PGECons
PostgreSQL Enterprise Consortium

2020年度WG3活動成果報告
クラウド検証編
Azure PostgreSQL PaaS検証
～FlexibleServer～

PostgreSQL エンタープライズコンソーシアム
WG3 パブリッククラウド検証チーム
日鉄ソリューションズ株式会社　永井 光

責任範囲

- 本資料は、PGEConsが独自に検証した結果であり、結果はPGEConsの責任の元、公開しています。

Contents

- 検証の背景
- FlexibleServer 机上検証
- FlexibleServer 実機検証

本検証の背景

- 昨年度（2019年度）にAzureのPostgreSQLのPaaSである
  Azure Database for PostgreSQLのうち、SingleServerの机上/実機検証、
  およびHyperscaleの机上検証を実施

- 2020年度は、上記の継続として、新サービスとして登場したFlexibleServerと
  SingleServerとの比較を目的にした机上/実機検証、HyperScaleの実機検証
  を実施
  - ※ HyperScaleの実機検証は現在検証中の為、本資料に含めず

机上調査結果
（アーキテクチャ紹介）

AzureにおけるPostgreSQL PaaS

- Azure Database for PostgreSQLでは、
  従来の2つのデプロイオプションに加え
  新オプションが追加されたため、機能調査を実施
  - SingleServer
  - Hyperscale (Citus)
  - FlexibleServer
    - SingleServerの強化版
      アーキテクチャから変更が入り、機能面が向上

- 2021/3時点でFlexibleServerは
  パブリックプレビュー（東日本リージョンで試用可）

サービス・アーキテクチャ概要 (FlexibleServer)

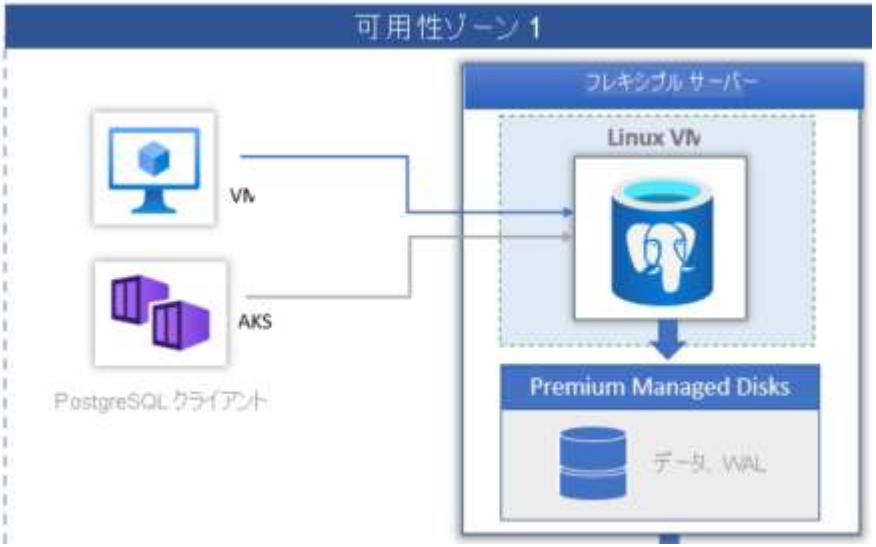
- Linux上のDockerコンテナ上で動く、PostgreSQLのマネージドサービス
  - SingleServerはService Fabric上のWindowsコンテナ

- PostgreSQLは、公開されているものをカスタマイズ無しでそのまま利用している
  - 2021年3月現在で、12.4,11.9が利用可能
    - マイナーバージョンは指定不可能

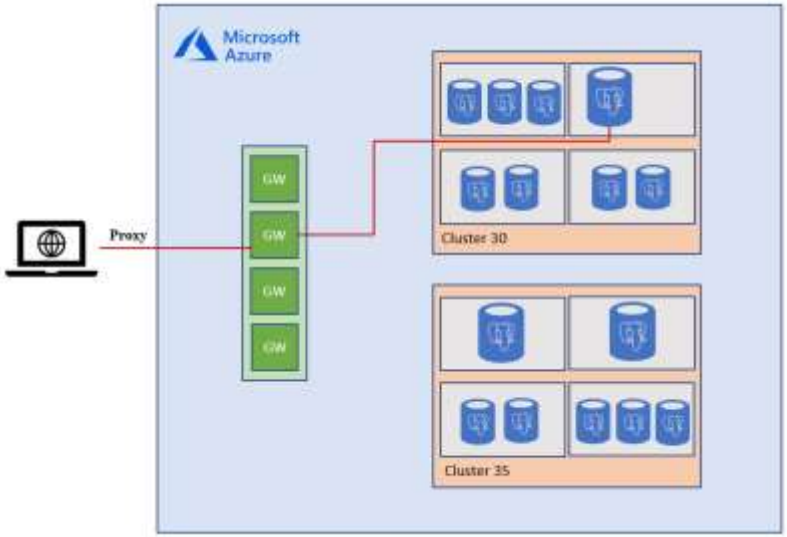
FlexibleServerの特徴①

- DBアクセスの際、エンドポイントからの接続が不要となり、<u>PostgreSQLサーバへの直接接続が可能に</u>
  - SingleServerはエンドポイント（GW）経由の接続が必須

FlexibleServer



SingleServer



出典：https://docs.microsoft.com/ja-jp/azure/postgresql/flexible-server/overview
https://docs.microsoft.com/ja-jp/azure/postgresql/concepts-connectivity-architecture

FlexibleServerの特徴②

- 接続周りがSingleServerと比べてシンプルに
  - ”@”を用いたホスト名の指定が不要
    - FlexibleServer:
      - psql “host=<ホスト名> dbname=<DB名>
        user=<ユーザ>　password=<パスワード>
    - SingleServer:
      - psql “host=<ホスト名> dbname=<DB名>
        user=<ユーザ>@<ホスト名>　password=<パスワード>“
  - Private IPをインスタンスに付与し接続が可能（Vnet統合）
    - SingleServerがインスタンス毎にPrivateEndpointの作成が必要

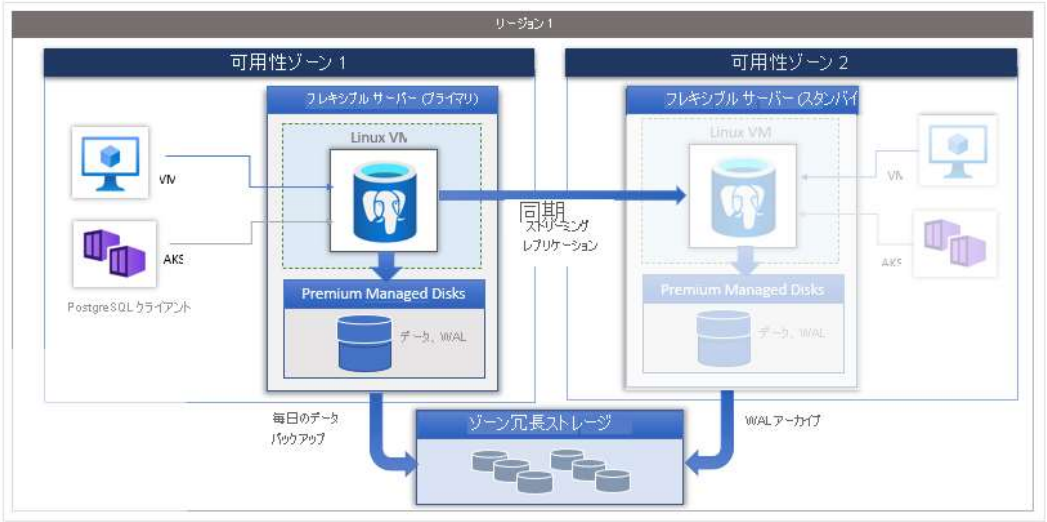
※Vnet統合の際にはFlexibleServer専用のsubnetを作成する必要があり、
subnet内に別サービスのインスタンスは作成不可になることに注意。

FlexibleServerの特徴③

- ゾーン冗長構成が使用可能
  - SingleServerは不可であり、リストアなどの復元が必要

- ただし、リードレプリカ作成、およびDR対策としてのGEO冗長バックアップは2021/3時点では不可
  - SingleServerはどちらも可能

ゾーン間のデータ同期は
PostgreSQLのストリーミング
レプリケーション機能を使用。



出典：https://docs.microsoft.com/ja-jp/azure/postgresql/flexible-server/overview

FlexibleServerの特徴④

- インスタンスモデルに”バースト可能”が登場
  安価かつ突発的なアクセスに対応可能
  - **バースト可能**：SingleServerのBasicの代替　1-2vCPU
  - 汎用：一般的なワークロード用　2-64vCPU
  - メモリ最適化：“汎用”のメモリサイズを増加　2-64vCPU

- ストレージも512GB以下を選択した場合、
  IOPS上限がバースト可能
  - 例: 256GBのストレージ
    - 通常時：1100IOPS　→ 3500IOPSまでバースト可能

机上調査成果物のご紹介

- FlexibleServer,SingleServerの機能比較（非機能面）は比較表を作成し、今年度成果物として公開中

項目	比較の観点	FlexibleServer	SingleServer
継続性	復元可能な直近の時間は何分前か	障害発生の5分前まで復元出来ることを保証	
	HA構成における切り替え時間の目安	ゾーン冗長で構成され、障害時は別ゾーンのレプリカが昇格する フェイルオーバーの時間は60-120秒程度の予測	ダウンタイムの目安（数値）について公開されている情報は無いが、対象インスタンスが稼働するコンテナを別のホストで起動し直し、GWのルーティングを変更する形での切替となるため、短時間での切替は可能とのこと
	SLAとして年間何%の稼働率を規定しているか	パブリックプレビュー時点では、SLAは無し ※GA後においても、SingleServerの99.99%よりは下がらない見込み	99.99% （SLAとして公表）
耐障害性	HAとしてサーバを多重化する仕組みは何があるか	ゾーン冗長構成をONにすると、自動的に別の可用性ゾーンにスタンバイレプリカが構成される データはPostgreSQLのストリーミングレプリケーション機能によって同期	コンテナ層は多重化されておらず、設定変更等で多重化させることも出来ない。そのため、障害時には別ノードでの再起動となる（再起動は自動で実行される）
	データの多重化の仕組みとして何があるか	実データが格納されるストレージ層は、データ同期方式で三重化されている	
災害対策	広域圏災害における、DRの方式は何が用意されているか	・ゾーン冗長による別ゾーンへの自動フェイルオーバが可能 ・GEO冗長バックアップは現状未サポートなため、別リージョンへのリストアはpg_dumpを使用するなど、利用者側で仕組みを検討する必要あり	・リードレプリカを別リージョンに作成しておき、有事の際には手動で切り替える（自動切替機能は未実装） ・GEO冗長バックアップをONにし、有事には別リージョンでインスタンスを手動リストアする

