PostgreSQL 16
Support load balancing in libpq

PostgreSQL Unconferecnce #41 (2023-04-24)

自己紹介

- ぬこ@横浜，@nuko_yokohama
- にゃーん
- 趣味でポスグレをやってる者だ
- 要はバランスおじさん

libpq の改造項目

Support load balancing in libpq

- Commitfest 2023-03 で commit された改善項目。
  - https://commitfest.postgresql.org/42/3679/
- libpq にロードバランス機能が追加された！
- host のリスト指定と組み合わせて使う。

Support load balancing in libpq

- 接続文字列にロードバランスを行うオプション load_balance_hosts が追加された。

設定値	意味
any	デフォルト値（ PostgreSQL 15 までの挙動も同様） ホスト間のロードバランシングは行われない。ホストは提供された順に試行され、アドレスは DNS または hosts ファイルから受信した順に試行される。
random	ホストまたはアドレスは、ランダムな順序で試行される。この方法で、複数の PostgreSQL サーバに接続を負荷分散することができる。

環境変数指定

- 環境変数でも同等の制御が可能となった。
- 環境変数名： PGLOADBALANCEHOSTS
  - 2023-04-21 commit ： 0a16512d40a58c5046c2ab4ca7eabb8393f31c18

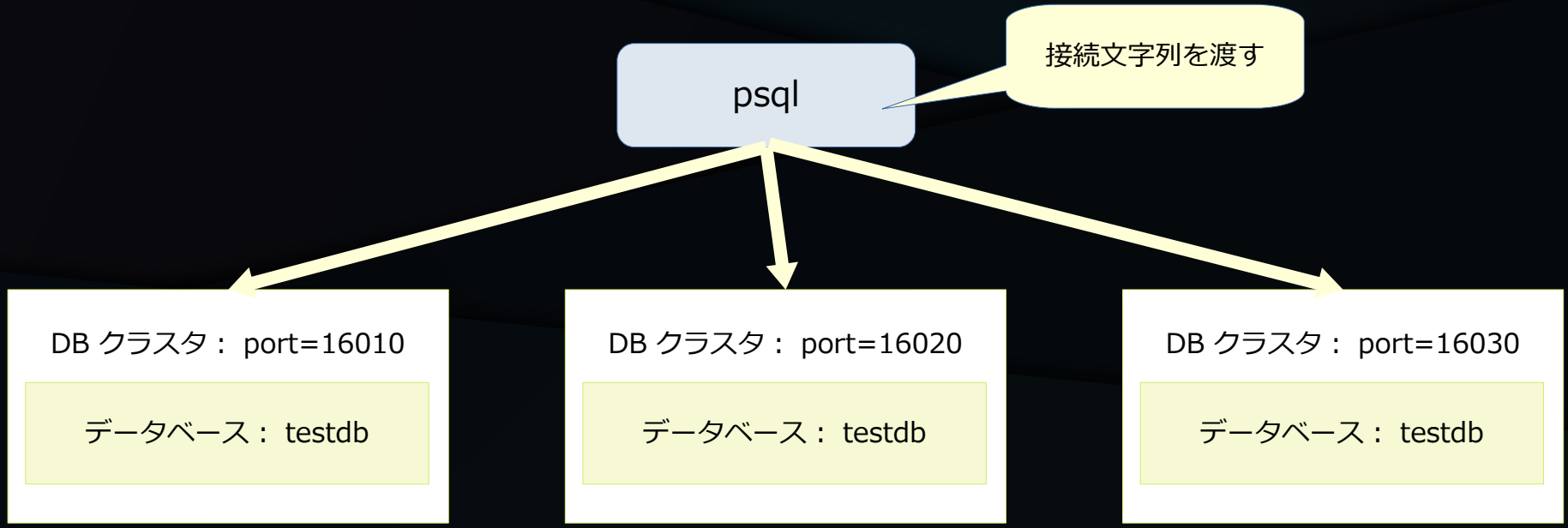
libpq への改造が意味するもの

- libpq を使うアプリケーションも、この改造の恩恵を受ける
  - psql, pgbench 、その他の libpq アプリケーション
  - libpq の接続関数に接続文字列を渡すようにしていれば、今回のロードバランス機能向けの改造はアプリケーション側に不要（なはず）
  - pg_dump, pg_basebackup 等のアプリケーションでも使えるかもしれないが、あまり意味はなさそう

