PGECons
PostgreSQL Enterprise Consortium

2024年度WG1活動報告
定点観測（バージョン間性能比較）

PostgreSQLエンタープライズ・コンソーシアム
WG1（新技術検証WG）

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。

目次

検証方法

検証概要

目的

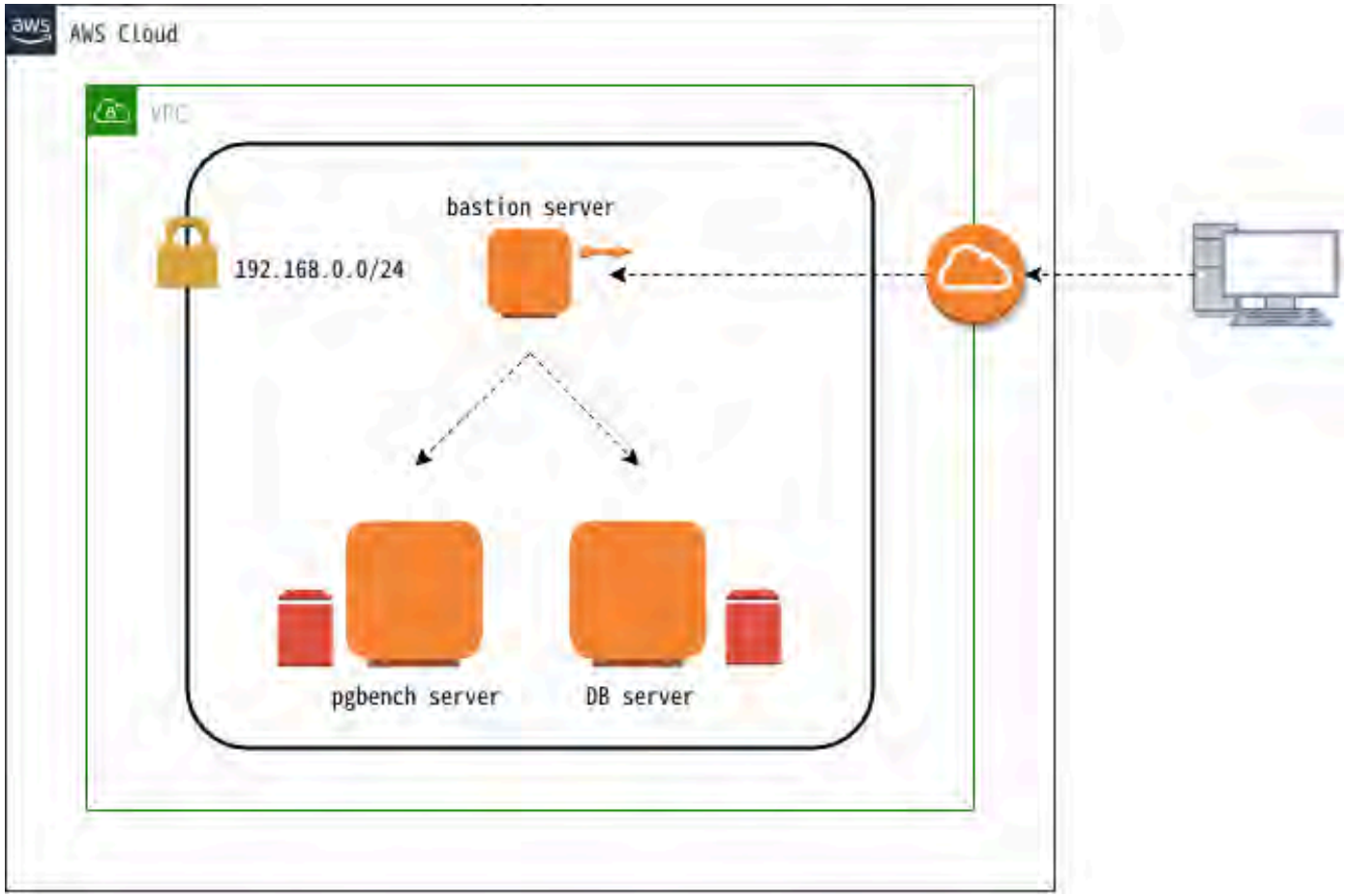
- メニーコアCPU上でのPostgreSQLのスケーラビリティを検証
- 新旧バージョンのPostgreSQLにおいてCPUマルチコアを活かして性能を出せるかどうか傾向を知る
  - PGECons発足当初（2012年度、PostgreSQL 9.2）から継続的に実施（定点観測）
    - 更新系性能に関する定点観測は2014年度から開始
  - 今年度はV16とV17の比較を実施