FlexibleServer性能検証結果

FlexibleServer性能検証概要

- Azure Database for PostgreSQLの新サービスである
  FlexibleServerに対し、昨年度同様、以下の性能検証を実施
  - 簡易なクエリでの参照/更新性能の確認（pgbench:カスタムクエリ）
  - 実際のワークロードを想定したスループット性能の確認（HammerDB）

- 従来のサービスであり、近いモデルのSingleServerとの
  比較を実施し、各サービスの特徴を確認

- 実際の計測条件やパラメータ等は成果物として公開中

FlexibleServer性能検証（pgbench）

- pgbench測定の大方針
  - カスタムクエリを使用して、参照系/更新系の2つのベンチマークを実施
  - データスケールは固定（ScaleFactor:2000、2億件程度）
  - 同時接続数は1,4,16,32,64,96 と、順次上げていく
  - SingleServerとスループット等の比較を行い、性質・特徴を調査

- PostgreSQLバージョン
  - FlexibleServer,SingleServerで比較ができるver11で実施

  ※昨年度検証でのSingleServerはver10のため、ver11で計測し直し

- PostgreSQLパラメータ
  - work_mem といったHWスペックに依存しやすいパラメータは、DBインスタンスのスペックに合わせて適宜変更する

実機検証 使用インスタンス

項番	インスタンス	スペック
1	負荷掛け用マシン	E8as v4 (8 vCPU,64 GB RAM) OS: Windows
2	DBインスタンス:FlexibleServer メモリ最適化モデル	E8 v3 (8vCPU 64GB RAM) E32 v3 (32vCPU 256GB RAM) ストレージ256GB (1100 IOPS)
3	DBインスタンス:SingleServer メモリ最適化モデル	Gen5 8vCPU 80GB RAM Gen5 32vCPU 320GB RAM ストレージ377GB (1101 IOPS)

検証方法

- pgbench（参照系）検証
  - PGEConsで実施している検証方法を踏襲
    - 初期化
      - pgbench -i -s 2000 [dbname]
    - キャッシュへの読み込み
      - SELECT pg_prewarm('pgbench_accounts');
    - カスタムスクリプト

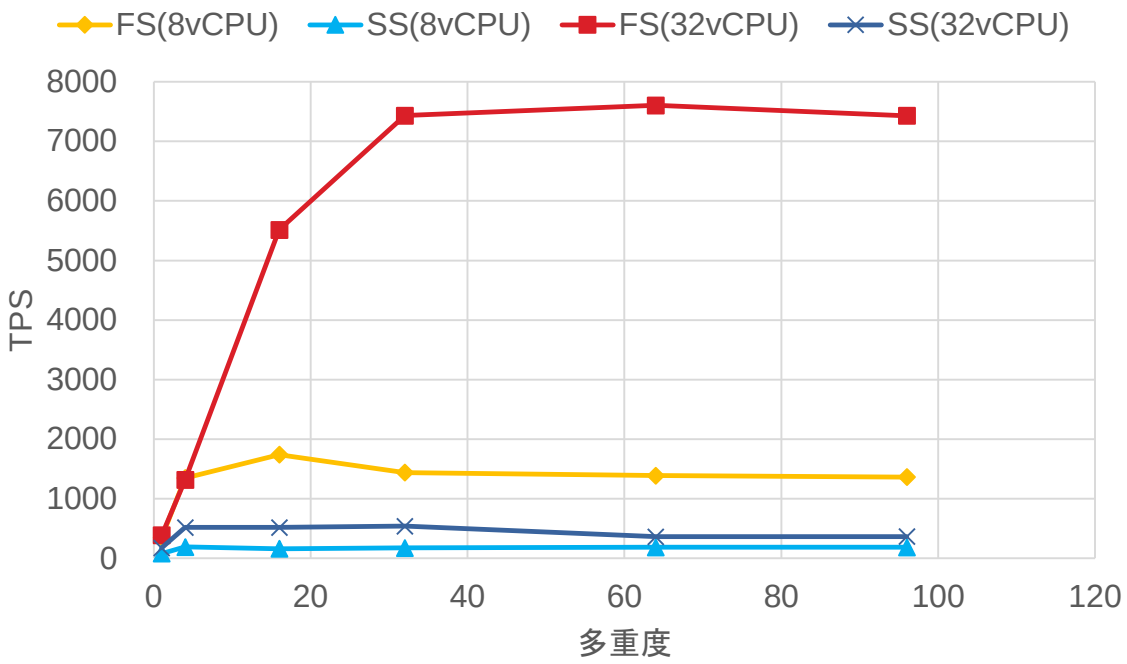
```
¥set naccounts 100000 * :scale
¥set row_count 10000
¥set aid_max :naccounts - :row_count
¥set aid random(1, :aid_max)

SELECT count(abalance) FROM pgbench_accounts WHERE aid BETWEEN :aid and :aid + :row_count;
```

    - 実行

```
pgbench -r -P 1 -n -c [clients] -j [threads] -f [カスタムクエリ] -T 300 -s 2000
-h [host] -p [port] [dbname]
```

検証結果:参照系 (TPS)

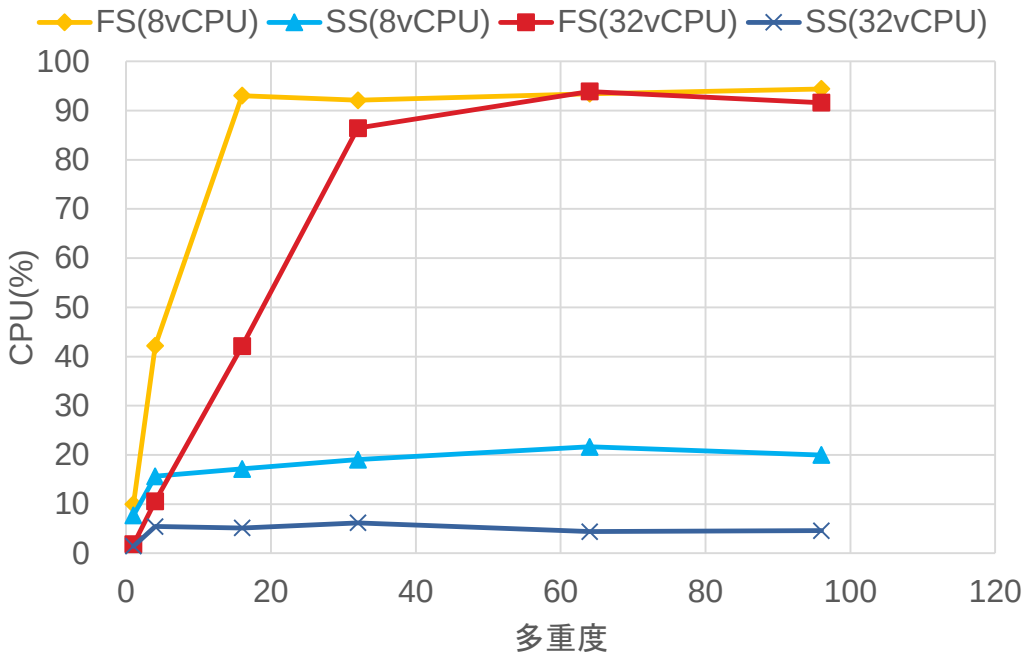


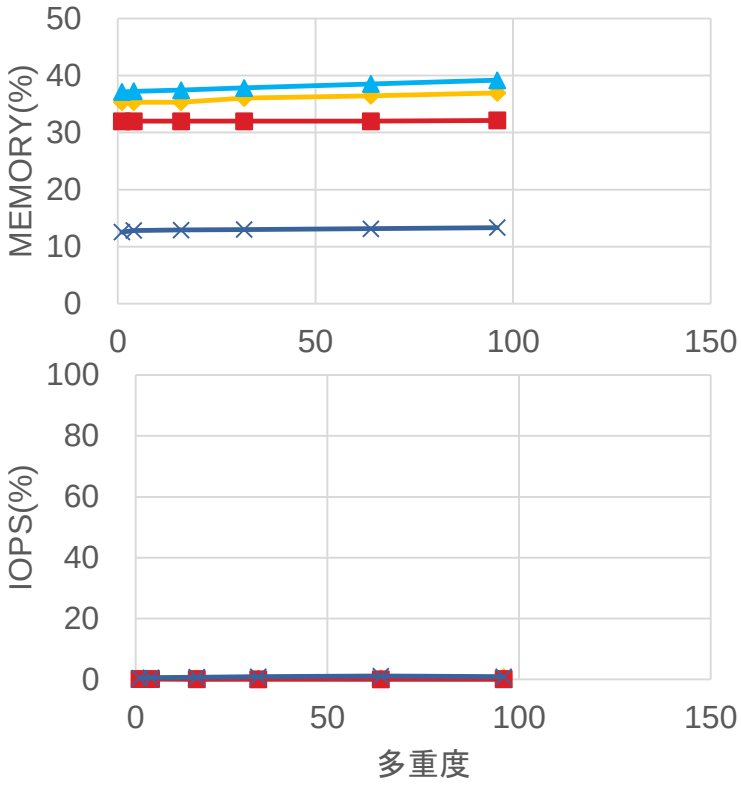
FlexibleServer:FS
SingleServer:SS

■ 傾向

- □ 8コア、32コアともに全ての多重度において、
  FlexibleServerの方がSingleServerより高いスループット (TPS) を発揮
- □ FlexibleServerはコア数増加に伴いTPSが増加するが、
  SingleServerはコア数を増やしてもTPSはほぼ変化なし

