Enterprise Postgres のスケールアウト機能の動作を検証

富士通のデータベース「FUJITSU Software Enterprise Postgres 14 SP1（以降、Enterprise Postgres 14 SP1 と略す）」が 2022 年 8 月にリリースされました。Enterprise Postgres 14 SP1 では、スケールアウト機能を提供しています。

本記事では、Enterprise Postgres 14 SP1 に興味をお持ちの方に、スケールアウト機能のセットアップ手順をご説明し、その利用効果を理解していただくことを目的としています。また、2 台および 3 台でスケールアウト機能を使用した際の構成や設定を記載した本記事と Enterprise Postgres 14 SP1 のマニュアルを合わせて読むことで、スケールアウト機能の設計やセットアップに関してより理解できます。

Enterprise Postgres 14 SP1 のスケールアウト機能のセットアップ手順および利用効果を以下の流れで解説します。

- スケールアウト機能の動作検証の概要
- 2 つのデータノード構成（冗長化あり）でスケールアウト機能を検証
- 3 つのデータノード構成（冗長化あり）でスケールアウト機能を検証
- スケールアウト機能の検証結果を評価
- まとめ

なお、「Enterprise Postgres の概要」を知りたい方には、以下の製品サイトがお薦めです。

- Enterprise Postgres の概要

また、スケールアウト機能紹介や技術解説については、以下の記事がお薦めです。

- Enterprise Postgres 14 SP1 のスケールアウト機能を紹介 ～ Active-Active 構成を実現 ～
- Enterprise Postgres 14 SP1 のスケールアウト機能の技術を解説

スケールアウト機能の動作検証の概要

Enterprise Postgres 14 SP1 のスケールアウト機能における動作検証の概要をご紹介します。

目的

本記事でご紹介する動作検証の目的は、以下の 3 つです。

1. Enterprise Postgres 14 SP1 のスケールアウト機能のセットアップ手順を理解
2. データノードの追加手順を理解
3. データノード追加による処理能力の向上を確認

モデル

本記事では、以下の業務モデルを前提に Enterprise Postgres 14 SP1 のスケールアウト機能を検証します（図 1 参照）。

- 同一系統業務を支店ごとに実施する。
- 今後も同様に支店ごとに業務が追加される。
- 支店ごとに独立した業務を継続する。ただし、マスター情報は一元管理する。

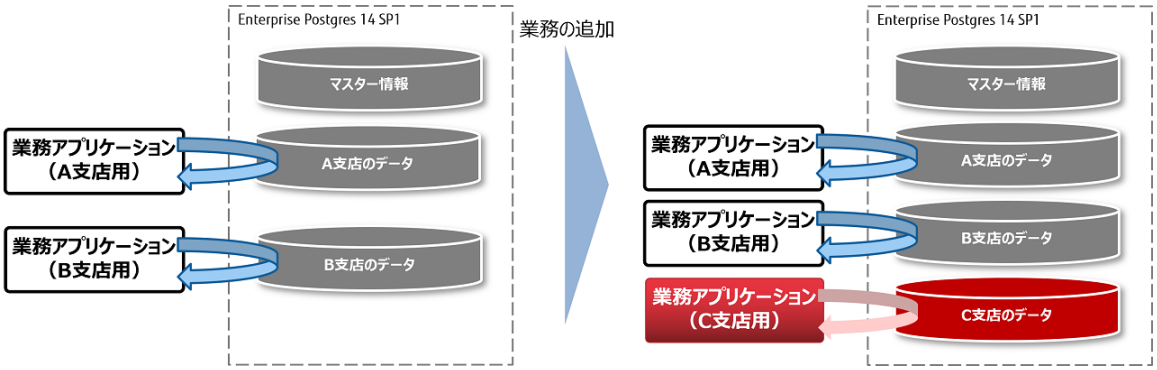


図 1 スケールアウト機能の検証モデル

検証内容

本記事では、Enterprise Postgres 14 SP1 のスケールアウト機能を以下の 3 項目で検証します。

1. 2 つのデータノード構成で構築でき、スケールアウト機能が動作すること
2. 構築したクラスタシステムにデータノードを追加でき、スケールアウト機能が動作すること
3. データノードの数に比例して処理能力が向上すること

本記事の後半で、上記項目の手順および結果をご紹介します。

環境

Enterprise Postgres 14 SP1 のスケールアウト機能を検証する環境は、以下となります（図 2 と図 3 参照）。

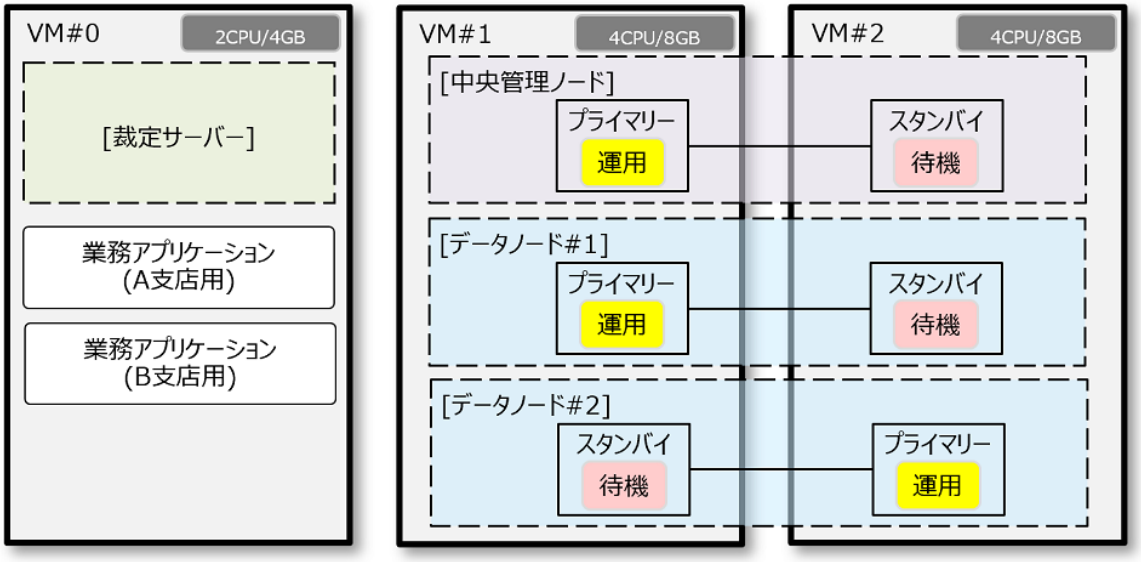


図 2 2 つのデータノード構成時の環境

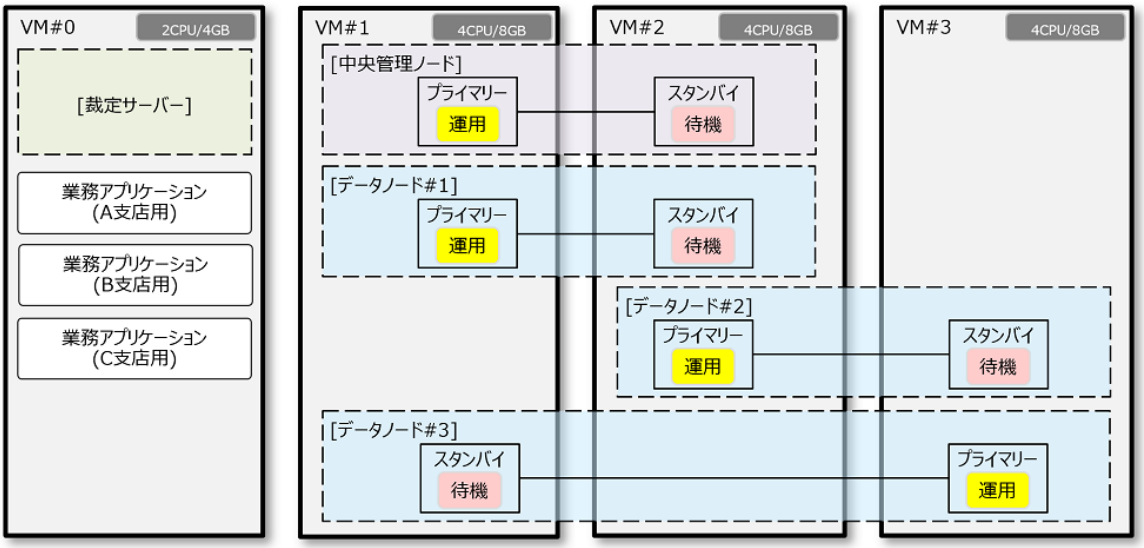


図 3 3 つのデータノード構成時の環境

上記 4 台のソフトウェア環境とハードウェア環境は、以下となります。

- ［ソフトウェア環境］
  OS：Red Hat Enterprise Linux release 8.6（Ootpa）
- ［VM#0 のハードウェア環境］
  コア数：2
  CPU のモデル名：Intel® Xeon® Platinum 8260 CPU @ 2.40GHz
  メモリー：4GB
  ディスク容量：120GB
- ［VM#1／VM#2／VM#3 のハードウェア環境］
  コア数：4
  CPU のモデル名：Intel® Xeon® Platinum 8260 CPU @ 2.40GHz
  メモリー：8GB
  ディスク容量：120GB