検証内容

- 参照性能
- 更新性能

検証環境

- Amazon Web Services (AWS)の仮想マシンを使用



検証環境

スペック

名称	インスタンスタイプ	仮想cpuコア数	物理cpuコア数	メモリ（GiB）	ルートストレージサイズ（GiB/IOPS）	追加ストレージサイズ（GiB/IOPS）
bastion server	t2.micro	1		1	10/100	
pgbench server	m5a.8xlarge	32	16	128	20/100	20/100
DB server	m5a.8xlarge	32	16	128	20/100	200/600

- 2012年度から同じCPUコア数
- メモリは試験用データが載るサイズを確保
- ストレージはonキャッシュで試験を実行するので最低限

ソフトウェア

名称	OS	PostgreSQL	pgbench
bastion server	RHEL 9.5		
pgbench server	RHEL 9.5		17.2
DB server	RHEL 9.5	16.6, 17.2	

検証環境

データベースサイズ

```
$ pgbench -i -s 2000 《データベース名》 -F 80
```

- pgbenchで30GBほどの試験用データベース(*1)を作成
- 更新系試験のみフィルファクタ80(*2)をオプション指定

postgresql.conf

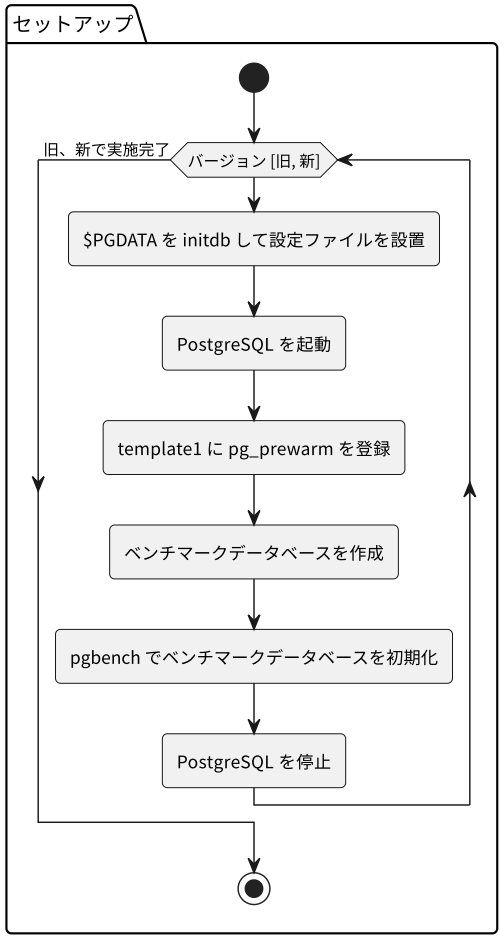
```
listen_addresses     = '*'   # クライアント用サーバからの接続用
max_connections      = 2000  # 頭打ちを観測するため、2021年度の500から増やした
shared_buffers       = 40GB  # 試験用データがすべてメモリに載るように設定
work_mem             = 1GB
maintenance_work_mem = 20GB
checkpoint_timeout   = 60min # 試験中にチェックポイントを発生させない
max_wal_size         = 160GB # 試験中にチェックポイントを発生させない
logging_collector    = on
log_checkpoints      = on
log_lock_waits       = on
autovacuum           = off   # 試験中にI/O処理を発生させない
```