検証結果:参照系 (CPU,Memory,IOPS)



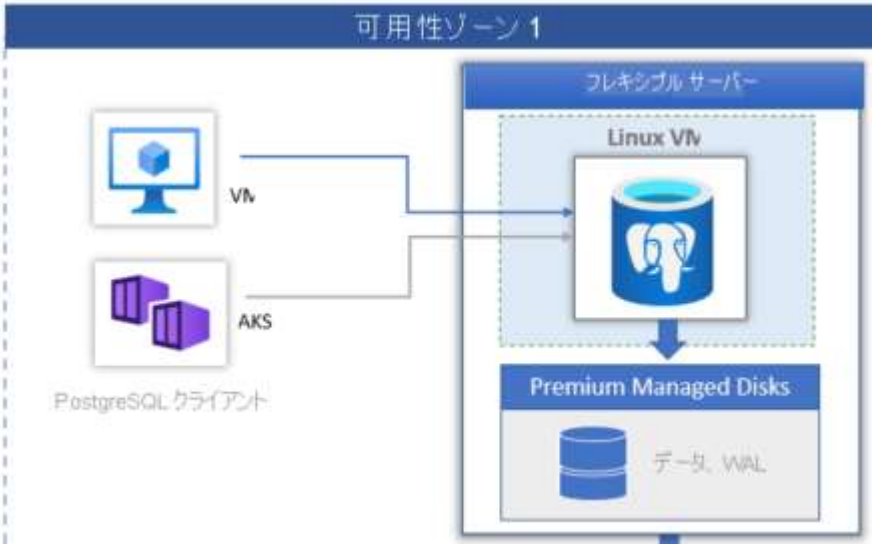


- 傾向
  - FlexibleServerはコア数×2の多重度でCPU利用率が100%近くに上昇
    - Memory,IOPSは使い切れておらず、CPUネックでTPSが頭打ちになったと考えられる。
  - SingleServerはCPU,Memory,IOPSを使い切れていない
    - DBのリソースを使い切れてないことから、AzureのNW構成の仕様に起因していると推測

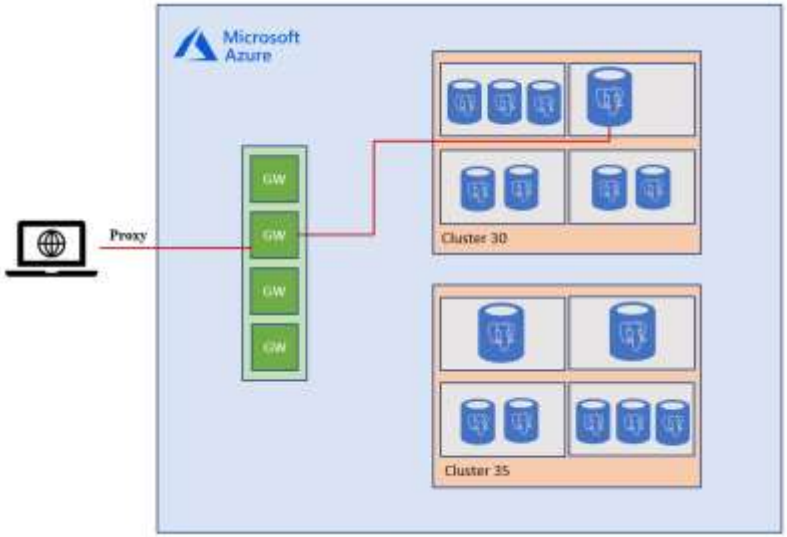
参照系:レイテンシ原因考察

- アーキテクチャ上、SingleServerはDB接続の際にGatewayを経由するが、FlexibleServerは直接接続であるため、Round-Trip Time (RTT) の差が大きい
  - 成果物には追加検証も実施

FlexibleServer



SingleServer



出典:https://docs.microsoft.com/ja-jp/azure/postgresql/flexible-server/overview
https://docs.microsoft.com/ja-jp/azure/postgresql/concepts-connectivity-architecture

ここを経由するため、
DB到達に時間がかかる

性能テスト結果（pgbench）
更新系処理

検証方法

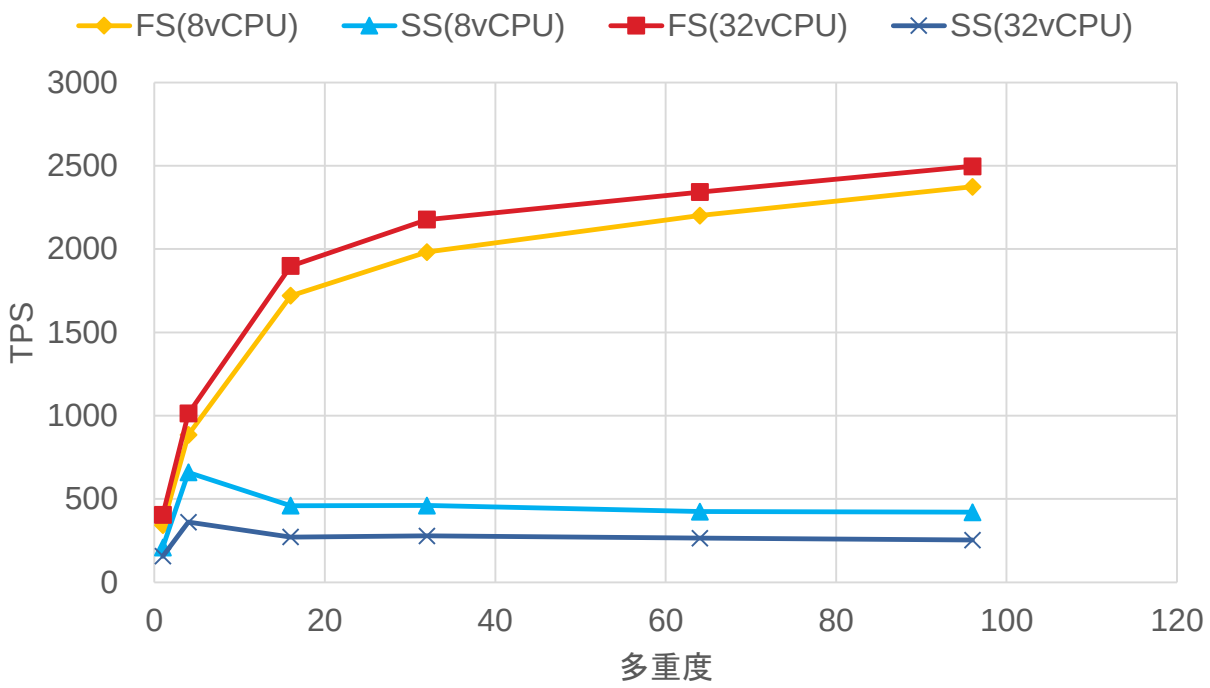
- pgbench（更新系）検証
  - PGEConsで実施している検証方法を踏襲
  - 初期化
    - pgbench -i -s 2000 [dbname] -F 80
  - キャッシュへの読み込み
    - SELECT pg_prewarm('pgbench_accounts');
  - カスタムスクリプト

```
¥set naccounts 100000 * :scale
¥set aid_val random(1, :naccounts)
UPDATE pgbench_accounts SET filler=repeat(md5(current_timestamp::text),2) WHERE  aid= :aid_val;
```

  - 実行

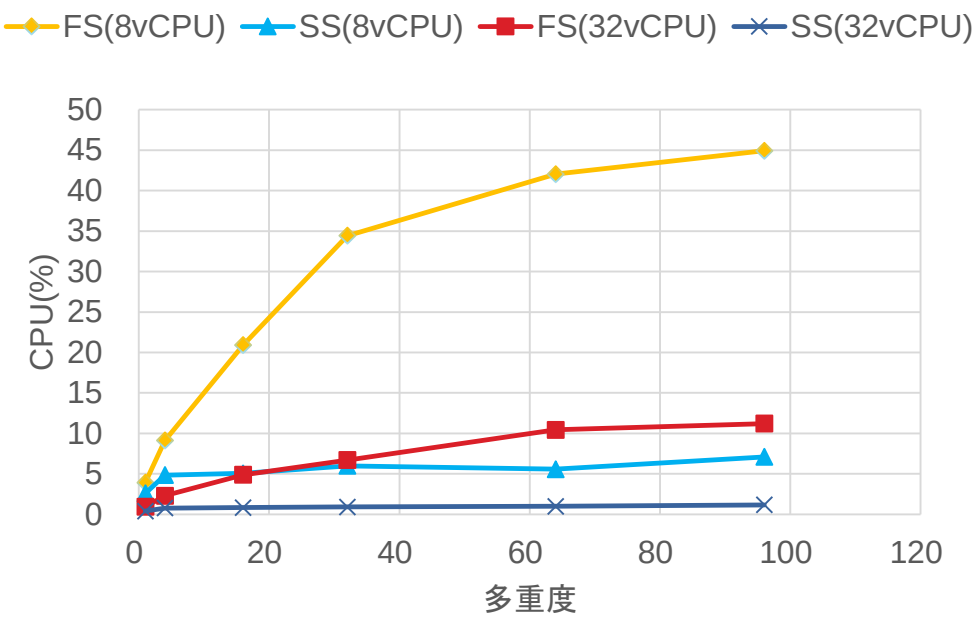
```
pgbench -r -P 1 -n -c [clients] -j [threads] -f [カスタムクエリ] -T 300 -s 2000
-h [host] -p [port] [dbname]
```

