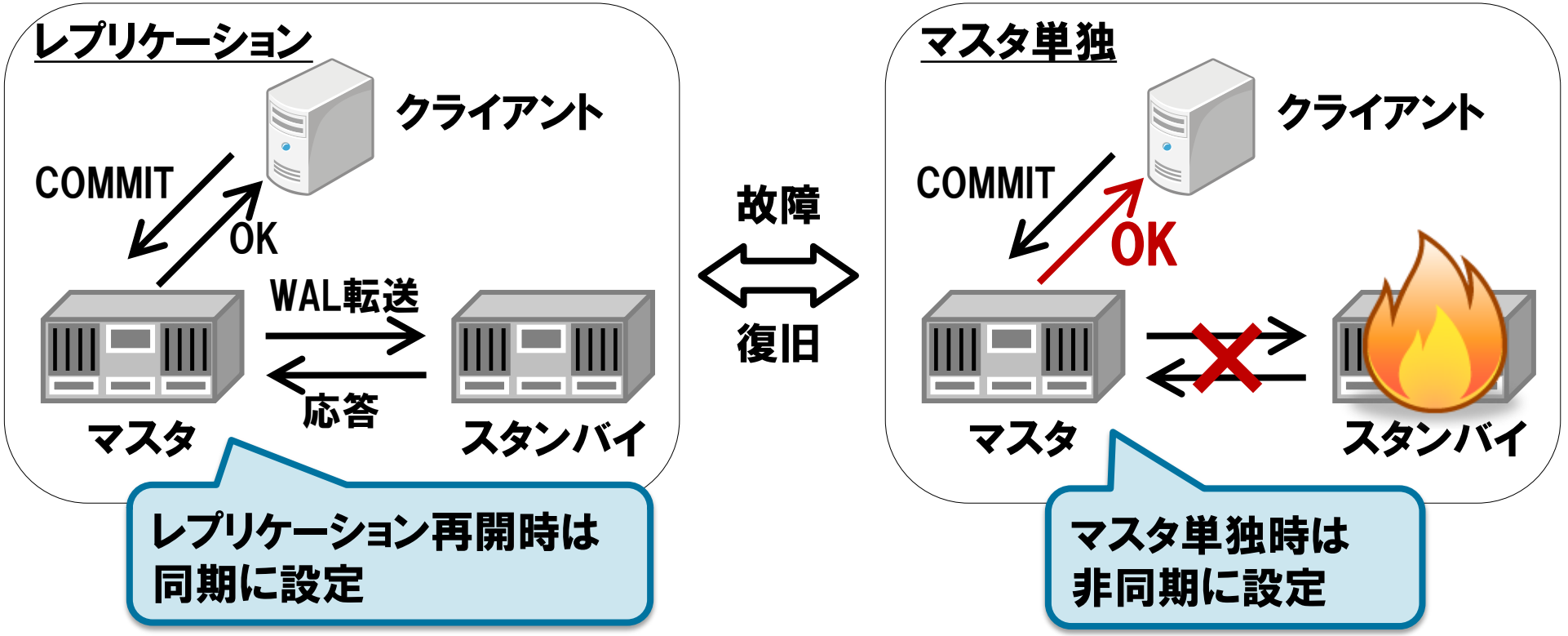


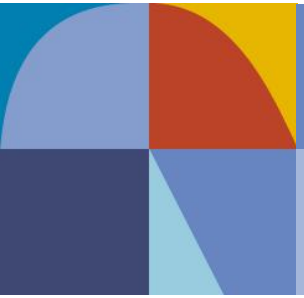
同期レプリケーションでの高可用の実現

NTT DATA

① マルチスタンバイ構成

② 故障/復旧時にレプリケーションモードを非同期/同期に設定変更
データ損失リスクは依然あるため、(RAID等)ディスク故障対策が必要
状況に応じてモードを設定変更する機能をPacemakerは搭載済!