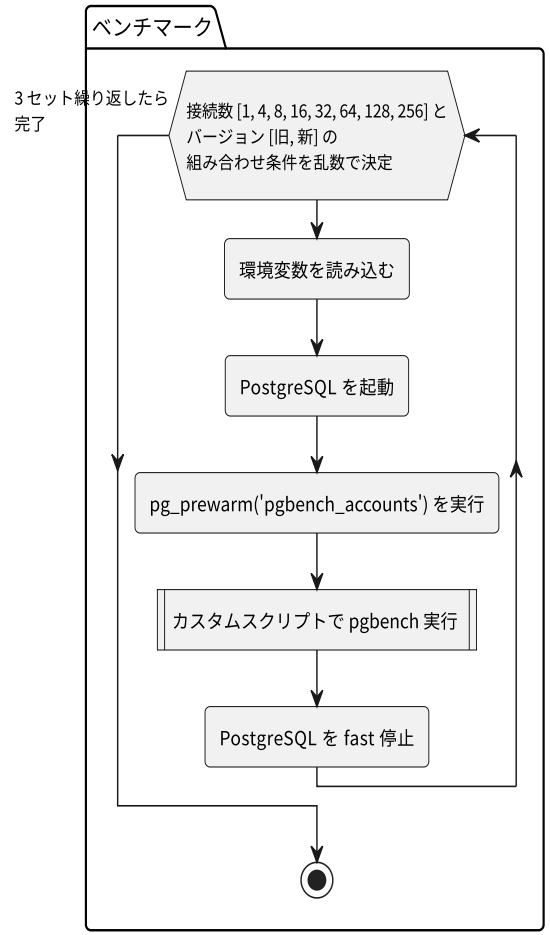
info

(*1)作成されるベンチマーク用標準テーブルはPGECons勉強会資料のPostgreSQL の標準的ベンチマークツール pgbench についてを参照。
(*2)テーブルブロックに空き領域を残して更新処理が効率よくできるようにしている。

参照系手続き

セットアップ後にベンチマークを実施





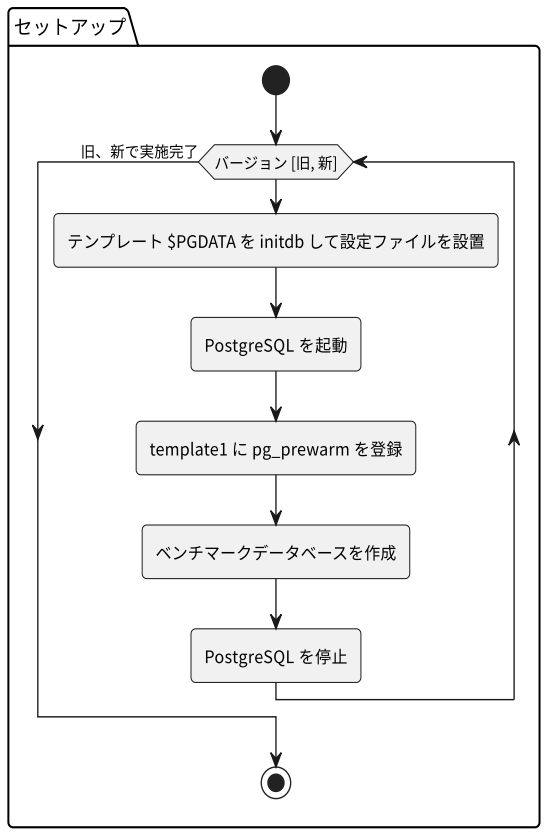
- PostgreSQLバージョン(2パターン)とクライアント同時接続数(8パターン)がTPSに及ぼす影響を計測
  - 1, 4, 8, 16, 32, 64, 128, 256
- 2×8=16パターンのベンチマークをランダムな順序で3回繰り返す
- 得られたTPSの中央値を結果として採用

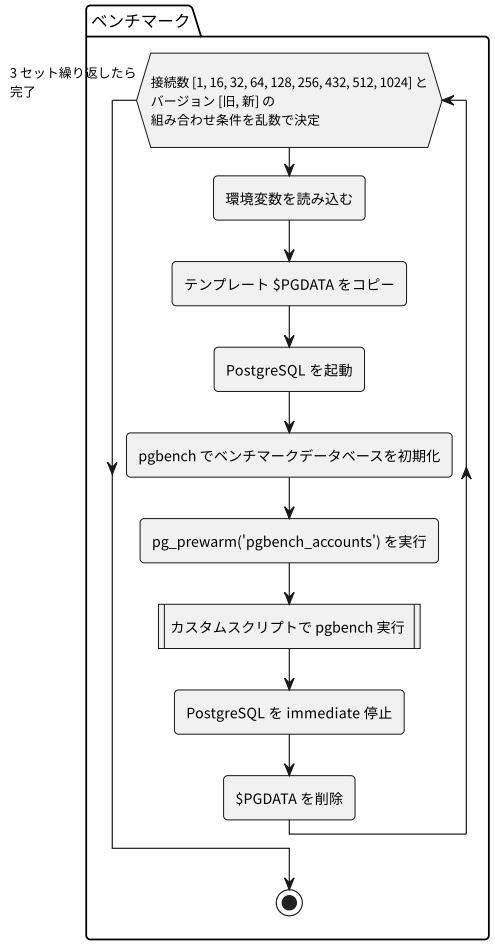
参照系カスタムスクリプトの内容

```
\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;
```