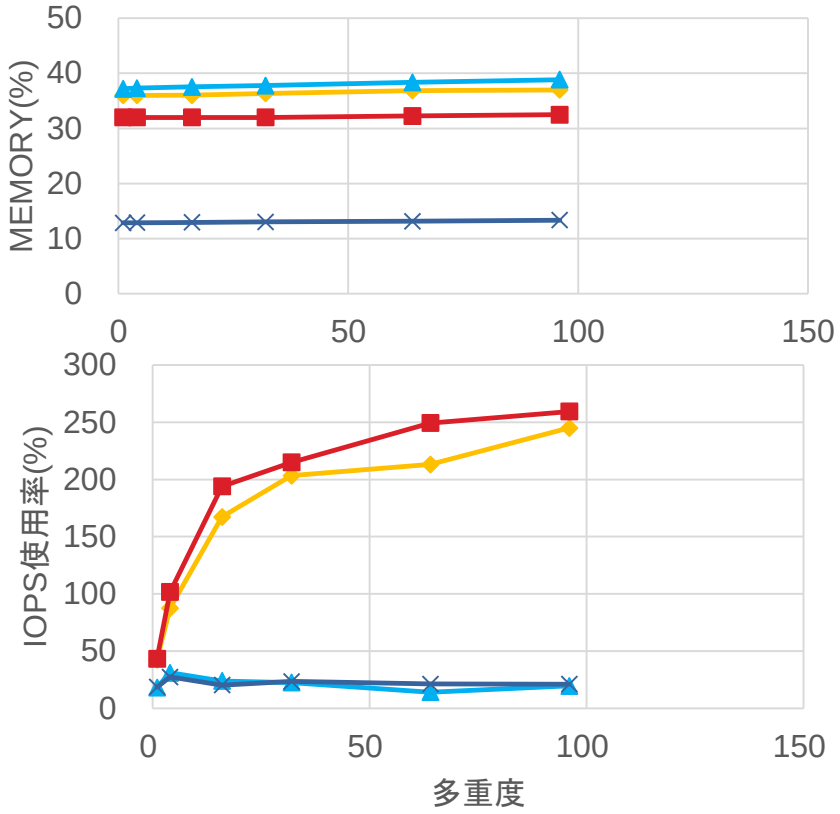
検証結果:更新系 (TPS)



- **傾向**
  - 8コア、32コアともに全ての多重度において、
    FlexibleServerの方がSingleServerより高いスループット (TPS) を発揮
  - FlexibleServer,SingleServerのどちらもvCPUコア数を増加させてもTPSはほぼ変化せず、特にSingleServerはvCPUコア数を増やすと低いTPSとなった
    - 参照系で確認したRTTの差が原因だと考えられる

検証結果:更新系 (CPU利用率)





- 傾向
  - FlexibleServerはIOPS使用率が100%を超えていた
    - IOPSバースト (1100→3500) が発生
  - SingleServerは参照系同様、リソースを使い切れず

更新系:IOPS考察

- FlexibleServerはIOPSがボトルネックとなり
  TPSが頭打ちとなっていると考えられるため、
  IOPSを1100→5000に増加させて追加計測を実施
  - FlexibleServer：256GB → 1TB
  - SingleServer：377GB → 1667GB

- その結果、FlexibleServerはTPSが向上した
  - IOPSがボトルネックになっていることを確認
  - SingleServerはTPSが変化せず、ボトルネックとは無関係

結果整理 (pgbench)

- FlexibleServerは今回のケースではSingleServerより良いスループットを発揮した。
  - 検証の中でCPUやIOPSが上限に達したが、リソースを増強すると性能を向上できることを確認

- 一方、SingleServerはAzureのNW構成の仕様に起因し、DB側のリソースを使い切ることが出来なかった。

FlexibleServer性能検証結果 (HammerDB)

HammerDBとは

- PostgreSQLを含む複数のDBMSに対応した
  ベンチマーク実行ツール
- OLTP/OLAP を模したベンチマーク実行が可能
  - TPC-C/TPC-H
  - 今回は TPC-C にて実施

HammerDBでの検証:環境

- 性能測定の 大方針/PostgreSQLバージョン/負荷掛けマシンスペック はpgbenchに準じる
  - PostgreSQLパラメータはリソース作成時から変更なしとする
  - リソーススペックは以下とする
    - インスタンスタイプ: メモリ最適化
    - CPU: 8、32
    - ディスクIOPS（サイズ）:
      - FlexibleServer: 1,100（256GB）
      - SingleServer: 768（256GB）
  - 実施する多重度は以下とする
    - 1、4、8、16、32、48、64、80、96、112、128、144
      - 8vCPU は性能の頭打ちが見えたため 96までとした

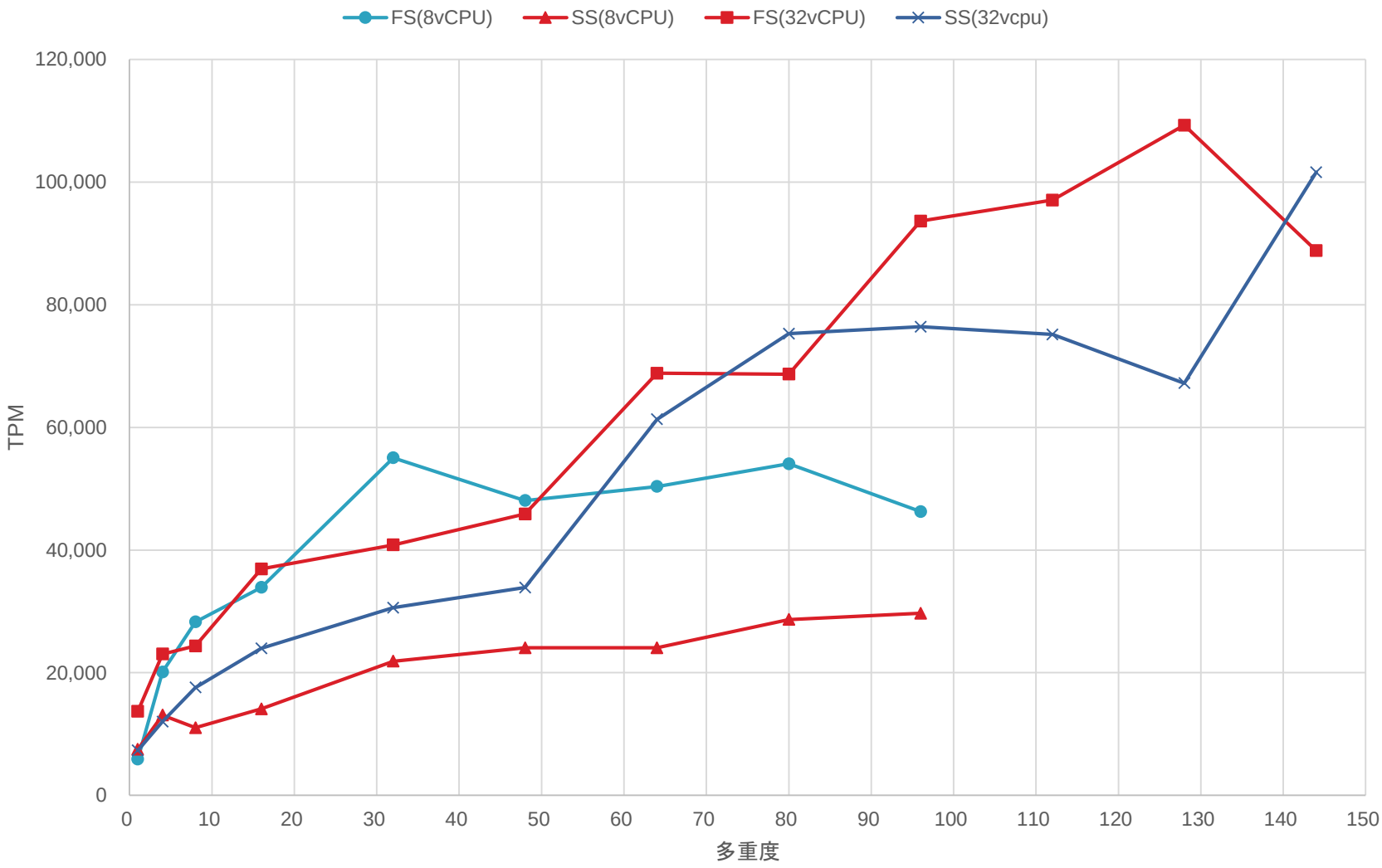
HammerDBでの検証:設定

- HammerDBの主な設定箇所
  1. ベンチマーク方式は TPC-C とする
     - 設定コマンド: dbset bm TPC-C
  2. データサイズは昨年実施を踏まえた設定とする
     - 設定コマンド: diset tpcc pg_count_ware 200
       - データベースサイズ: 40GB
  3. 処理対象を特定データのみとしない
     - 設定コマンド: diset tpcc pg_allwarehouse true
       ※デフォルトはfalse(特定データのみを処理対処とする)
  4. 検証開始直後のデータ読込によるIO影響を避けるため開始3分は計測対象としない
     - 設定コマンド: diset tpcc pg_rampup 3