また、検証には以下のテーブルを使用します（図 4 と図 5 参照）。

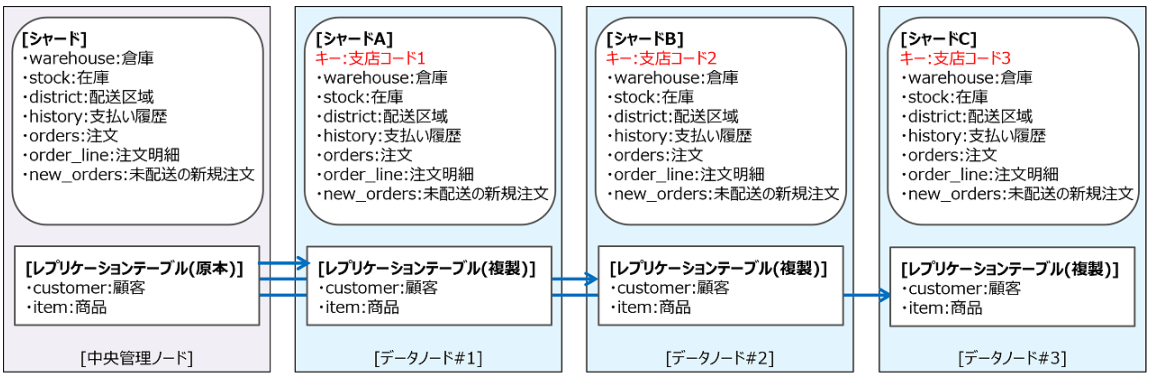


図 4 テーブル構成

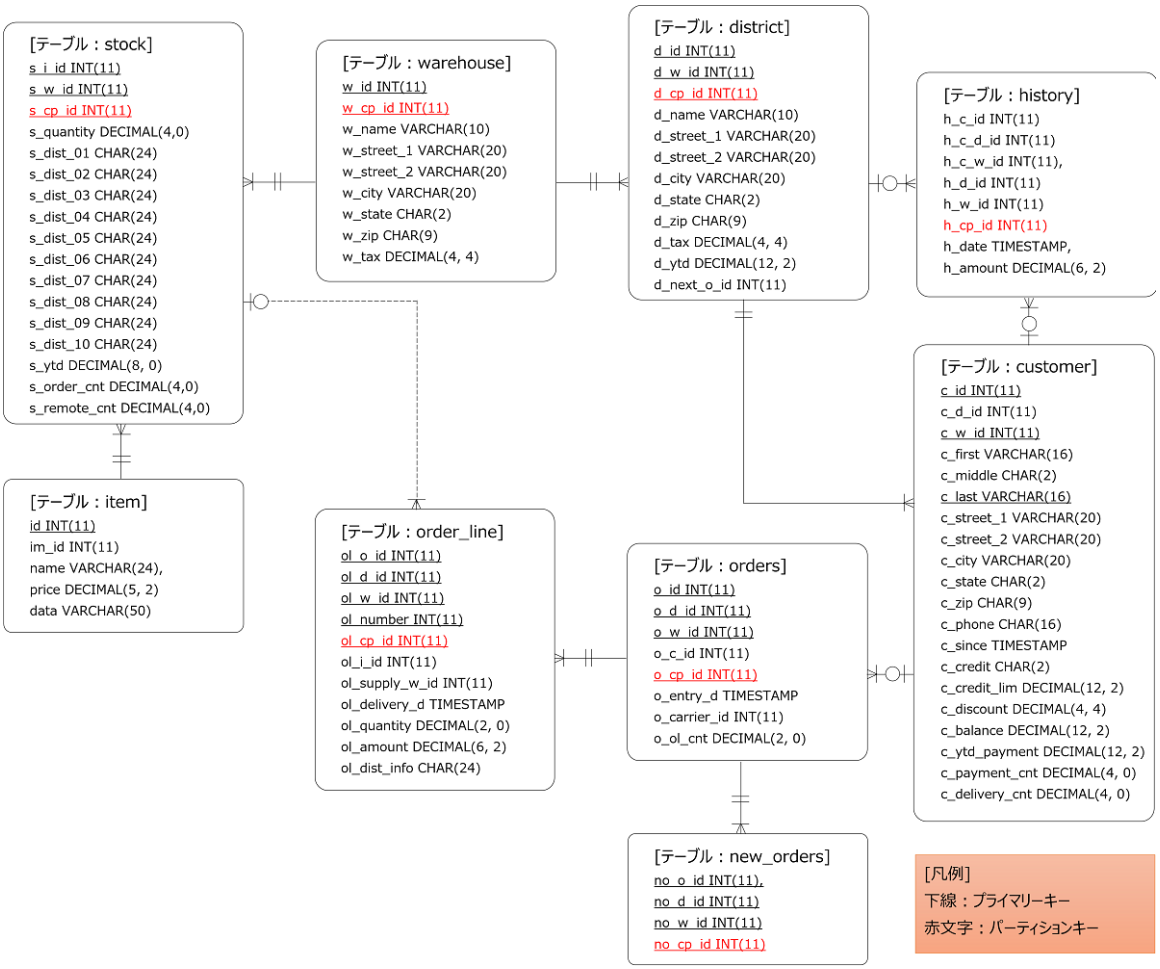


図 5 テーブルの ER 図

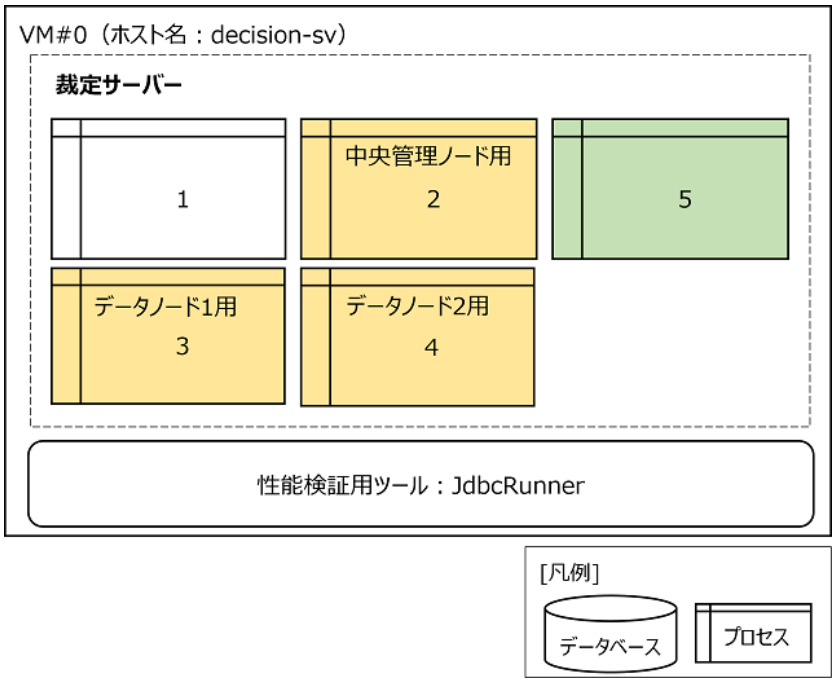
「業務アプリケーション」および「テーブル構成」は、JdbcRunner の TPC-C をもとにしています。業務アプリケーションとして利用する JdbcRunner の TPC-C の処理内容やテーブル構成の詳細に関しては、以下 JdbcRunner のオフィシャルページをご覧ください。