psql での実行例

psql での実行例

- 3 つのデータベースクラスタに対して psql で接続する例



psql に渡す接続文字列

- psql を実行するときに、 -h, -p -d 等のオプションを与えず、直接接続文字列を渡すことができる
- 前スライドのように 3 つのノードにランダムに接続する場合、以下の接続文字列を渡す

```
psql 'host=localhost,localhost,localhost port=16010,16020,16030
load_balance_hosts=random dbname=testdb'
 -c "SHOW port"
```

psql に渡す接続文字列

- 前スライドの接続文字列の説明

キーワード	設定値	意味
host	localhost,localhost,localhost	今回は 3 ノードとも同一 EC2 上に作っているので、全て同じ localhost を指定する。
port	16010,16020,16030	今回は 3 ノードを別ポートに設定している。 host の数とカンマリストを合わせる必要がある。
load_balance_host	random	random にするとロードバランスする
dbname	testdb	3 つとも同じ DB 名なので設定は一つで OK

psql でのロードバランス実行例

- psql を実行するとカンマリスト中の host, port, dbname の組のどれかに接続し、 port 番号を SHOW で表示する。

```
$ psql 'host=localhost,localhost,localhost port=16010,16020,16030 load_balance_hosts=random dbname=testdb' -
c "SHOW port"
 port
-------
 16010
(1 row)

$ psql 'host=localhost,localhost,localhost port=16010,16020,16030 load_balance_hosts=random dbname=testdb' -
c "SHOW port"
 port
-------
 16030
(1 row)

$ psql 'host=localhost,localhost,localhost port=16010,16020,16030 load_balance_hosts=random dbname=testdb' -
c "SHOW port"
 port
-------
 16020
(1 row)
```

psql に渡す接続文字列（参考）

- -h, -p にカンマリストを渡して実行することも可能
  - load_balance_host は psql に対応するオプションがないので接続文字列として渡す。

```
$ psql -h localhost,localhost,localhost -p 16010,16020,16030
'load_balance_hosts=random dbname=testdb' -c "SHOW port"
 port
-------
 16030
(1 row)
```