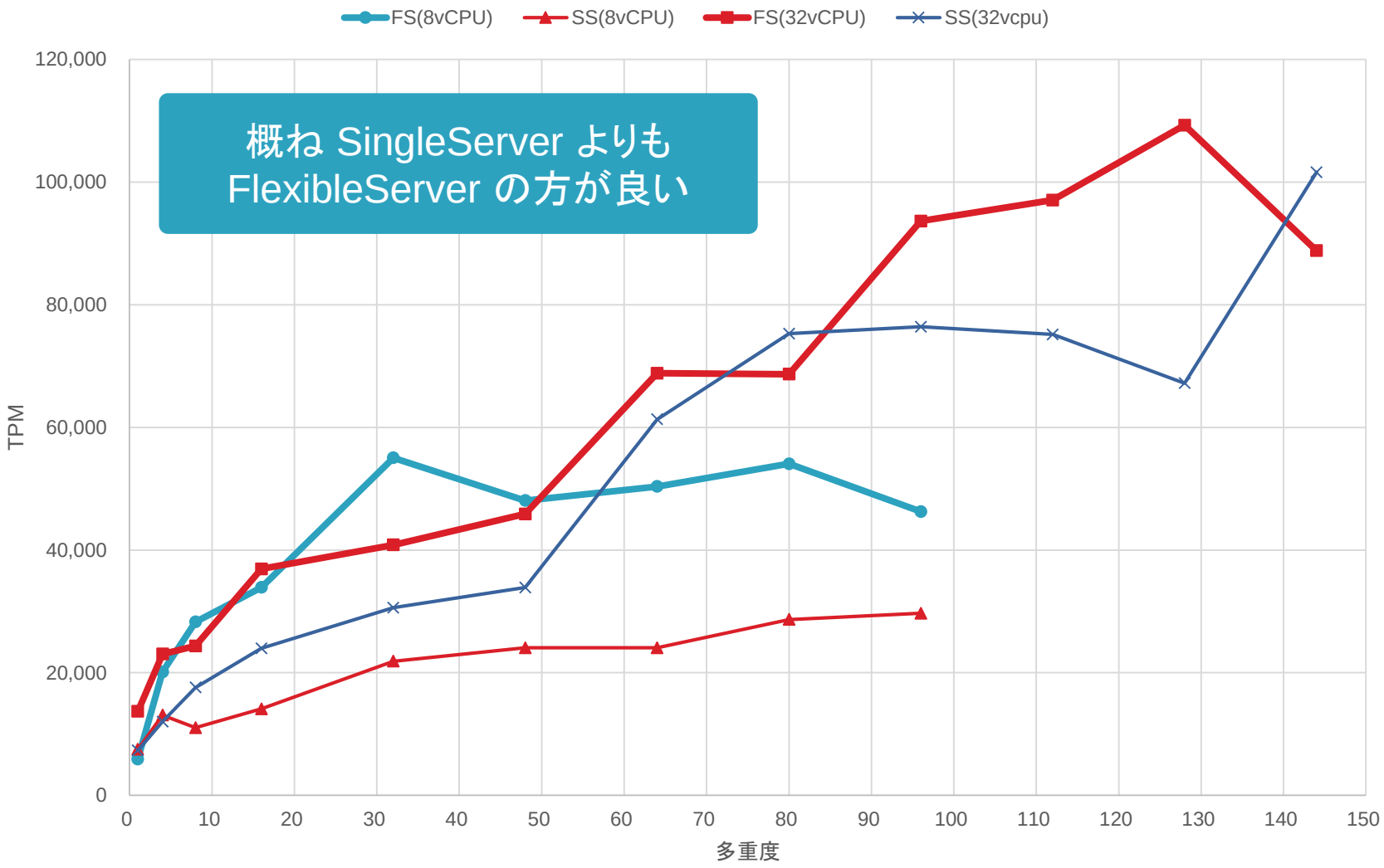
HammerDBでの検証:手順

- ベンチマーク処理には更新が含まれるため
  試行ごとにデータを初期化する
- 構築した直後の環境を退避した後
  以下の手順を実施
  1. データ投入、バキューム処理
  2. ベンチマーク実施/測定
  3. データ削除