参考

- JdbcRunner の TPC-C とは（JdbcRunner のオフィシャルページへ）
  https://dbstudy.info/jdbcrunner/docs_ja/tpc-c.html#tpc-c

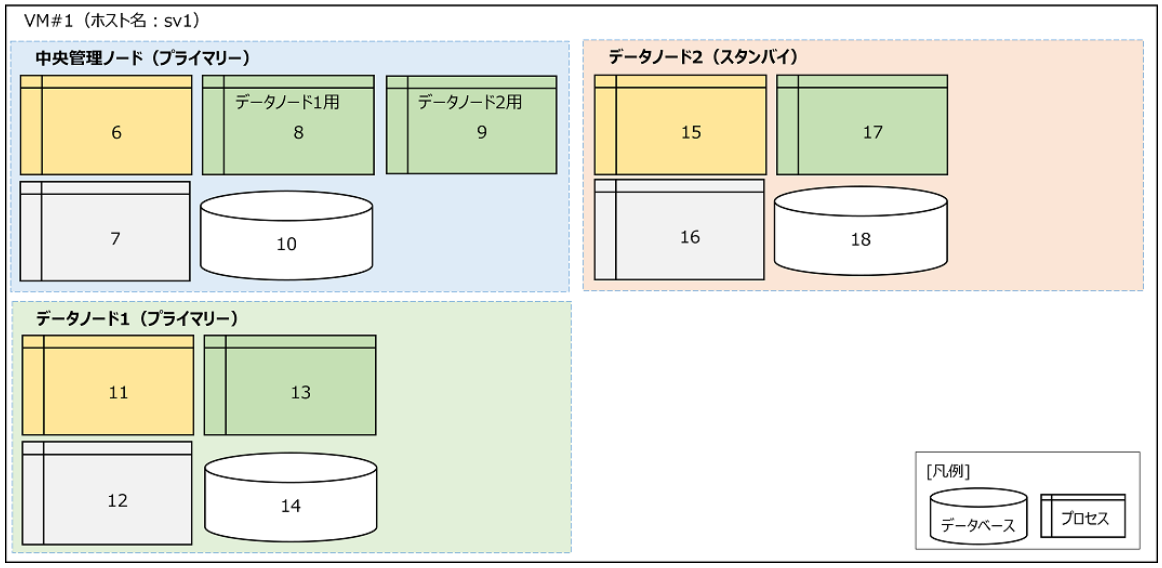
2 つのデータノード構成（冗長化あり）でスケールアウト機能を検証

ここでは、2 つのデータノード構成のセットアップ手順と性能検証手順について説明します。
スケールアウト機能を検証するために、Enterprise Postgres 14 SP1 を用いて、図 6 から図 8 の環境を構築します。



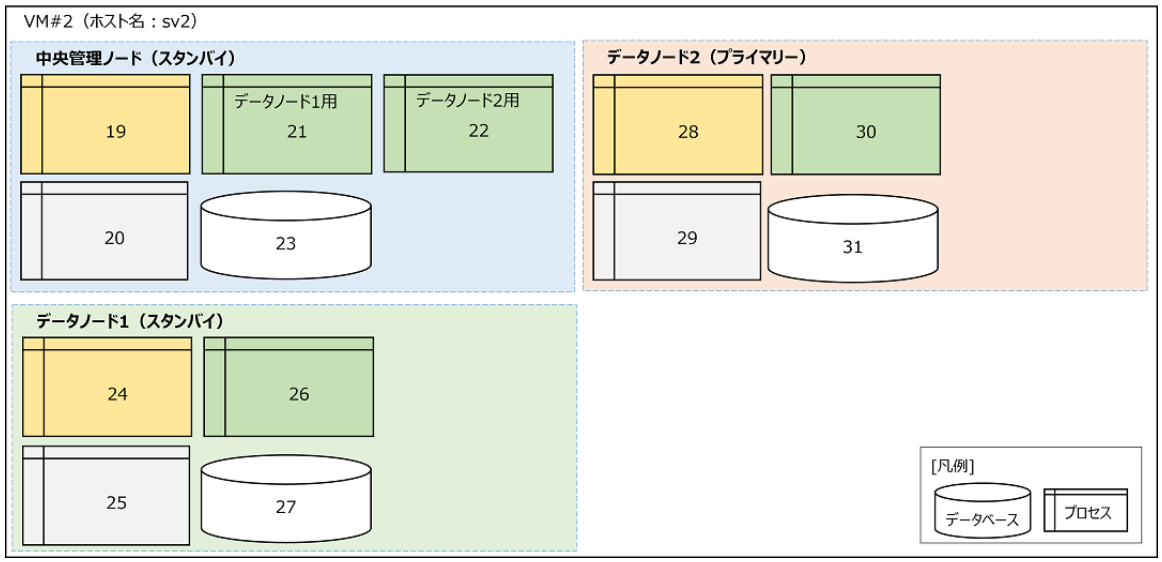
No.	プロセス	ポート	ディレクトリー
1	Scale out Controller プロセス	30000	/home/fsepuser/scalout_controller
2	Mirroring Controller 裁定プロセス	27542	/home/fsepuser/mc_dir/arbiter1
3	Mirroring Controller 裁定プロセス	27543	/home/fsepuser/mc_dir/arbiter2
4	Mirroring Controller 裁定プロセス	27544	/home/fsepuser/mc_dir/arbiter3
5	Connection Manager プロセス	27546	/home/fsepuser/app_server_conmgr

図 6 2 つのデータノード構成時の VM#0 環境



No.	プロセス	ポート	ディレクトリー
6	Mirroring Controller 裁定プロセス	27540 27541	/home/fsepuser/mc_dir
7	watchdog プロセス	27545	–
8	Connection Manager プロセス	27546	/home/fsepuser/conmgr_primary_dir
9	Connection Manager プロセス	27549	/home/fsepuser/conmgr_primary2_dir
10	–	27500	/database/inst2
11	Mirroring Controller 裁定プロセス	27542 27543	/home/fsepuser/mc_datanode1_dir
12	watchdog プロセス	27548	–
13	Connection Manager プロセス	27547	/home/fsepuser/conmgr_datanode1_dir
14	–	27501	/database/inst3
15	Mirroring Controller 裁定プロセス	27644 27645	/home/fsepuser/mc_datanode2_dir
16	watchdog プロセス	27648	–
17	Connection Manager プロセス	27647	/home/fsepuser/conmgr_datanode2_dir
18	–	27601	/database/inst4

図 7 2 つのデータノード構成時の VM#1 環境



No.	プロセス	ポート	ディレクトリー
19	Mirroring Controller 裁定プロセス	27540 27541	/home/fsepuser/mc_dir
20	watchdog プロセス	27545	–
21	Connection Manager プロセス	27546	/home/fsepuser/conmgr_standby_dir
22	Connection Manager プロセス	27549	/home/fsepuser/conmgr_standby2_dir
23	–	27500	/database/inst2
24	Mirroring Controller 裁定プロセス	27542 27543	/home/fsepuser/mc_datanode1_dir
25	watchdog プロセス	27548	–
26	Connection Manager プロセス	27547	/home/fsepuser/conmgr_datanode1_dir
27	–	27501	/database/inst3
28	Mirroring Controller 裁定プロセス	27644 27645	/home/fsepuser/mc_datanode2_dir
29	watchdog プロセス	27648	–

No.	プロセス	ポート	ディレクトリー
30	Connection Manager プロセス	27647	/home/fsepuser/conmgr_datanode2_dir
31	–	27601	/database/inst4