PostgreSQLレプリケーションの利用事例

利用事例：Instagram

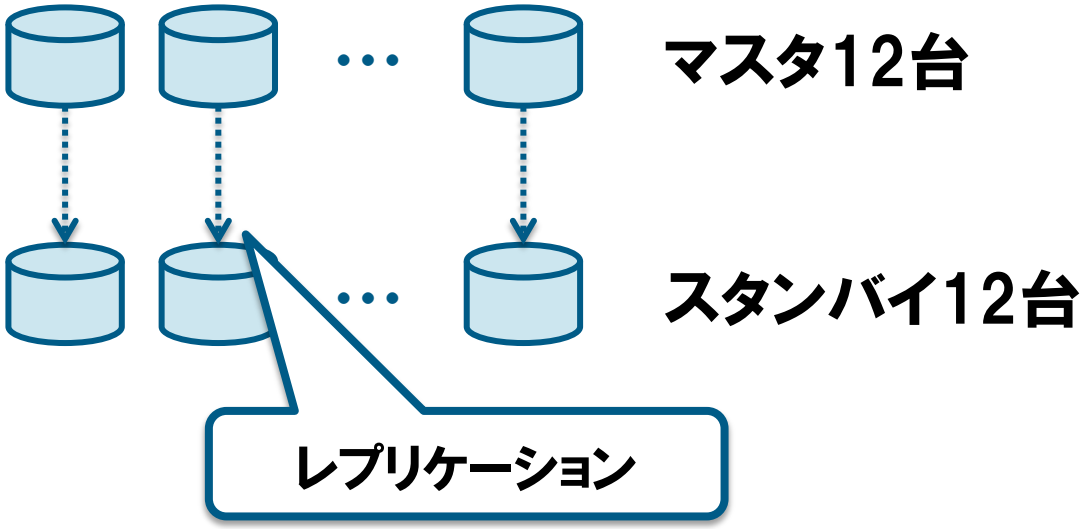
NTT DATA

写真共有アプリ/SNS。Facebookが10億ドルで買収

利用者4000万人超の写真データを保管

データベースをマスタ12台に水平分散

マスタ12台、スタンバイ12台でレプリケーション

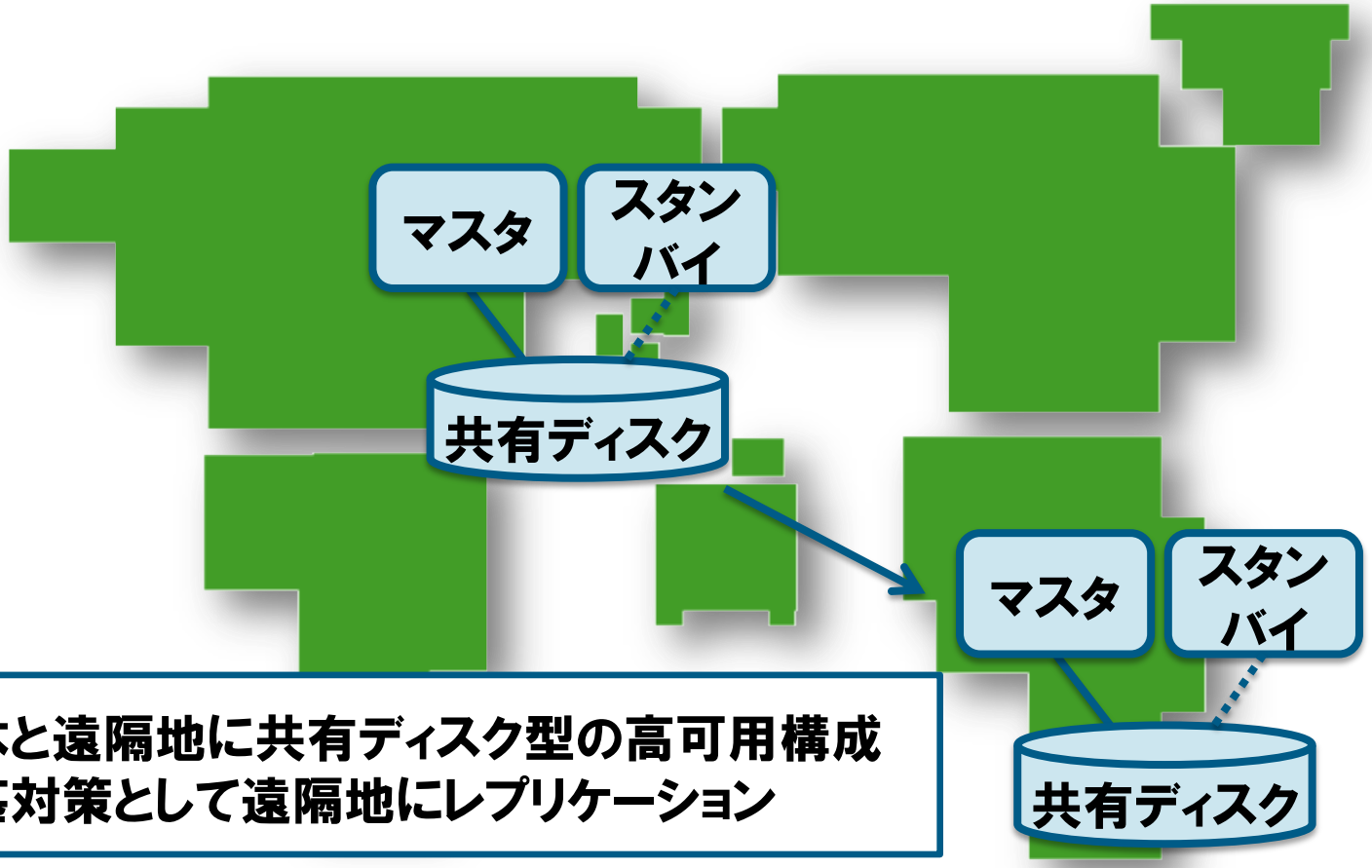


（出典：Scaling Instagram @ AirBnB Tech Talk 2012）

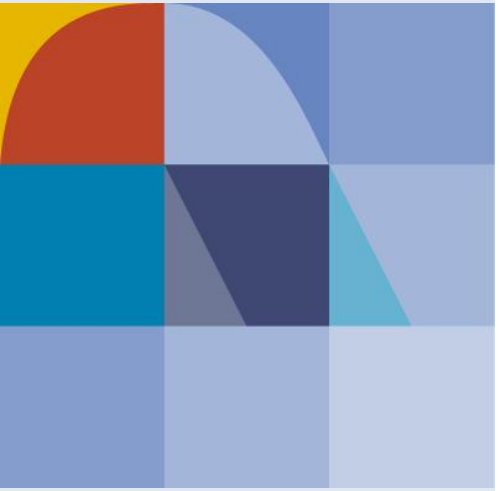
レプリケーション構成事例

NTT DATA

共有ディスク型とレプリケーションの組み合わせ

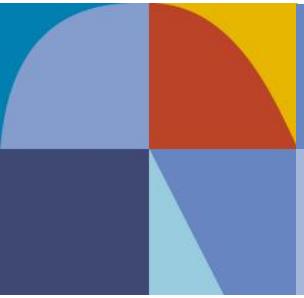


- 日本と遠隔地に共有ディスク型の高可用構成
- 激甚対策として遠隔地にレプリケーション



NTT DATA

変える力を、ともに生み出す。



PostgreSQLレプリケーションのアーキテクチャ

プロセス構成