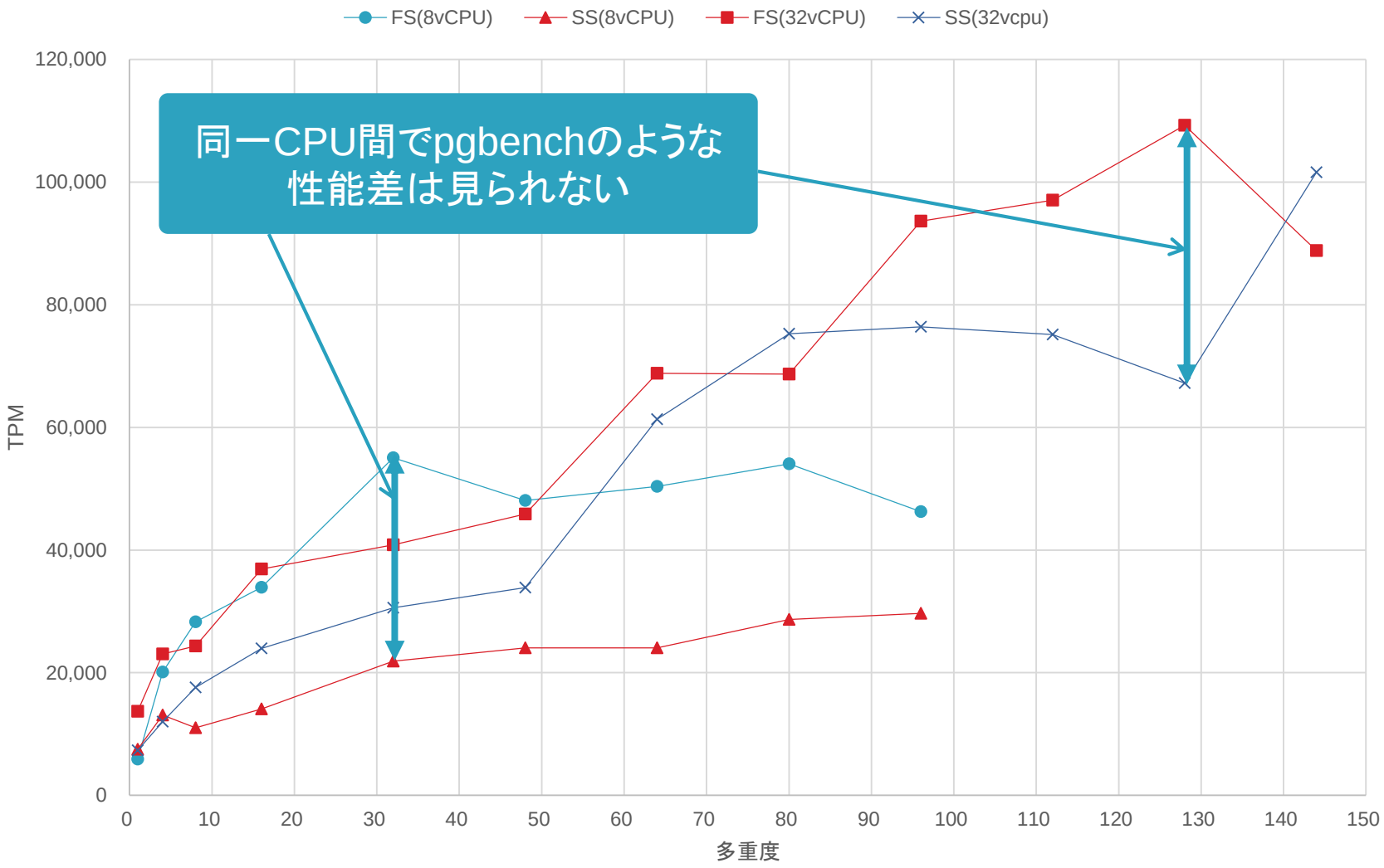
検証結果:性能



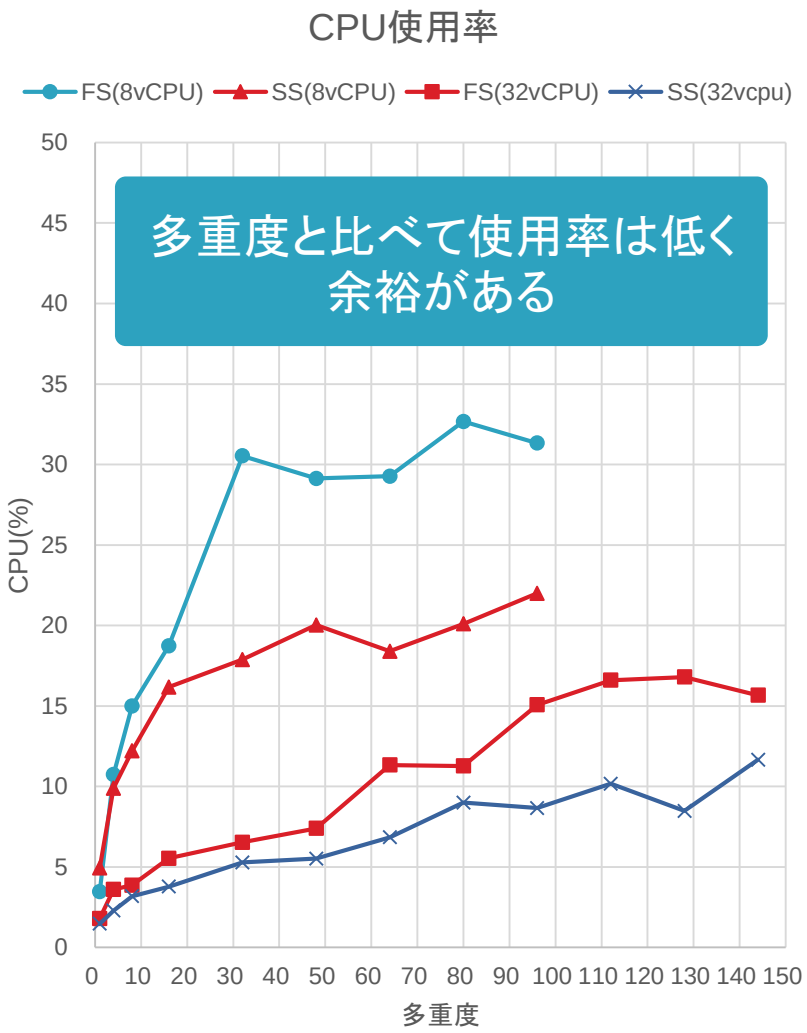
検証結果:性能

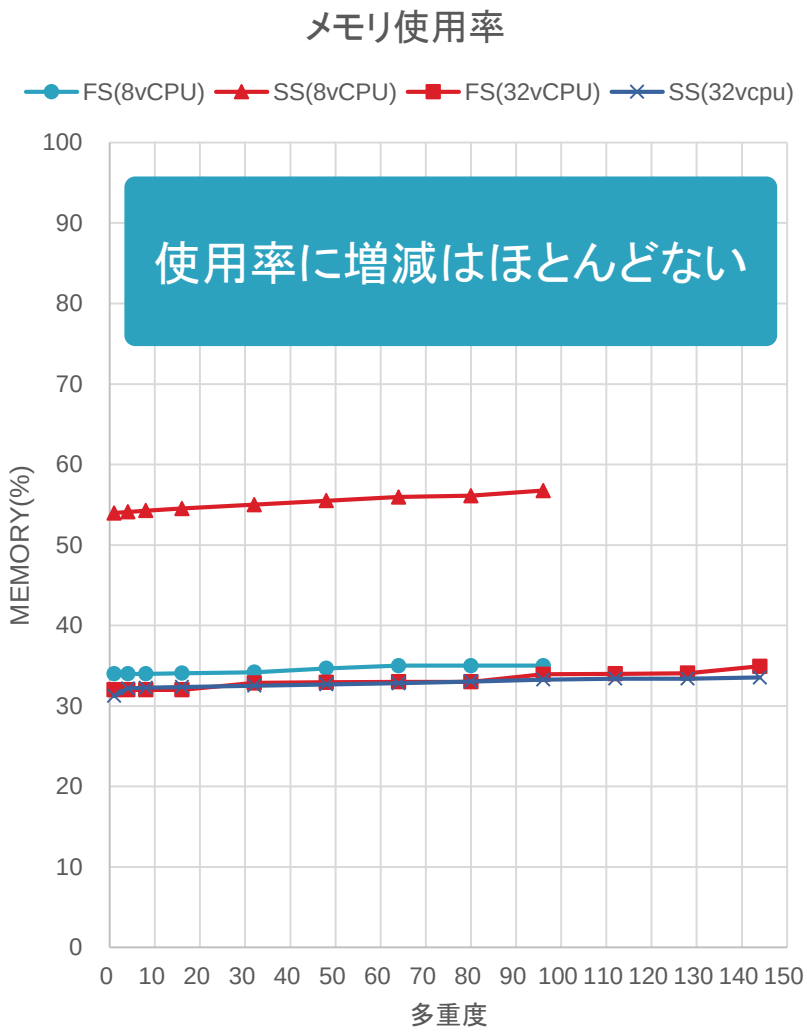


検証結果:性能

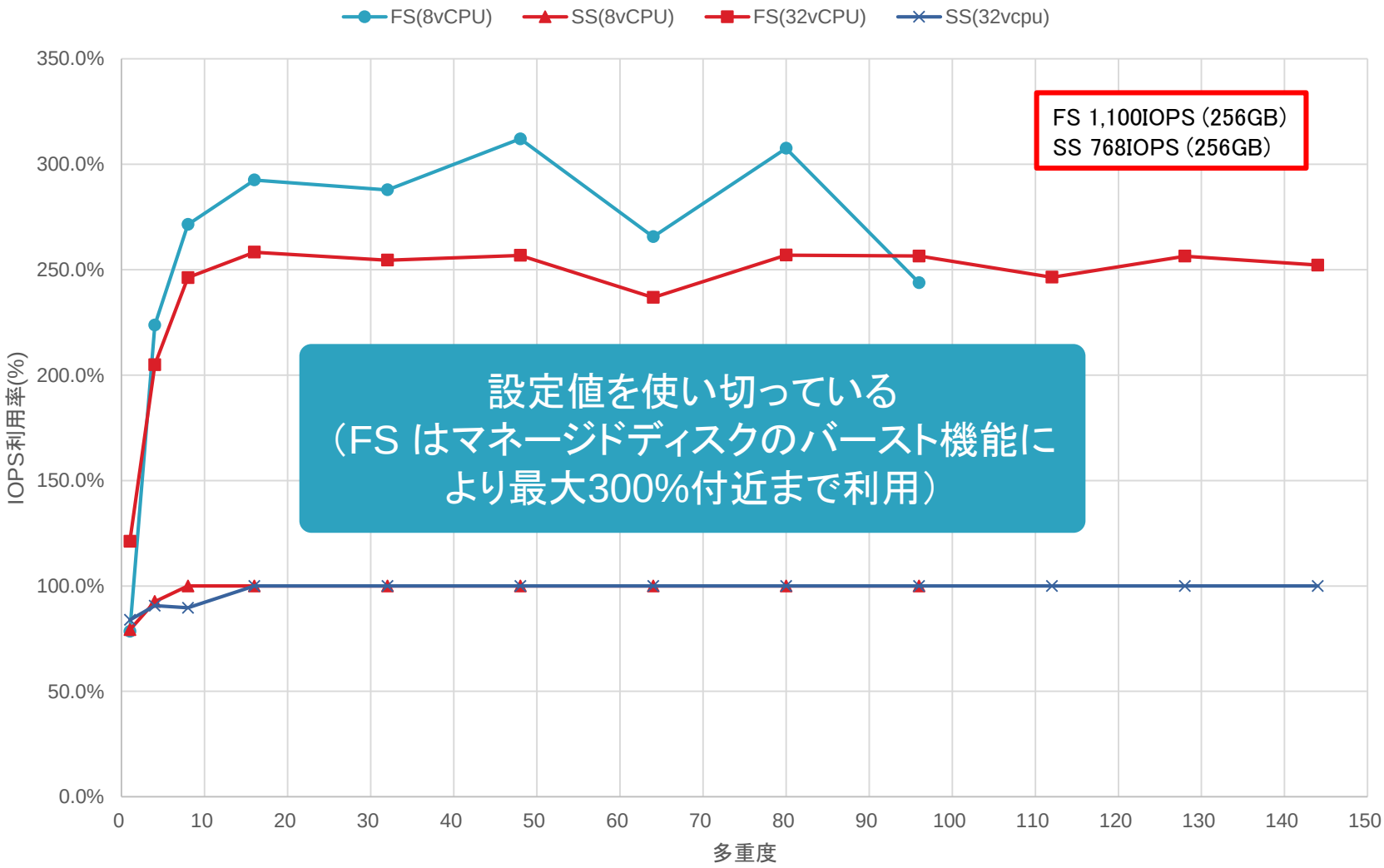


検証結果:CPU/メモリ利用率





検証結果:IOPS利用率



HammerDB結果考察

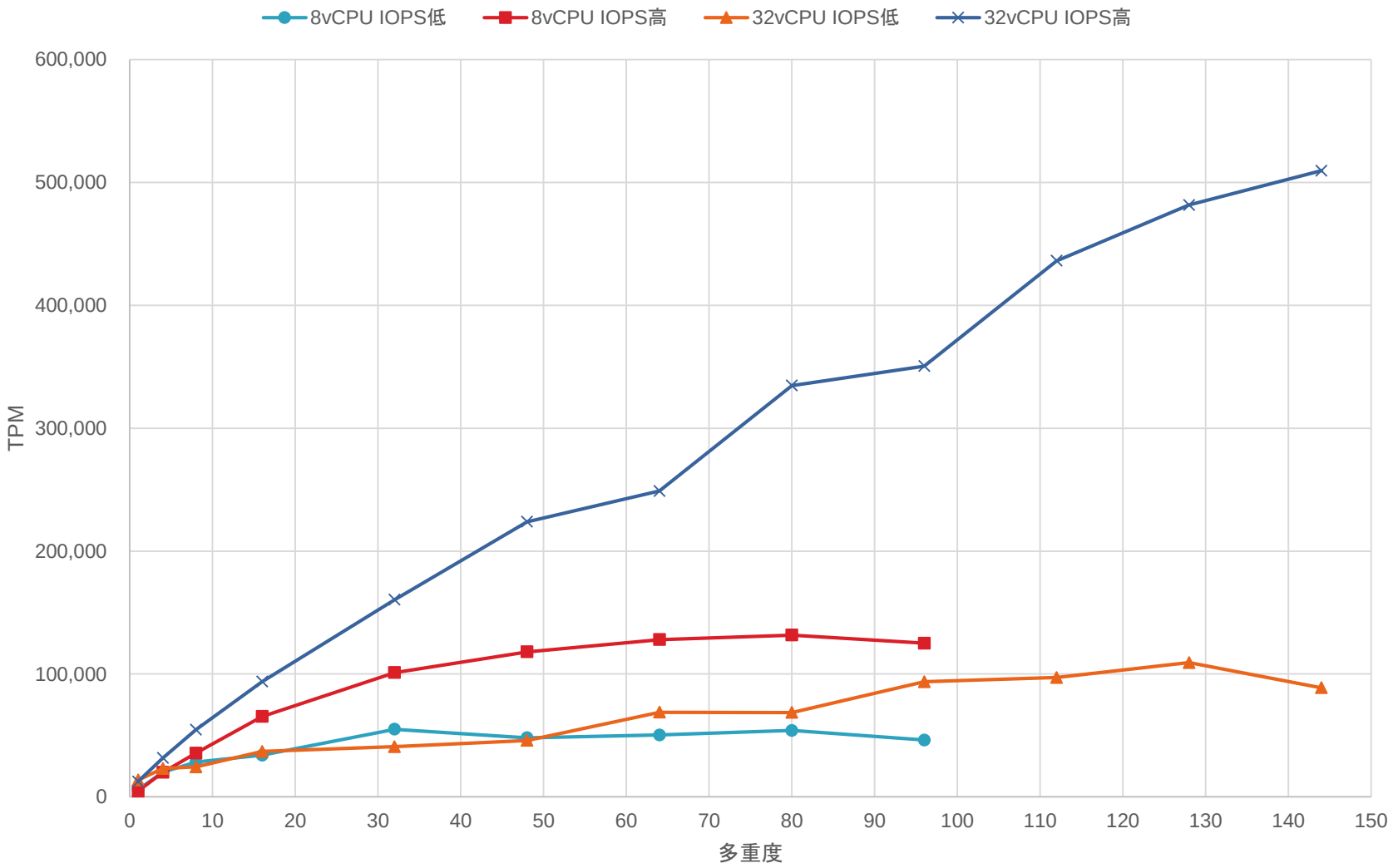
- 今回のリソースでの性能ボトルネックはIOと推測される
  - CPUの利用率が低く、またメモリの利用率は一定であるが IOの利用率が100%以上となっている
- FlexibleServer の方が SingleServer よりも 性能が良いがIOを使い切るワークロードの場合 pgbenchほどの差がない
  - pgbenchでは処理時間が極小のためRTTの影響を受けたが HammerDBは相対的に複雑なSQLを発行しており 処理時間が長くなる※ことでRTTの影響を受けにくいと推測