図 8 2 つのデータノード構成時の VM#2 環境

スケールアウト機能のセットアップ手順

2 つのデータノード構成でスケールアウト機能を検証するために、Enterprise Postgres 14 SP1 を用いて環境を構築します。

① 設計

② Enterprise Postgres 14 SP1 をインストール

③ スケールアウト機能をセットアップ

④ 動作確認

以降で、上記の各手順をご紹介します。

① 設計

まず、Enterprise Postgres 14 SP1 のスケールアウト機能をセットアップするために以下を設計してください。

- インスタンス管理者ユーザー（fsepuser とする）
- 図 6 から図 8 内の各プロセスが利用するポート番号
- 図 6 から図 8 内の各プロセスが利用するディレクトリー

② Enterprise Postgres 14 SP1 をインストール

マニュアルを参考にし、各サーバーに Enterprise Postgres 14 SP1 をインストールしてください。

［VM#0 の場合］

- インストール機能：
  - サーバー機能
  - クライアント機能
  - サーバーアシスタント機能

参照マニュアル

- FUJITSU Enterprise Postgres 14 SP1 導入ガイド（サーバ編）
- FUJITSU Enterprise Postgres 14 SP1 導入ガイド（クライアント編）
- FUJITSU Enterprise Postgres 14 SP1 導入ガイド（サーバアシスタント編）

［VM#1 と VM#2 の場合］

- インストール機能：
    - サーバー機能
    - クライアント機能

参照マニュアル：

- FUJITSU Enterprise Postgres 14 SP1 導入ガイド（サーバ編）
- FUJITSU Enterprise Postgres 14 SP1 導入ガイド（クライアント編）

③ スケールアウト機能をセットアップ

続いて、以下のマニュアルを参照し Enterprise Postgres 14 SP1 のスケールアウト機能をセットアップしましょう。

- FUJITSU Enterprise Postgres 14 SP1 スケールアウト運用ガイド
  第 3 章 スケールアウト環境のセットアップ

本検証では図 9 の流れでセットアップを実施します。

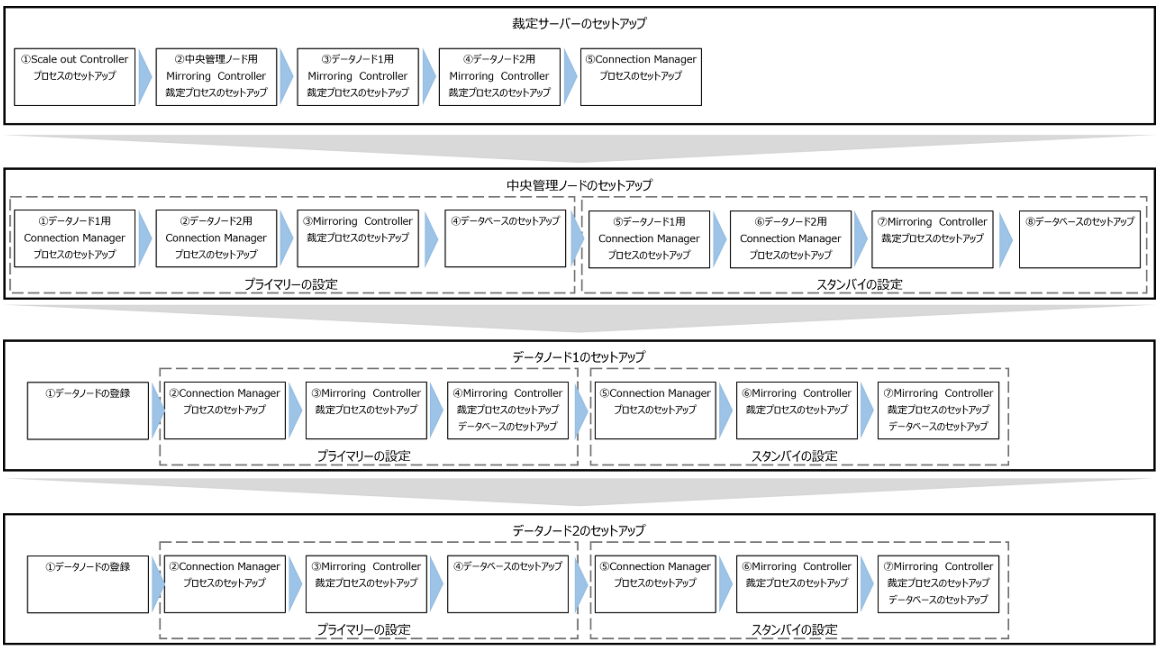


図 9 2 つのデータノード構成でスケールアウト機能をセットアップする流れ

裁定サーバーのセットアップ「⑤Connection Manager プロセスのセットアップ」で作成する設定ファイル conmgr.conf には、注意が必要です。裁定サーバー上の conmgr.conf は、中央管理ノードやデータノードで作成する conmgr.conf と異なり、全ノード分の情報を設定する必要があります。本セットアップで作成した設定ファイル conmgr.conf は、以下となります。

Connection Manager の設定ファイルである conmgr.conf の一部抜粋

```
port=27546
#中央管理ノード(プライマリ)の情報
node_name0='COORDINATOR'
backend_host0='sv1'
```



```
backend_port0=27500
watchdog_port0=27545
#中央管理ノード(スタンバイ)の情報
node_name1='COORDINATOR'
backend_host1='sv2'
backend_port1=27500
watchdog_port1=27545
・
・
・
#データノード 2(プライマリ)の情報
node_name4='datanode2'
backend_host4='sv2'
backend_port4=27601
watchdog_port4=27648
#データノード 2(スタンバイ)の情報
node_name5='datanode2'
backend_host5='sv1'
backend_port5=27601
watchdog_port5=27648
```

④ 動作確認

最後に、Enterprise Postgres 14 SP1 のスケールアウト機能のセットアップが完了したことを確認するために、裁定サーバー上で以下のコマンドを順番に実行します。

```
# su – fsepuser <enter>
$ cm_ctl status -D /home/fsepuser/app_server_conmgr/ -i all <enter>
conmgr_status:
status           pid
ready            2110956

instance_information:
addr                                  port  database_attr node_name
sv2                                   27500 standby       COORDINATOR
sv1                                   27500 primary       COORDINATOR
sv2                                   27501 standby       datanode1
sv1                                   27501 primary       datanode1
sv1                                   27601 standby       datanode2
sv2                                   27601 primary       datanode2

application_information:
addr                                  port  pid        connected_time

shard_information:
shard_name       node_name
shard_datanode2  datanode2
shard_datanode1  datanode1
```


コマンド実行後、「shard_datanode1」と「shard_datanode2」が表示された場合、「Enterprise Postgres 14 SP1 のスケール機能のセットアップが正常に完了した」と判断できます。

性能検証の手順

以下の流れで「2 つのデータノード構成でスケールアウト機能の性能検証」を実施します。

⑤ 性能検証ツールのダウンロードと修正

⑥ 検証用データを作成

⑦ 作成したデータをインポート

⑧ スケールアウト機能の性能検証を実施

上記手順をご紹介します。

⑤ 性能検証ツールのダウンロードと修正

性能検証には、汎用データベース負荷テストツール「JdbcRunner」を使います。
以下から JdbcRunner バージョン:1.3 をダウンロードしてください。

- JdbcRunner をダウンロード（JdbcRunner のオフィシャルページへ）
  https://dbstudy.info/jdbcrunner/

ダウンロード後、裁定サーバー上で JdbcRunner を展開します。
その後、scripts/tpcc.js をコピーし、表 1 の箇所を修正した「tpcc_scaleout_datanode1.js」「tpcc_scaleout_datanode2.js」を作成しましょう。

表 1 tpcc.js の修正箇所

行番号	修正前	修正後
16	// var jdbcUrl = "jdbc:postgresql://localhost:5432/tpcc";	var jdbcUrl = "jdbc:postgresql://localhost:27546/postgres?shardName=shard_datanode1"; //データノード 2 側の性能検証をする場合は「shard_datanode2」を設定すること
18	var jdbcUser = "tpcc";	var jdbcUser = "fsepuser";
19	var jdbcPass = "tpcc";	var jdbcPass = "fsepuser";
26	var logDir = "logs";	var logDir = "logs"; var PartitionKey = 1;//データノード 2 側の性能検証をする場合は 2 を設定すること
188	"(o_id, o_d_id, o_w_id, o_c_id, o_entry_d, "	"(o_id, o_d_id, o_w_id, o_c_id, o_cp_id, o_entry_d, "
190	"VALUES ($int, $int, $int, $int, $timestamp, "	"VALUES ($int, $int, $int, $int, $int, $timestamp, "