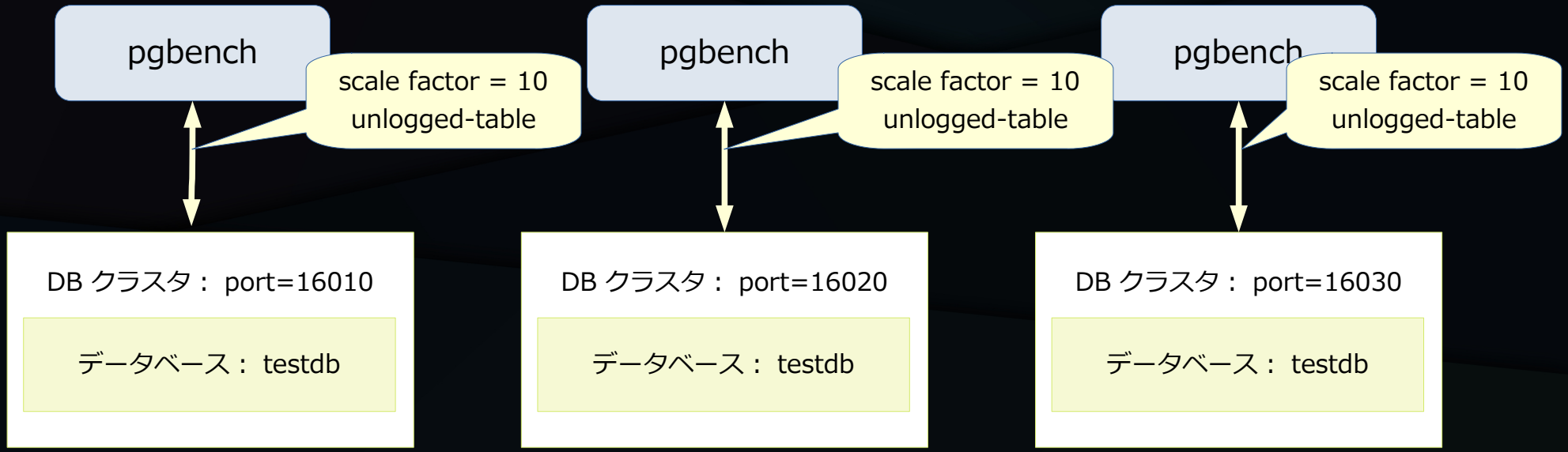
ロードバランス機能の挙動

- load_balance_host=random 時の挙動
  - ホストのリストをランダムにシャッフル
  - シャッフル後のリストに対して順々に接続を試行
- 接続エラーになったら次のホストへ接続にいく
- 接続エラーが返るまでに時間がかかるケースを想定して、接続文字列 connect_timeout （単位：秒）の設定も検討する

pgbench での実行例

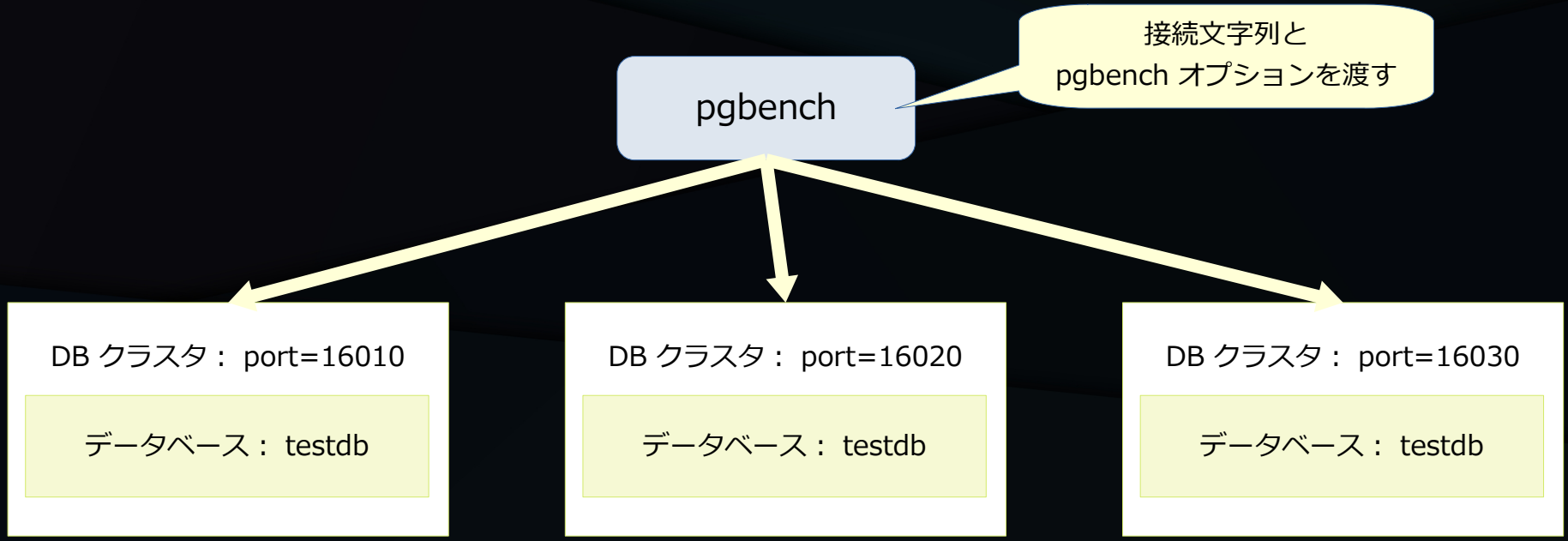
pgbench での実行例

- 3 つのデータベースクラスタに対して pgbench で同じスケールファクタで初期化しておく



pgbench での実行例

- 3 つのデータベースクラスタに対して pgbench で負荷をかける例



pgbench 実行時の引数

- pgbench 実行時の引数

引数	設定値	意味
-n	（なし）	ベンチマーク実行前の VACUUM を抑止
-P	1	1 秒毎に tps/latency を出力
-b	tpcb-like	デフォルトトランザクション (SELECT, UPDATE, INSERT ）を実行
-C	（なし）	トランザクション毎に接続を行う
-c	2	同時接続数
-t	500	合計 1000 トランザクションを実行
（接続文字列）	'host=localhost,localhost,localhost port=16010,16020,16030 load_balance_hosts=random dbname=testdb'	接続文字列内容は psql 実行時を参照