更新系手続き

セットアップ後にベンチマークを実施





- 試行の都度$PGDATAを作り直す
- PostgreSQLバージョン(2パターン)とクライアント同時接続数(9パターン)がTPSに及ぼす影響を計測
  - 1, 16, 32, 64, 128, 256, 432, 512, 1024
- 2×9=18パターンのベンチマークをランダムな順序で3回繰り返す
- 得られたTPSの中央値を結果として採用

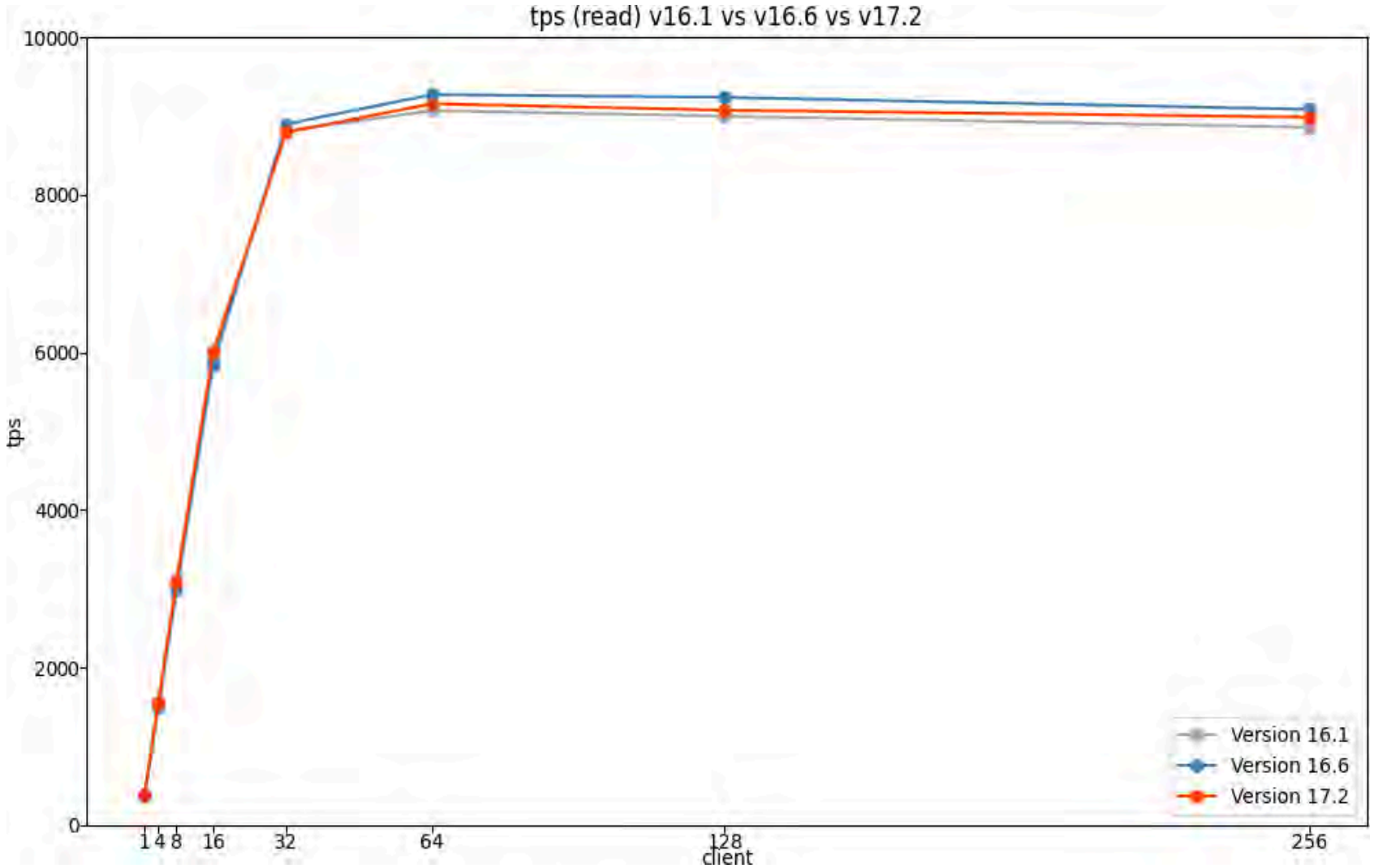
更新系カスタムスクリプトの内容

```
\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;
```