行番号	修正前	修正後
192	rs02[0][1], d_id, w_id, c_id, new Date(),	rs02[0][1], d_id, w_id, c_id, PartitionKey, new Date(),
196	+ "(no_o_id, no_d_id, no_w_id) " + "VALUES ($int, $int, $int)", rs02[0][1], d_id, w_id);	+ "(no_o_id, no_d_id, no_w_id, no_cp_id) " + "VALUES ($int, $int, $int, $int)", rs02[0][1], d_id, w_id, PartitionKey);
243	+ "(ol_o_id, ol_d_id, ol_w_id, ol_number, ol_i_id, "	+ "(ol_o_id, ol_d_id, ol_w_id, ol_number, ol_cp_id, ol_i_id, "
246	+ "VALUES ($int, $int, $int, $int, $int, "	+ "VALUES ($int, $int, $int, $int, $int, $int, "
249	rs02[0][1], d_id, w_id, index + 1, i_id[order[index]],	rs02[0][1], d_id, w_id, index + 1, PartitionKey, i_id[order[index]],
411	+ "(h_c_id, h_c_d_id, h_c_w_id, h_d_id, h_w_id, "	+ "(h_c_id, h_c_d_id, h_c_w_id, h_d_id, h_w_id, h_cp_id,"
413	+ "VALUES ($int, $int, $int, $int, $int, "	+ "VALUES ($int, $int, $int, $int, $int, $int,"
415	c_id, c_d_id, c_w_id, d_id, w_id,	c_id, c_d_id, c_w_id, d_id, w_id, PartitionKey,

⑥ 検証用データを作成

次に、ダウンロードした JdbcRunner を用いて、性能検証で利用するデータを作成します。JdbcRunner のオフィシャルページでテストデータの作成方法が説明されていますのでご覧ください。

- JdbcRunner でテストデータの作成（JdbcRunner のオフィシャルページへ）
  https://dbstudy.info/jdbcrunner/docs_ja/tpc-c.html#id6

作成後、図 5 内に記載された「パーティションキー」のカラムを該当テーブルに追加してください。
追加後、データノード 1 用のテストデータを作成するために、パーティションキーカラムに「1」を挿入し、CSV としてエクスポートします。
その後、データノード 2 用のテストデータを作成するために、パーティションキーカラムに「2」を挿入し、CSV としてエクスポートします。

⑦ 作成したデータをインポート

続いて、図 4 のようにシャードとレプリケーションテーブルを作成します。

- FUJITSU Enterprise Postgres 14 SP1 スケールアウト運用ガイド
  第 5 章 スケールアウト環境の運用管理
  5.1 シャード表とテーブル空間
- FUJITSU Enterprise Postgres 14 SP1 スケールアウト運用ガイド
  第 5 章 スケールアウト環境の運用管理
  5.2 レプリケーションテーブル

テーブル作成後、作成したデータを各データノードにインポートします。

Enterprise Postgres が提供する高速データロード機能（pgx_loader コマンド）を使った例

```
$ pgx_loader load -j 3 -c "copy warehouse from '/tmp/warehouse.csv' delimiter ',' csv" -d postgres -h sv1 -p 27501; <enter>
$ pgx_loader load -j 3 -c "copy district from '/tmp/district.csv' delimiter ',' csv" -d postgres -h sv1 -p 27501; <enter>
$ pgx_loader load -j 3 -c "copy history from '/tmp/history.csv' delimiter ',' csv" -d postgres -h sv1 -p 27501; <enter>
$ pgx_loader load -j 3 -c "copy stock from '/tmp/stock.csv' delimiter ',' csv" -d postgres -h sv1 -p 27501; <enter>
$ pgx_loader load -j 3 -c "copy orders from '/tmp/orders.csv' delimiter ',' csv" -d postgres -h sv1 -p 27501; <enter>
$ pgx_loader load -j 3 -c "copy new_orders from '/tmp/new_orders.csv' delimiter ',' csv" -d postgres -h sv1 -p 27501; <enter>
$ pgx_loader load -j 3 -c "copy order_line from '/tmp/order_line.csv' delimiter ',' csv" -d postgres -h sv1 -p 27501; <enter>
$ pgx_loader load -j 3 -c "copy customer from '/tmp/customer.csv' delimiter ',' csv" -d postgres -h sv1 -p 27500; <enter>
$ pgx_loader load -j 3 -c "copy item from '/tmp/item.csv' delimiter ',' csv" -d postgres -h sv1 -p 27500; <enter>
```

pgx_loader コマンドの詳しい使い方に関しては、以下を参照してください。

- FUJITSU Enterprise Postgres 14 SP1 リファレンス
  第 2 章 クライアントコマンド
  2.2 pgx_loader

⑧ スケールアウト機能の性能検証を実施

最後に、裁定サーバー上で以下のコマンドを実行することで、Enterprise Postgres 14 SP1 のスケールアウト機能の性能検証を実施できます。

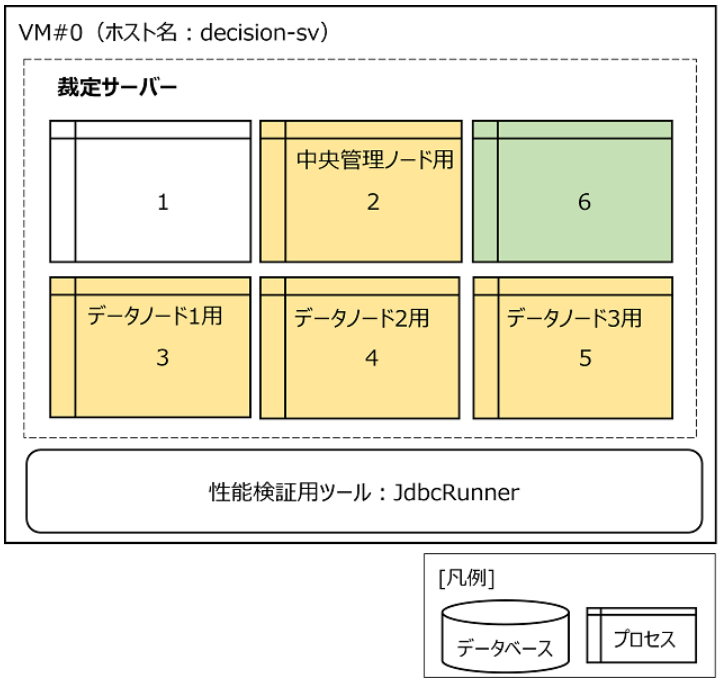
```
# 環境変数を設定
$ export CLASSPATH=/ext/jdbcrunner-1.3/jdbcrunner-1.3-nojdbc.jar:/opt/fsepv14client64/jdbc/lib/postgresql-jdbc42.jar:$CLASSPATH
<enter>


# 性能検証
$ java JR scripts/tpcc_scaleout_datanode1.js -nAgents 100 & <enter>
$ java JR scripts/tpcc_scaleout_datanode2.js -nAgents 100 & <enter>
```

上記のコマンドを順番に実行することで、ログファイル jdbcrunner.log と 4 つの CSV ファイルが作成されます。

以上の手順により、2 つのデータノード構成で Enterprise Postgres 14 SP1 のスケールアウト機能を検証できました。
性能検証の結果に関しては、本記事の後半に掲載します。