pgbench でのロードバランス実行例

- 実行中の例

```
$ pgbench -n -P 1 -C -c 2 -t 500 'host=localhost,localhost,localhost port=16010,16020,16030
load_balance_hosts=random dbname=
testdb'
pgbench (16devel)
progress: 1.0 s, 276.6 tps, lat 4.879 ms stddev 2.208, 0 failed
progress: 2.0 s, 274.4 tps, lat 4.886 ms stddev 2.150, 0 failed
progress: 3.0 s, 276.8 tps, lat 4.809 ms stddev 2.245, 0 failed
transaction type: <builtin: TPC-B (sort of)>
scaling factor: 10
query mode: simple
number of clients: 2
number of threads: 1
maximum number of tries: 1
number of transactions per client: 500
number of transactions actually processed: 1000/1000
number of failed transactions: 0 (0.000%)
latency average = 4.799 ms
latency stddev = 2.194 ms
average connection time = 2.279 ms
tps = 276.235768 (including reconnection times)
$
```

pgbench でのロードバランス実行例

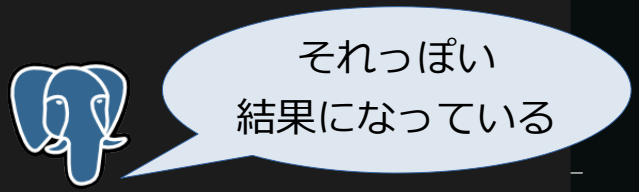
- 実行後に各データベースクラスタの testdb に接続し、pgbench_hisotry の件数を確認する

```
$ psql -p 16010 -U postgres testdb -c "SELECT COUNT(*) FROM pgbench_history"
 count
-------
   349
(1 row)

$ psql -p 16020 -U postgres testdb -c "SELECT COUNT(*) FROM pgbench_history"
 count
-------
   320
(1 row)

$ psql -p 16030 -U postgres testdb -c "SELECT COUNT(*) FROM pgbench_history"
 count
-------
   331
(1 row)

$
```



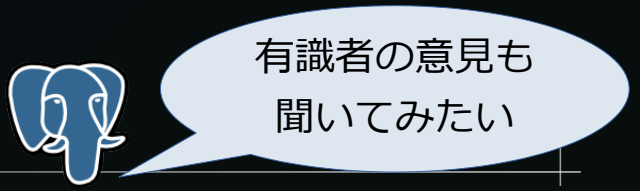
従来のロードバランサとの使い分け

ロードバランサ製品 / サービス

- ロードバランサ機能をもつ既存製品
  - Pgpool-II
  - HAProxy
- クラウド基盤が提供するロードバランスサービス
  - AWS Elastic Load Balancing
  - GCP Cloud Load Balancing