※: HammerDBではSQL処理時間を計測する機能を持たないため推測となる

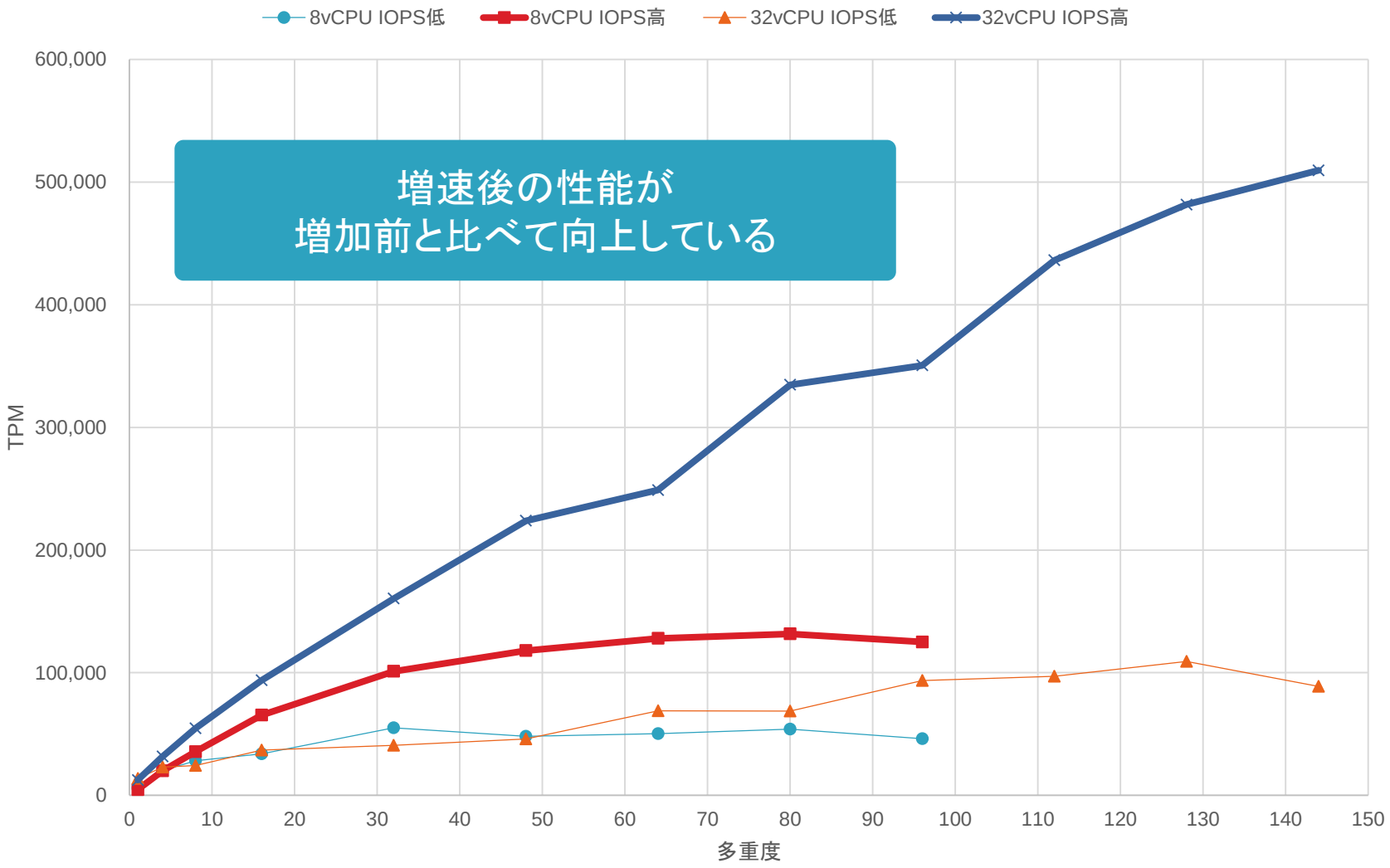
追加検証

- IOPSを増加した場合の性能向上を検証する
  - IOPSがバースト後含め上限に達しており
    性能のボトルネックになっていると推測
  - IOPSを以下の数値に増加することで
    性能が向上することを確認する
    - FlexibleServer
      - 8vCPU:　1,100 ⇒ 12,800
      - 32vCPU:　1,100 ⇒ 18,000
    - SingleServer
      - 8vCPU:　768 ⇒　5,133
      - 32vCPU:　768 ⇒ 10,080

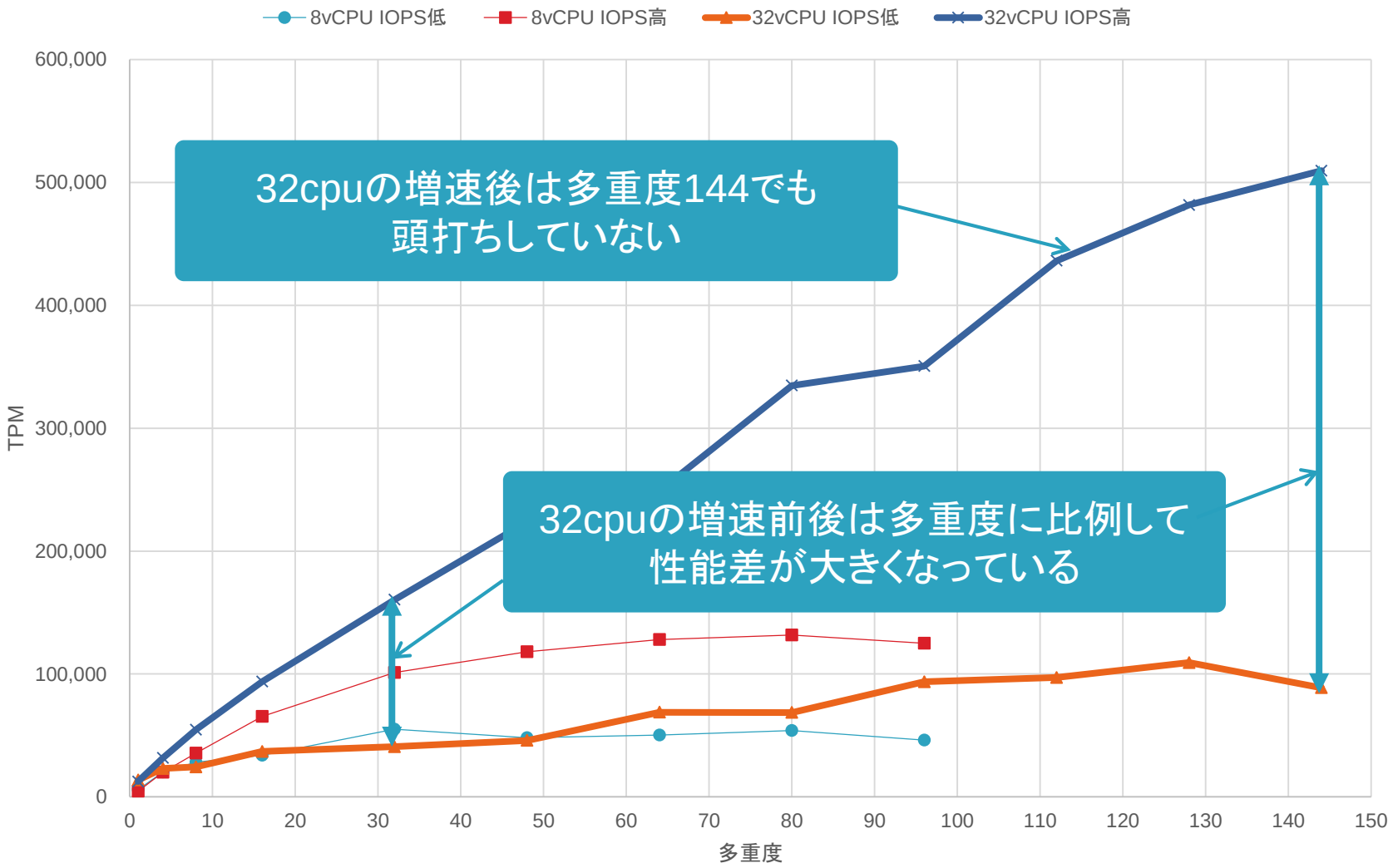
追加検証結果:性能 (FlexibleServer IOPS増速前後)



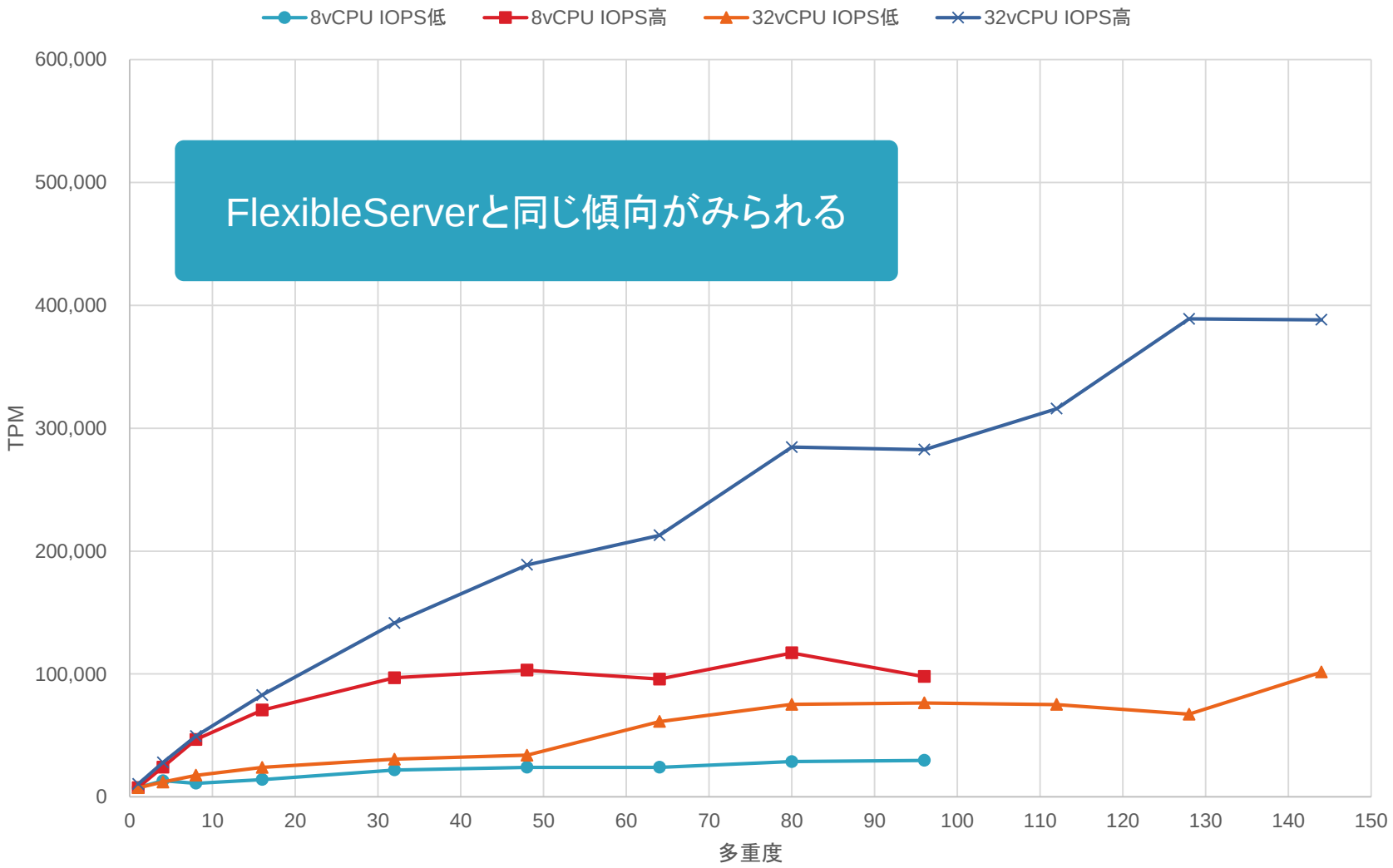
追加検証結果:性能 (FlexibleServer IOPS増速前後)