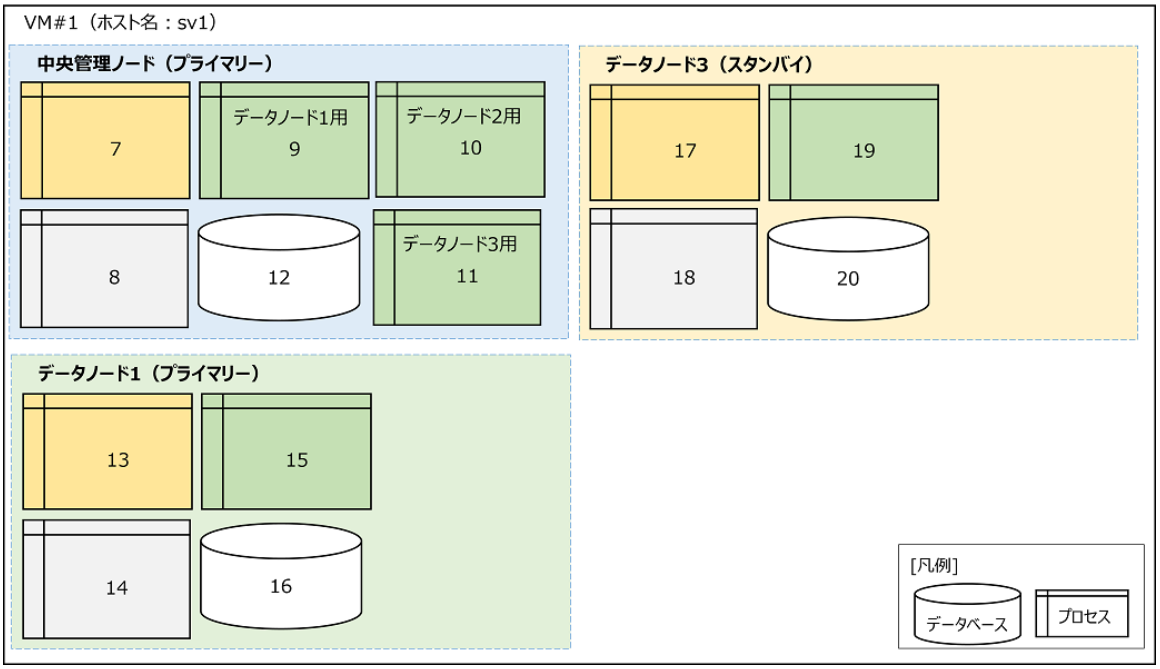
3 つのデータノード構成（冗長化あり）でスケールアウト機能を検証

ここでは、2 つのデータノード構成にデータノードを 1 つ追加する場合のセットアップ手順と性能検証手順について説明します。
スケールアウト機能の拡張性を検証するために、Enterprise Postgres 14 SP1 を用いて、図 10 から図 13 の環境を構築します。



No.	プロセス	ポート	ディレクトリー
1	Scale out Controller プロセス	30000	/home/fsepuser/scalout_controller
2	Mirroring Controller 裁定プロセス	27542	/home/fsepuser/mc_dir/arbiter1
3	Mirroring Controller 裁定プロセス	27543	/home/fsepuser/mc_dir/arbiter2
4	Mirroring Controller 裁定プロセス	27544	/home/fsepuser/mc_dir/arbiter3
5	Mirroring Controller 裁定プロセス	28545	/home/fsepuser/mc_dir/arbiter_for_sv3
6	Connection Manager プロセス	27546	/home/fsepuser/app_server_conmgr

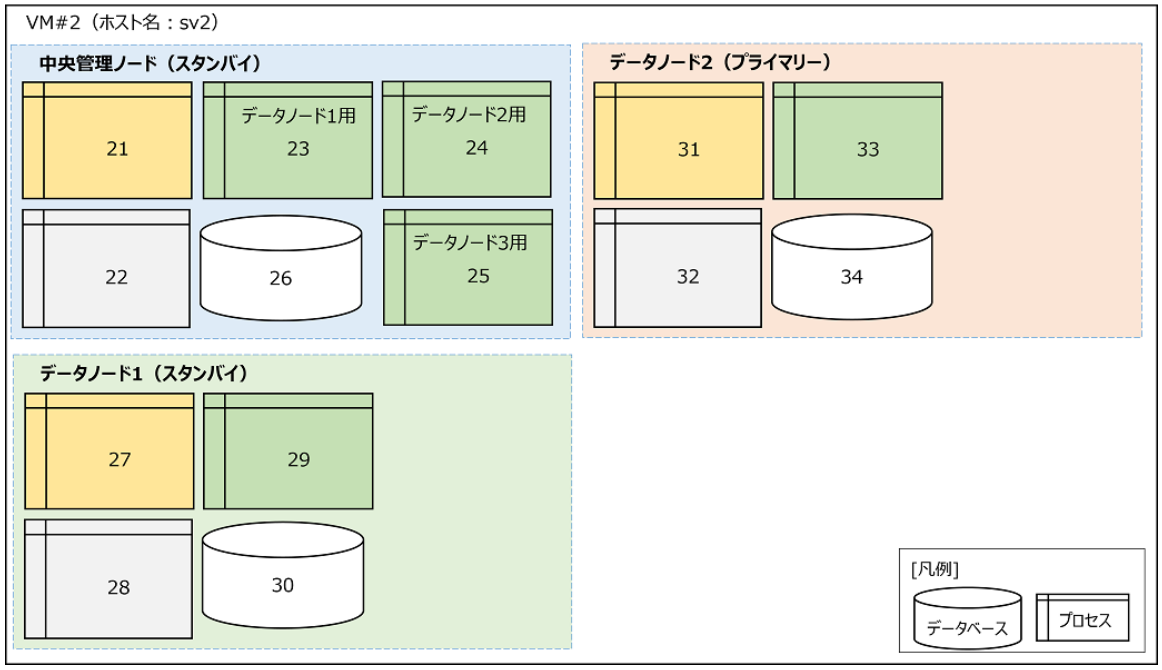
図 10 3 つのデータノード構成時の VM#0 環境



No.	プロセス	ポート	ディレクトリー
7	Mirroring Controller 裁定プロセス	27540 27541	/home/fsepuser/mc_dir
8	watchdog プロセス	27545	–
9	Connection Manager プロセス	27546	/home/fsepuser/conmgr_primary_dir
10	Connection Manager プロセス	27549	/home/fsepuser/conmgr_primary2_dir
11	Connection Manager プロセス	28546	/home/fsepuser/conmgr_primary_sv3_dir
12	–	27500	/database/inst2
13	Mirroring Controller 裁定プロセス	27542 27543	/home/fsepuser/mc_datanode1_dir
14	watchdog プロセス	27548	–
15	Connection Manager プロセス	27547	/home/fsepuser/conmgr_datanode1_dir
16	–	27501	/database/inst3
17	Mirroring Controller 裁定プロセス	27644 27645	/home/fsepuser/mc_datanode2_dir

No.	プロセス	ポート	ディレクトリー
18	watchdog プロセス	27648	–
19	Connection Manager プロセス	27647	/home/fsepuser/conmgr_datanode2_dir
20	–	27601	/database/inst4

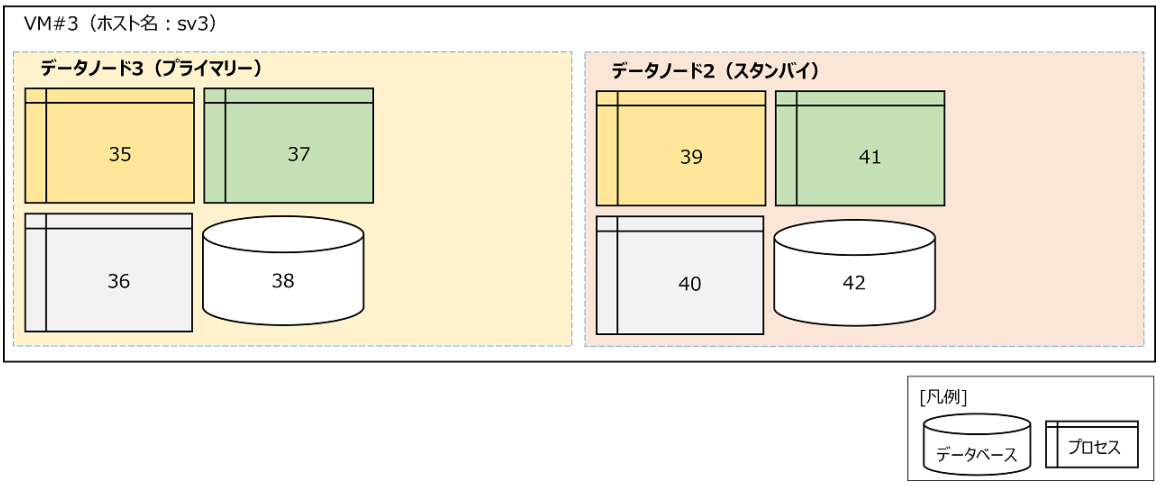
図 11 3 つのデータノード構成時の VM#1 環境



No.	プロセス	ポート	ディレクトリー
21	Mirroring Controller 裁定プロセス	27540 27541	/home/fsepuser/mc_dir
22	watchdog プロセス	27545	–
23	Connection Manager プロセス	27546	/home/fsepuser/conmgr_standby_dir
24	Connection Manager プロセス	27549	/home/fsepuser/conmgr_standby2_dir
25	Connection Manager プロセス	28546	/home/fsepuser/conmgr_standby_sv3_dir