どう使い分ける？

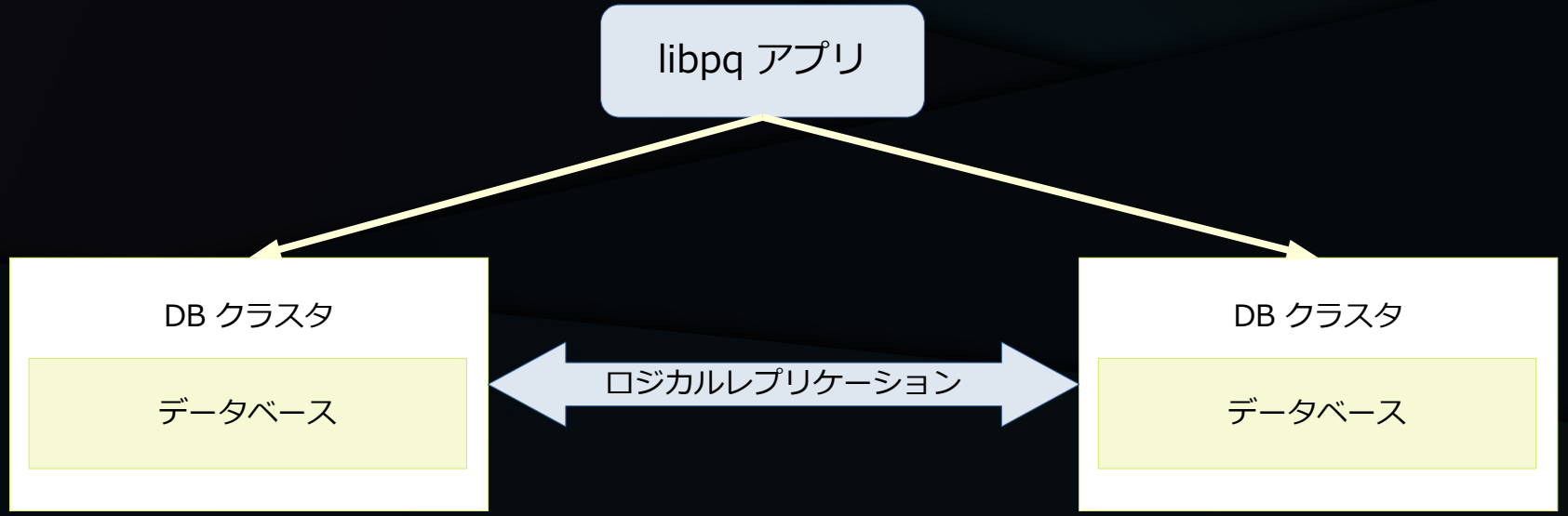
- クエリ内容からプライマリ /Read Replica への振り分けが可能なのは、たぶん Pgpool-II のみ
- その他製品はアプリ側で、プライマリ / リードレプリカ先への振り分けが必要？
  - libpq ロードバランスも同様の問題がある
- 環境（オンプレミス，クラウド基盤）によって選定？