検証結果

info
参考情報として前年度の16.1の結果も掲載。

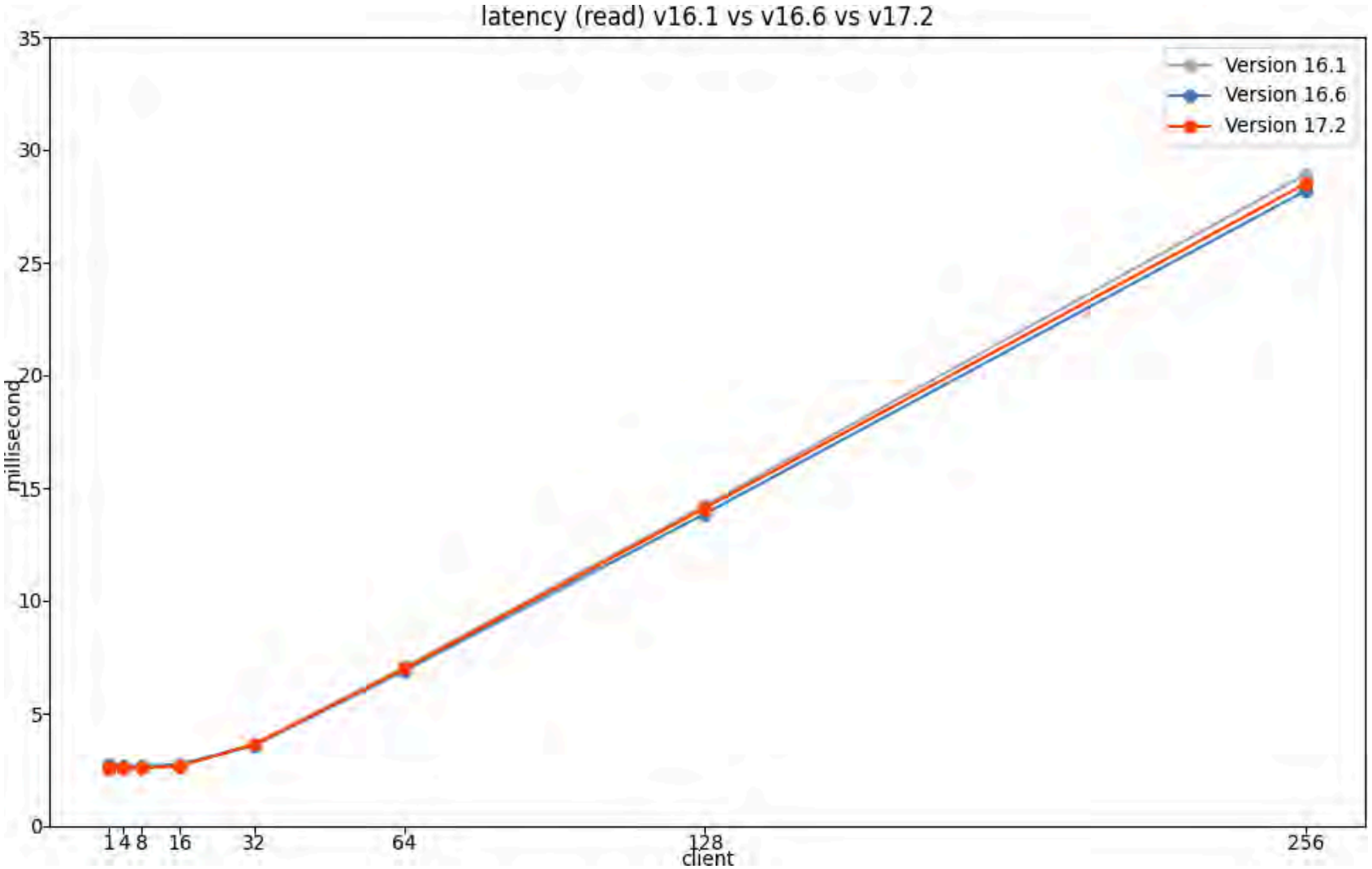
参照系TPS



- 64接続でTPSは頭打ち
- 低負荷(1～16接続)時に約3%TPSが向上
- 高負荷(32接続以上)時は約2%TPSが低下