No.	プロセス	ポート	ディレクトリー
26	–	27500	/database/inst2
27	Mirroring Controller 裁定プロセス	27542 27543	/home/fsepuser/mc_datanode1_dir
28	watchdog プロセス	27548	–
29	Connection Manager プロセス	27547	/home/fsepuser/conmgr_datanode1_dir
30	–	27501	/database/inst3
31	Mirroring Controller 裁定プロセス	27644 27645	/home/fsepuser/mc_datanode2_dir
32	watchdog プロセス	27648	–
33	Connection Manager プロセス	27647	/home/fsepuser/conmgr_datanode2_dir
34	–	27601	/database/inst4

図 12 3 つのデータノード構成時の VM#2 環境



No.	プロセス	ポート	ディレクトリー
35	Mirroring Controller 裁定プロセス	28644 28645	/home/fsepuser/mc_datanode3_dir

No.	プロセス	ポート	ディレクトリー
36	watchdog プロセス	27748	–
37	Connection Manager プロセス	27747	/home/fsepuser/conmgr_datanode3_dir
38	–	27701	/database/inst5
39	Mirroring Controller 裁定プロセス	27644 27645	/home/fsepuser/mc_datanode2_dir
40	watchdog プロセス	27648	–
41	Connection Manager プロセス	27647	/home/fsepuser/conmgr_datanode2_dir
42	–	27601	/database/inst4

図 13 3 つのデータノード構成時の VM#3 環境

データノードを追加する手順

3 つのデータノード構成でスケールアウト機能を検証するために、Enterprise Postgres 14 SP1 を用いて環境を構築しましょう。

① 設計

② Enterprise Postgres 14 SP1 をインストール

③ 追加データノードをセットアップ

④ 動作確認

以降で、上記手順をご紹介します。

① 設計

まず、データノードを追加するために、「図 10 から図 13 内の各プロセスが利用するポート番号」「図 10 から図 13 内の各プロセスが利用するディレクトリー」を設計してください。

以降の手順では、以下を想定として説明します。

- 図 10 から図 13 内に記載されたポート番号を使用する
- 図 10 から図 13 内に記載されたディレクトリーを使用する

② Enterprise Postgres 14 SP1 をインストール

次に、追加データノード用のサーバーに Enterprise Postgres 14 SP1 のサーバー機能とクライアント機能をインストールしてください。

- FUJITSU Enterprise Postgres 14 SP1 導入ガイド（サーバ編）
  第 3 章 インストール

③ 追加データノードをセットアップ

続いて、以下のマニュアルを参照し Enterprise Postgres 14 SP1 のスケールアウト機能をセットアップしましょう。

- FUJITSU Enterprise Postgres 14 SP1 スケールアウト運用ガイド
  第 5 章 スケールアウト環境の運用管理
  5.7 データノードの追加

本検証では、図 14 の流れでセットアップを実施します。

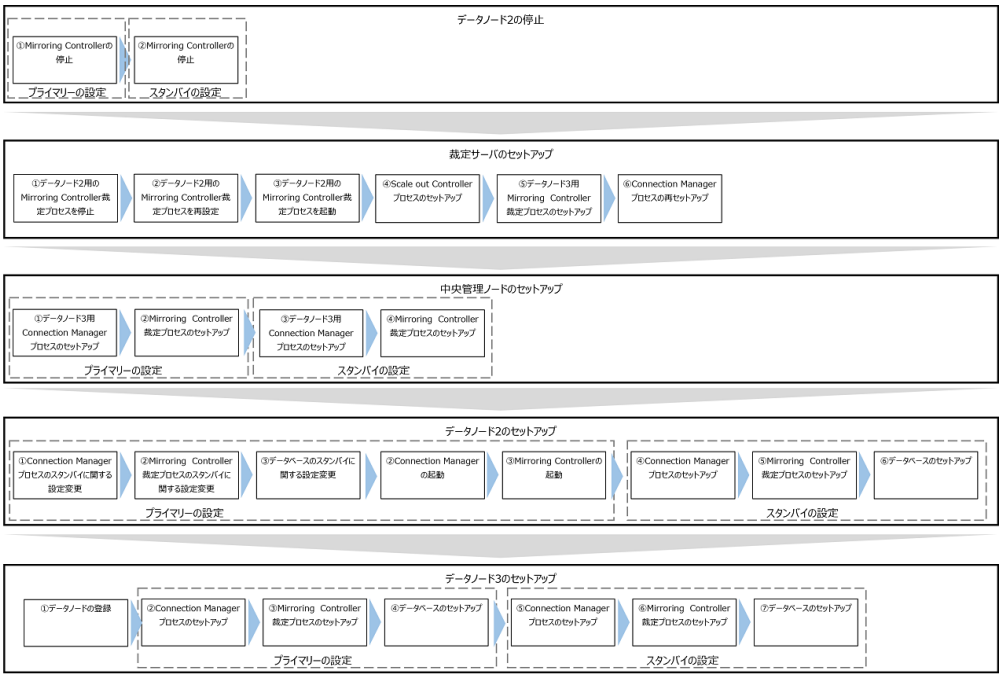


図 14 追加データノードをセットアップする流れ

④ 動作確認

最後に、セットアップが完了したことを確認するために、裁定サーバー上で以下のコマンドを順番に実行します。

```
# su - fsepuser <enter>
$ cm_ctl status -D /home/fsepuser/app_server_conmgr/ -i all <enter>
conmgr_status:
status          pid
ready           2110956

instance_information:
addr                                port  database_attr node_name
sv2                                 27500 standby       COORDINATOR
sv1                                 27500 primary       COORDINATOR
sv1                                 27701 standby       datanode3
```



```
sv3                                     27701 primary       datanode3
sv2                                     27501 standby       datanode1
sv1                                     27501 primary       datanode1
sv3                                     27601 standby       datanode2
sv2                                     27601 primary       datanode2

application_information:
addr                                    port  pid        connected_time

shard_information:
shard_name       node_name
shard_datanode2  datanode2
shard_datanode1  datanode1
shard_datanode3  datanode3
```

コマンド実行後、「shard_datanode3」が表示された場合、「追加データノードで Enterprise Postgres 14 SP1 のスケールアウト機能を正常にセットアップできた」と判断できます。

性能検証の手順

3 つのデータノード構成でスケールアウト機能の性能検証をするために、以下の手順を実施します。

⑤ 性能検証ツールの修正

⑥ 検証用データを作成

⑦ 作成したデータをインポート

⑧ スケールアウト機能の性能検証を実施

上記手順をご紹介します。

⑤ 性能検証ツールの修正

scripts/tpcc.js をコピーし、表 2 の箇所を修正した「tpcc_scaleout_datanode3.js」を作成しましょう。

表 2 tpcc.js の修正箇所

行番号	修正前	修正後
16	// var jdbcUrl = "jdbc:postgresql://localhost:5432/tpcc";	var jdbcUrl = "jdbc:postgresql://localhost:27546/postgres?shardName=shard_datanode3";
18	var jdbcUser = "tpcc";	var jdbcUser = "fsepuser";
19	var jdbcPass = "tpcc";	var jdbcPass = "fsepuser";
26	var logDir = "logs";	var logDir = "logs"; var PartitionKey = 3;