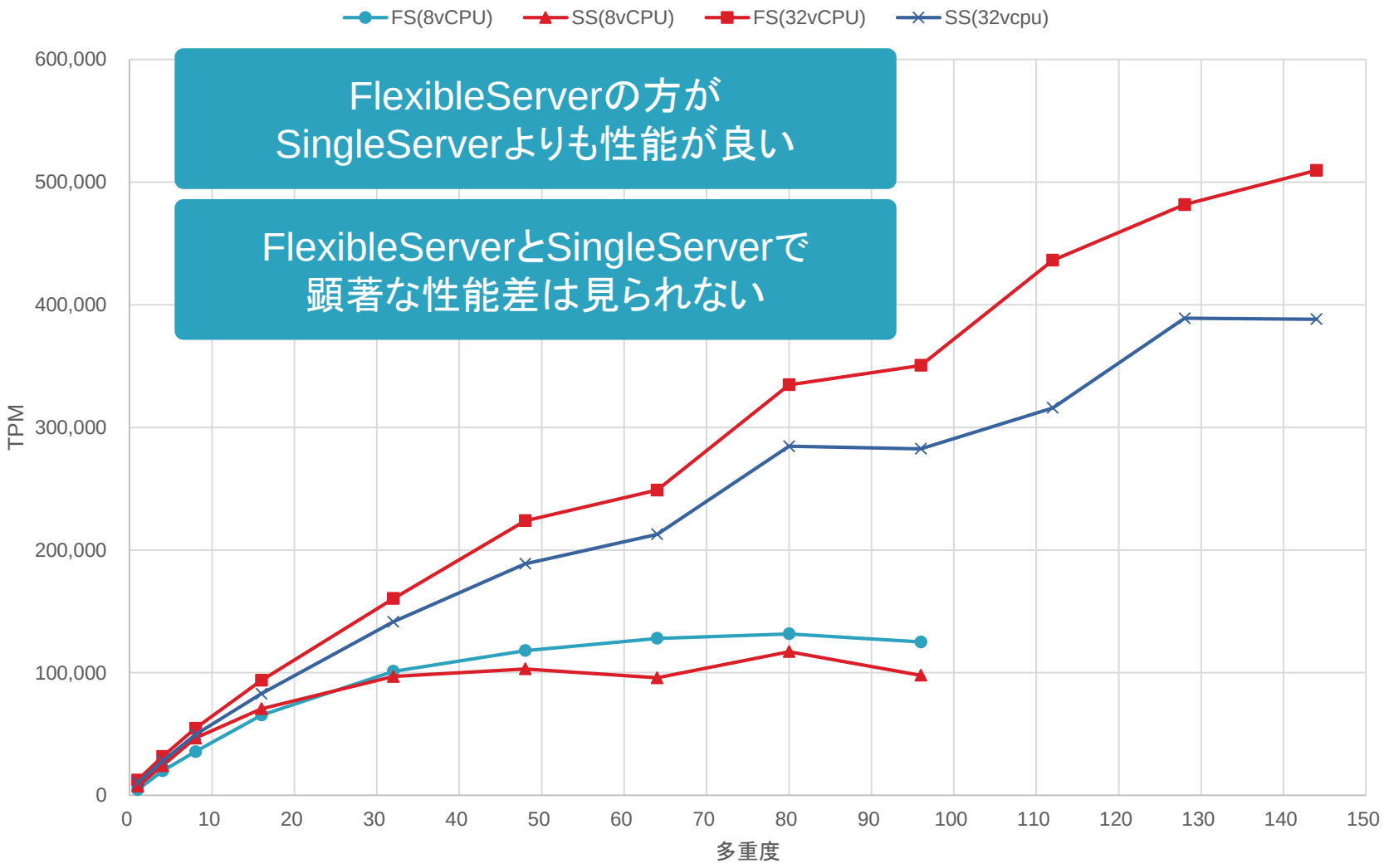
追加検証結果:性能 (FlexibleServer IOPS増速前後)



追加検証結果:性能 (SingleServer IOPS増速前後)



追加検証結果:性能 (IOPS増速後リソース間比較)



HammerDB追加検証結果考察

- 想定通りIOPS増速により性能が向上した
  - IOPS増速後もFlexibleServerの方がSingleServerより性能が良い
- FlexibleServerとSingleServerの間でpgbenchほどの性能差が見られなかった
  - IOPS増速により処理時間が短縮しRTTの影響を受けやすくなり性能差が大きくなると予想したがそうならなかった
  - HammerDB検証環境で各サービスに対するRTTを実測すると差が見られなかった
    - Microsoft社のRTT測定ではFlexibleServerはSingleServerの1/5程度となるはず
    - 差がない理由は特定に至らず

検証結果総括

総括

- FlexibleServer の方が SingleServer よりも性能（スループット）が良い結果を得られた
  - FlexibleServer は CPU や IOPS がボトルネックとなったがスケールアップさせることによって性能が向上した
  - リソースがバーストすることで想定以上のスループットとなる場合があるが実際の伸び幅については注意
- 『処理時間が極小なクエリ×多数の同時アクセス』のワークロードにおいて FlexibleServer の方が性能を得やすい
  - SingleServer はDB接続周りの仕様上FlexibleServer と比べるとRTTが長い

補足：RTT (Round-Trip Time) が与える FlexibleServerとSingleServerの性能差について 1/2

- RTTは“FlexibleServer < SingleServer”となる
  - サービスによりネットワーク構成に差がある
    - クライアントからの接続時、SingleServerはGWを経由するがFlexibleServerは直接接続する
    - SingleServerは可用性ゾーンを指定できない（ランダムとなる）
  - Microsoftにて公開されているRTT※
    - SingleServer:　2.76ms
    - FlexibleServer:　0.54ms

※Modernize your data on Azure Database for Postgres & MySQL | Data Platform Summit 2020 | Sunil Agarwal & Parikshit Savjani
(https://speakerdeck.com/azuredbpostgres/modernize-your-data-on-azure-database-for-postgres-and-mysql-data-platform-summit-2020-sunil-agarwal-and-parikshit-savjani?slide=15)

補足：RTT (Round-Trip Time) が与える FlexibleServerとSingleServerの性能差について 2/2

- SQL実行に対するRTTの影響
  - SQL実行時のネットワークパケットの流れは右図の通り
    そのため、SQL実行のレイテンシは最低でも以下の時間を要する

    RTT + サーバ側のSQL処理時間 (右図③)

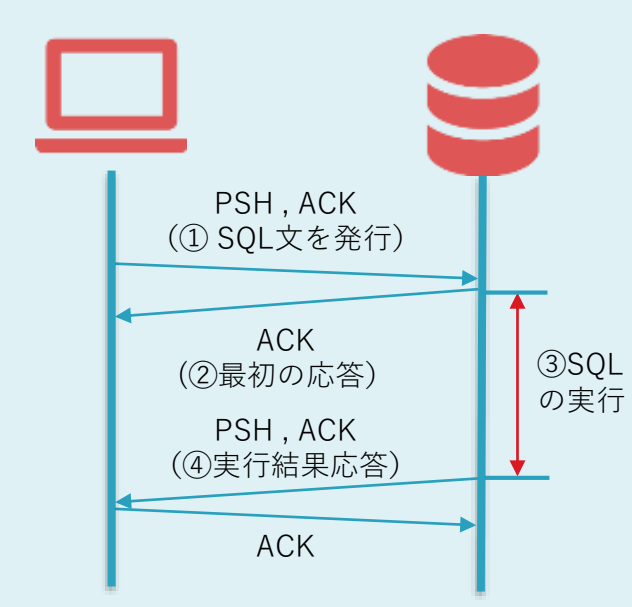


図: ネットワークパケットの流れ

- RTTが短かければ単位時間あたりのSQL実行回数の上限※は多くなり性能が向上する
  - SingleServer (RTT=2.76ms) : 362回/秒 (1/2.76×1,000)
  - FlexibleServer (RTT=0.54ms) : 1,851回/秒 (1/0.54 ×1,000)
  - 接続を多重化すれば性能は向上するが 多重化には限界があり特定の多重度以上は逆にスループットが下がってしまう事が知られている

※ “サーバ上でのSQL処理時間がほとんど0である”と仮定した場合

PGECons

PostgreSQL Enterprise Consortium