ロードバランス機能の応用例

libpq ロードバランサ利用例

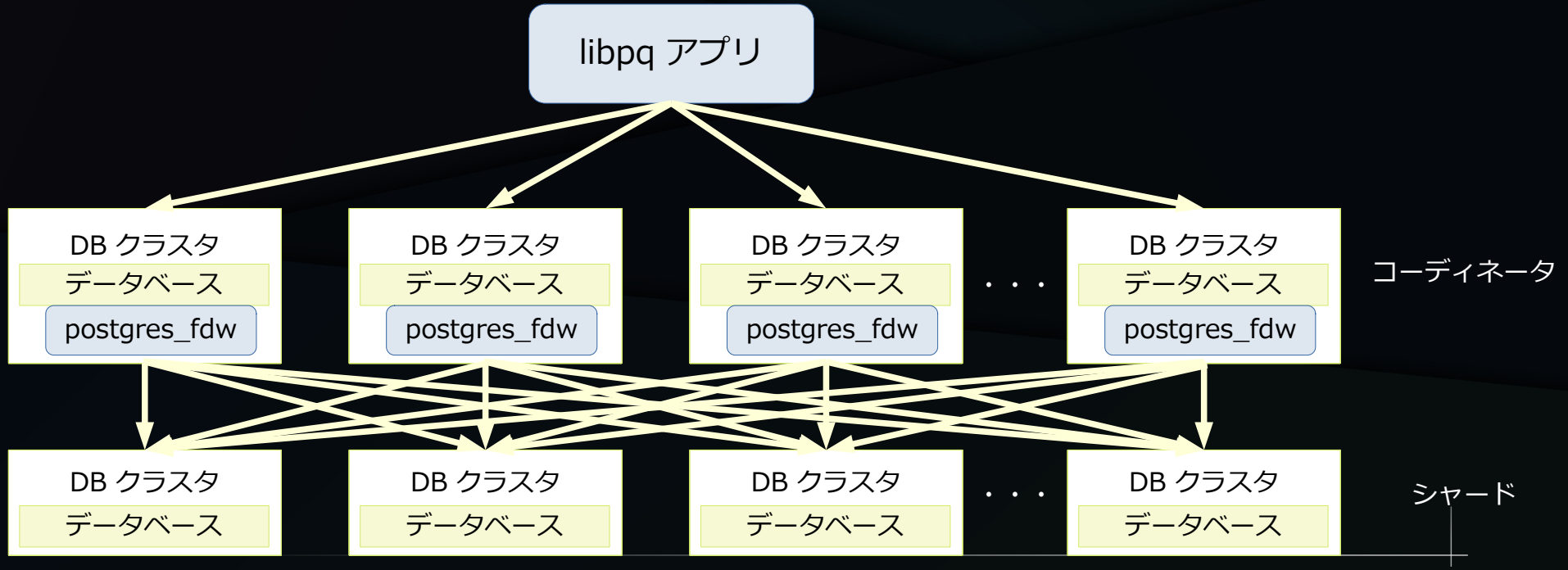
- マルチマスタ構成が組めるようになったら・・・



- マルチマスタ構成の可能性については第 40 回 PostgreSQL アンカンファレンスの発表を参照

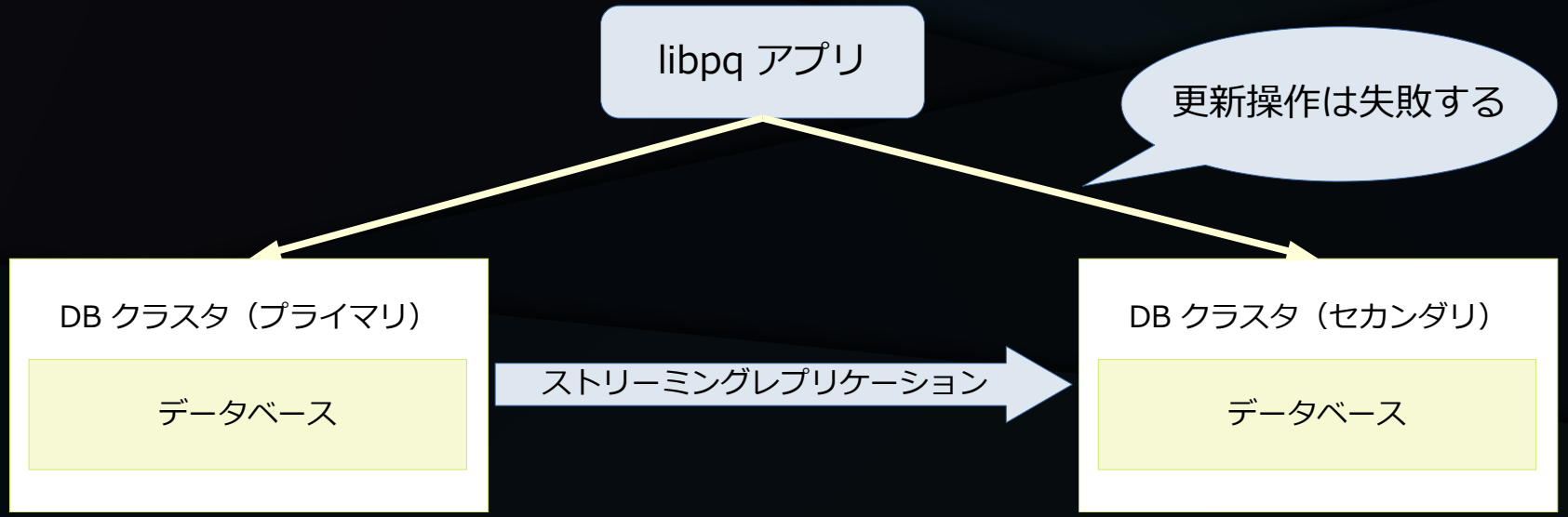
libpq ロードバランサ利用例

- パーティション＋ FDW 水平分散構成へのロードバランス



libpq ロードバランサ利用例

- Streaming Replication の参照分散には使えない🚫



- 参照のみのクエリを実行するアプリであれば利用可能？

まとめ

まとめ

- libpq の接続文字列に load_balance_host=random を指定するとロードバランス制御を行う機能が追加（予定）
- psql, pgbench 単体でもロードバランスできる
- 適切な connect_timeout 設定も合わせて検討する
- Pgpool-II, HAProxy 等、クラウド基盤がもつ既存のロードバランス機能との使い分けも考える
- マルチマスタ構成 / 水平分散構成への応用

おしまい