行番号	修正前	修正後
188	"(o_id, o_d_id, o_w_id, o_c_id, o_entry_d, "	"(o_id, o_d_id, o_w_id, o_c_id, o_cp_id, o_entry_d, "
190	"VALUES ($int, $int, $int, $int, $timestamp, "	"VALUES ($int, $int, $int, $int, $int, $timestamp, "
192	rs02[0][1], d_id, w_id, c_id, new Date(),	rs02[0][1], d_id, w_id, c_id, PartitionKey, new Date(),
196	+ "(no_o_id, no_d_id, no_w_id) " + "VALUES ($int, $int, $int)", rs02[0][1], d_id, w_id);	+ "(no_o_id, no_d_id, no_w_id, no_cp_id) " + "VALUES ($int, $int, $int, $int)", rs02[0][1], d_id, w_id, PartitionKey);
243	+ "(ol_o_id, ol_d_id, ol_w_id, ol_number, ol_i_id, "	+ "(ol_o_id, ol_d_id, ol_w_id, ol_number, ol_cp_id, ol_i_id, "
246	+ "VALUES ($int, $int, $int, $int, $int, "	+ "VALUES ($int, $int, $int, $int, $int, $int, "
249	rs02[0][1], d_id, w_id, index + 1, i_id[order[index]],	rs02[0][1], d_id, w_id, index + 1, PartitionKey, i_id[order[index]],
411	+ "(h_c_id, h_c_d_id, h_c_w_id, h_d_id, h_w_id, "	+ "(h_c_id, h_c_d_id, h_c_w_id, h_d_id, h_w_id, h_cp_id,"
413	+ "VALUES ($int, $int, $int, $int, $int, "	+ "VALUES ($int, $int, $int, $int, $int, $int,"
415	c_id, c_d_id, c_w_id, d_id, w_id,	c_id, c_d_id, c_w_id, d_id, w_id, PartitionKey,

⑥ 検証用データを作成

次に、ダウンロードした JdbcRunner を用いて、性能検証で利用するデータを作成します。
作成後、図 5 内に記載された「パーティションキー」のカラムを該当テーブルに追加してください。
その後、データノード 3 用のテストデータを作成するために、パーティションキーカラムに「3」を挿入し、CSV としてエクスポートします。

⑦ 作成したデータをインポート

続いて、図 4 のようにシャードを作成します。

- FUJITSU Enterprise Postgres 14 SP1 スケールアウト運用ガイド
  第 5 章 スケールアウト環境の運用管理
  5.1 シャード表とテーブル空間

検証用テーブル作成後、作成したデータをデータノード 3 にインポートします。

Enterprise Postgres が提供する高速データロード機能（pgx_loader コマンド）を使った例

```
$ pgx_loader load -j 3 -c "copy warehouse from '/tmp/warehouse.csv' delimiter ',' csv" -d postgres -h sv3 -p 27701; <enter>
$ pgx_loader load -j 3 -c "copy district from '/tmp/district.csv' delimiter ',' csv" -d postgres -h sv3 -p 27701; <enter>
$ pgx_loader load -j 3 -c "copy history from '/tmp/history.csv' delimiter ',' csv" -d postgres -h sv3 -p 27701; <enter>
$ pgx_loader load -j 3 -c "copy stock from '/tmp/stock.csv' delimiter ',' csv" -d postgres -h sv3 -p 27701; <enter>
```



```
$ pgx_loader load -j 3 -c "copy orders from '/tmp/orders.csv' delimiter ',' csv" -d postgres -h sv3 -p 27701; <enter>
$ pgx_loader load -j 3 -c "copy new_orders from '/tmp/new_orders.csv' delimiter ',' csv" -d postgres -h sv3 -p 27701; <enter>
$ pgx_loader load -j 3 -c "copy order_line from '/tmp/order_line.csv' delimiter ',' csv" -d postgres -h sv3 -p 27701; <enter>
```

⑧ スケールアウト機能の性能検証を実施

最後に、Enterprise Postgres 14 SP1 のスケールアウト機能の性能検証を実施するために、裁定サーバー上で以下のコマンドを実行します。

```
# 環境変数を設定
$ export CLASSPATH=/ext/jdbcrunner-1.3/jdbcrunner-1.3-nojdbc.jar:/opt/fsepv14client64/jdbc/lib/postgresql-jdbc42.jar:$CLASSPATH
<enter>


# 性能検証
$ java JR scripts/tpcc_scaleout_datanode1.js -nAgents 100 & <enter>
$ java JR scripts/tpcc_scaleout_datanode2.js -nAgents 100 & <enter>
$ java JR scripts/tpcc_scaleout_datanode3.js -nAgents 100 & <enter>
```

上記のコマンドを順番に実行することで、ログファイル jdbcrunner.log と 6 つの CSV ファイルが作成されます。

以上の手順により、3 つのデータノード構成で Enterprise Postgres 14 SP1 のスケールアウト機能を検証できました。
性能検証の結果に関しては、以降でご紹介します。

スケールアウト機能の検証結果を評価

データノードを「2 つ構成」「3 つ構成」で測定した性能値を以下の図に示します。
また、参考値としてスケールアウト機能を利用しなかった場合の性能値（台数 1）も掲載します。

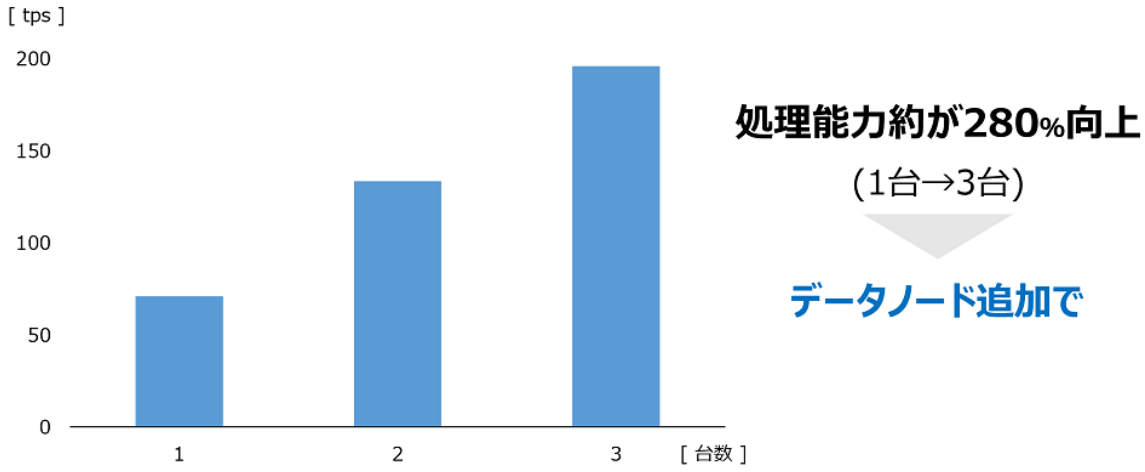


図 15 性能評価

上記結果から、検証に利用した業務モデルに Enterprise Postgres 14 SP1 のスケールアウト機能を適用することで、処理能力の向上が見込めます。

まとめ：Enterprise Postgres のスケールアウト機能で性能向上

本記事で紹介した「Enterprise Postgres 14 SP1 のスケールアウト機能の構築」および「データノードを追加する方法」を理解していただくと、今後、Enterprise Postgres 14 SP1 のスケールアウト機能の環境構築や既存業務システムの処理能力向上をスムーズに実現できます。

最後に本記事のまとめを掲載します。

［Enterprise Postgres 14 SP1 のスケールアウト機能の動作検証のまとめ］

- 最小で 2 つのデータノード構成からスモールスタートできる
- データノードを後から追加することで、処理能力を向上できる
- データノード間でロック競合などの影響を受けないため、銀行業務のような支店ごとに業務追加するモデルに利用できる

Enterprise Postgres 14 SP1 に関するお問い合わせは本ページに表示されている「お問い合わせ」ボタンから、または本ページ下部に記載の「本コンテンツに関するお問い合わせ」からご連絡ください。

2022 年 10 月 14 日