- 全体平均で1%のTPS向上

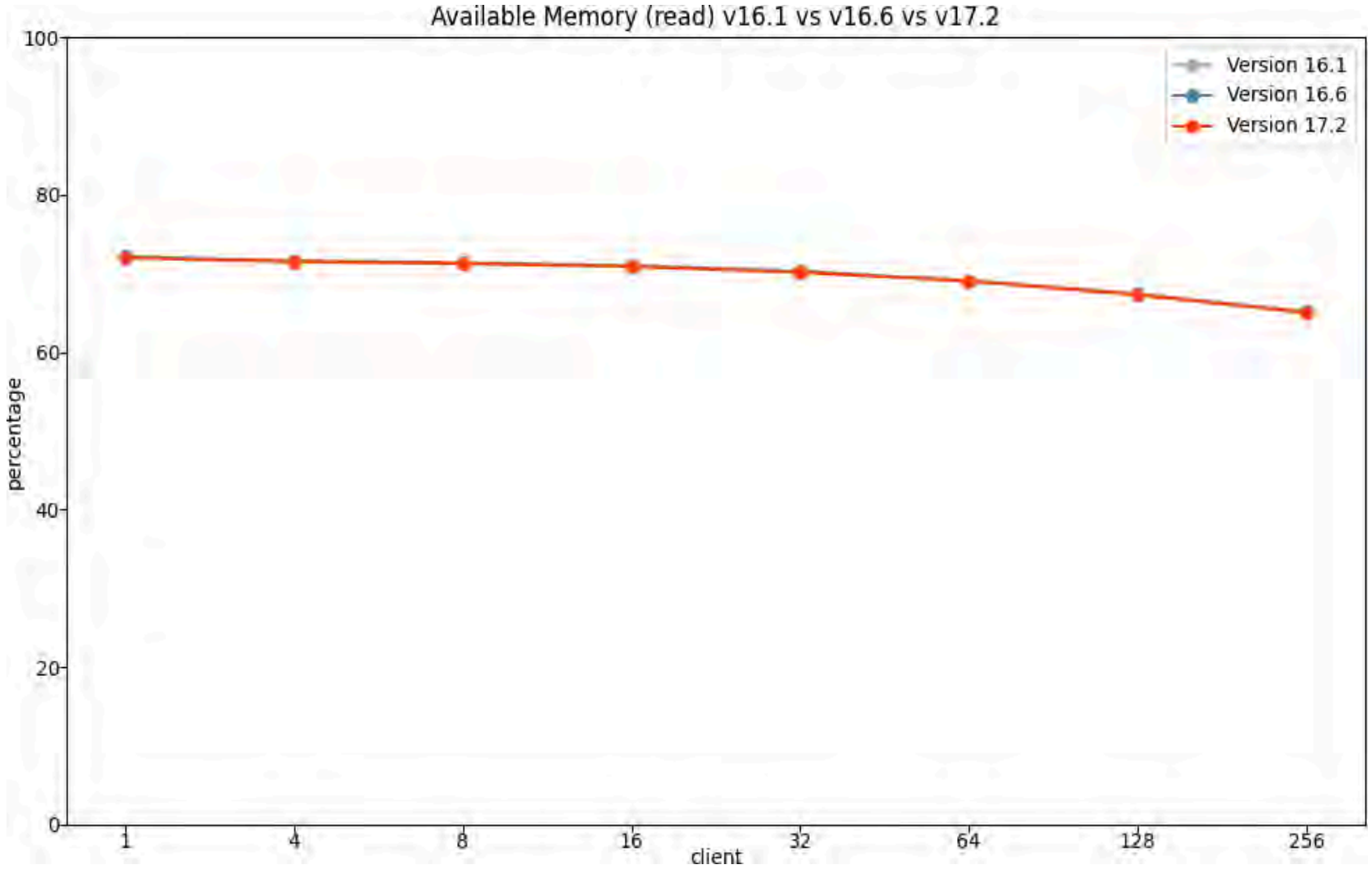
クライアント数	前バージョン比
1	1.04
4	1.03
8	1.03
16	1.02
32	0.98
64	0.98
128	0.98
256	0.98

参照系レイテンシ



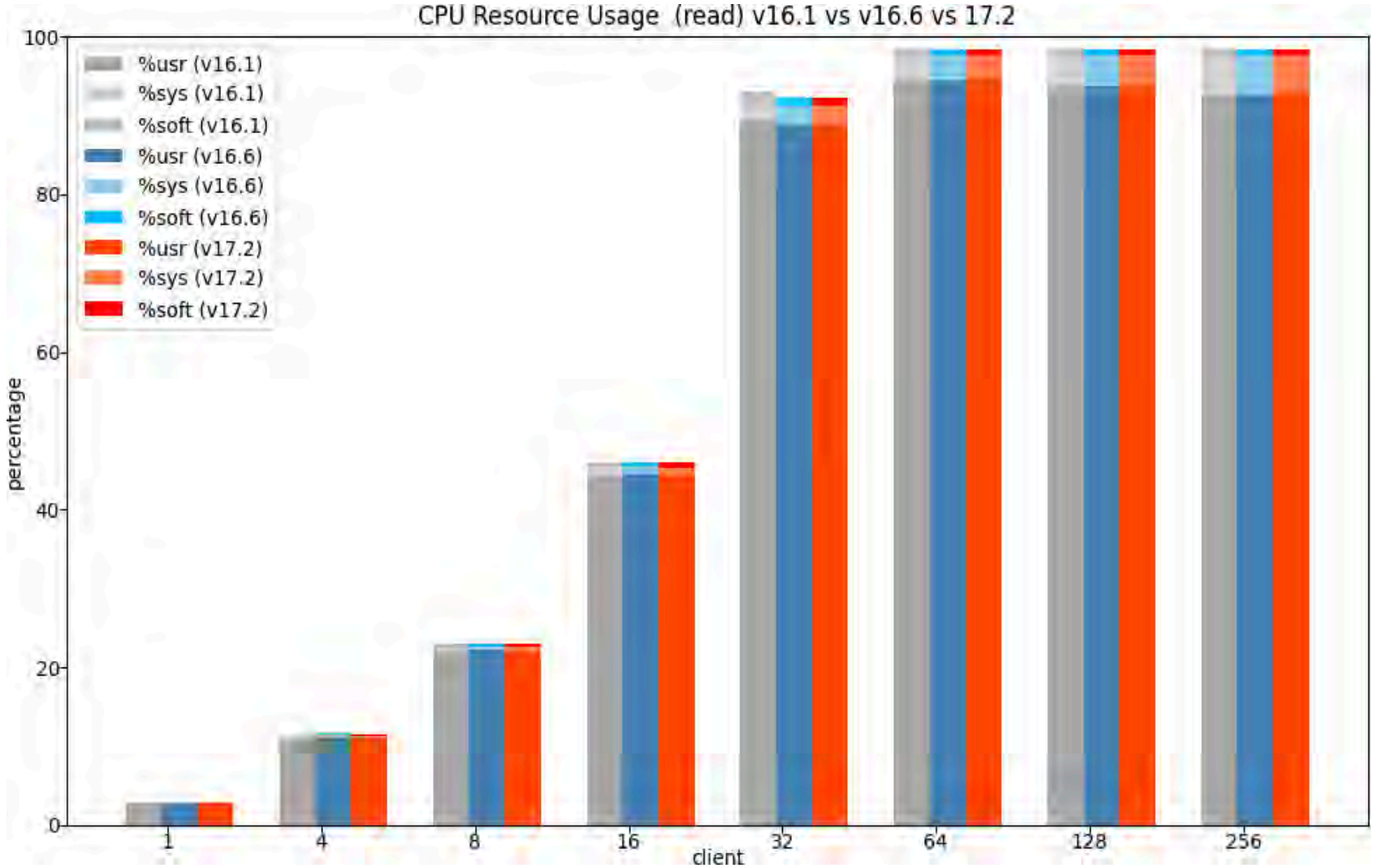
- TPSの結果を反映した形
- 全体平均で2%のレイテンシ短縮

参照系メモリ



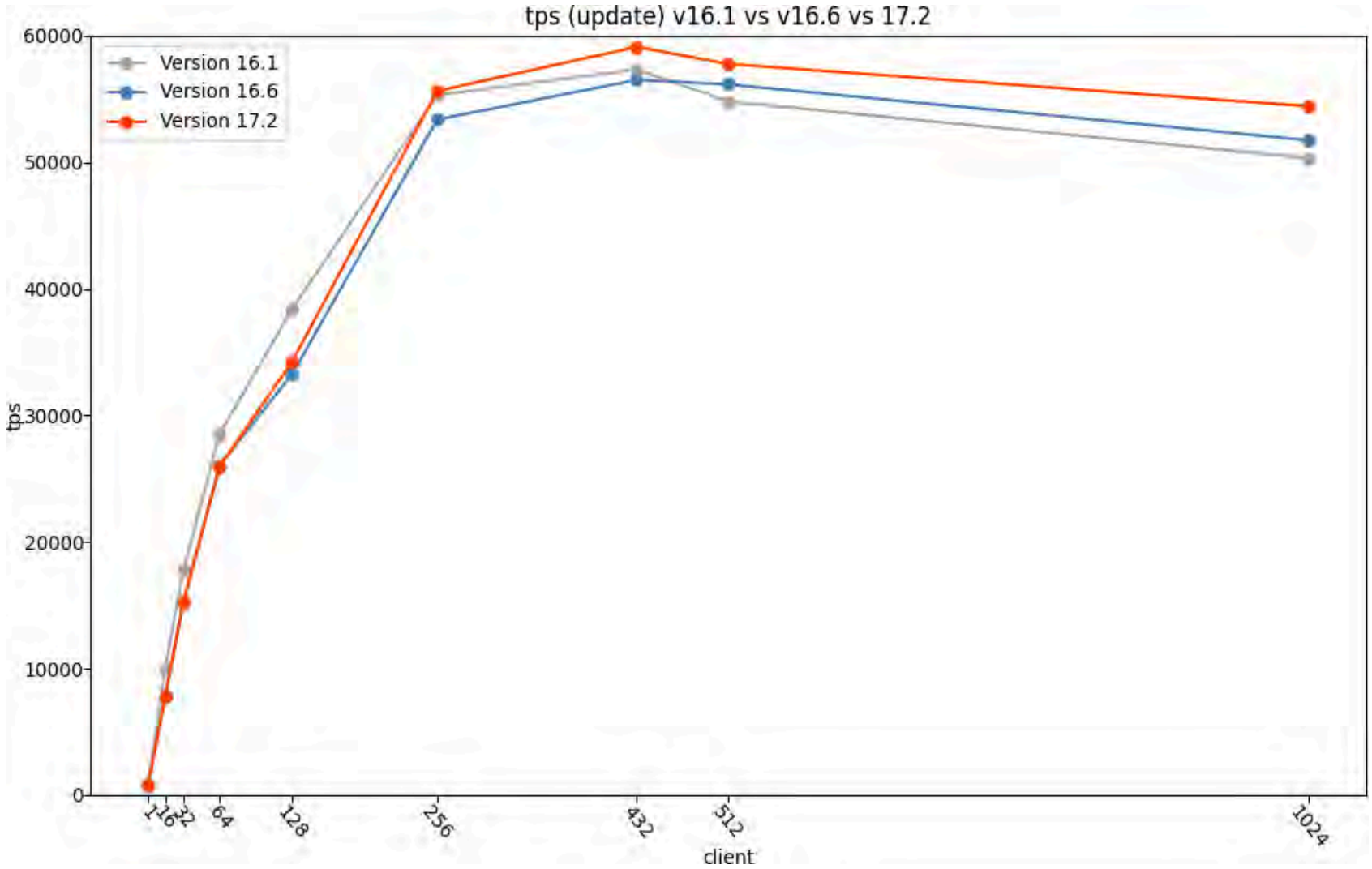
- バージョン間で差はない

参照系CPU



- 両バージョンともCPU使用率は32接続以上で上昇し、64接続でほぼCPU限界に達している
- %userが大部分を占める
- CPU使用率の傾向に両バージョン間に差はない

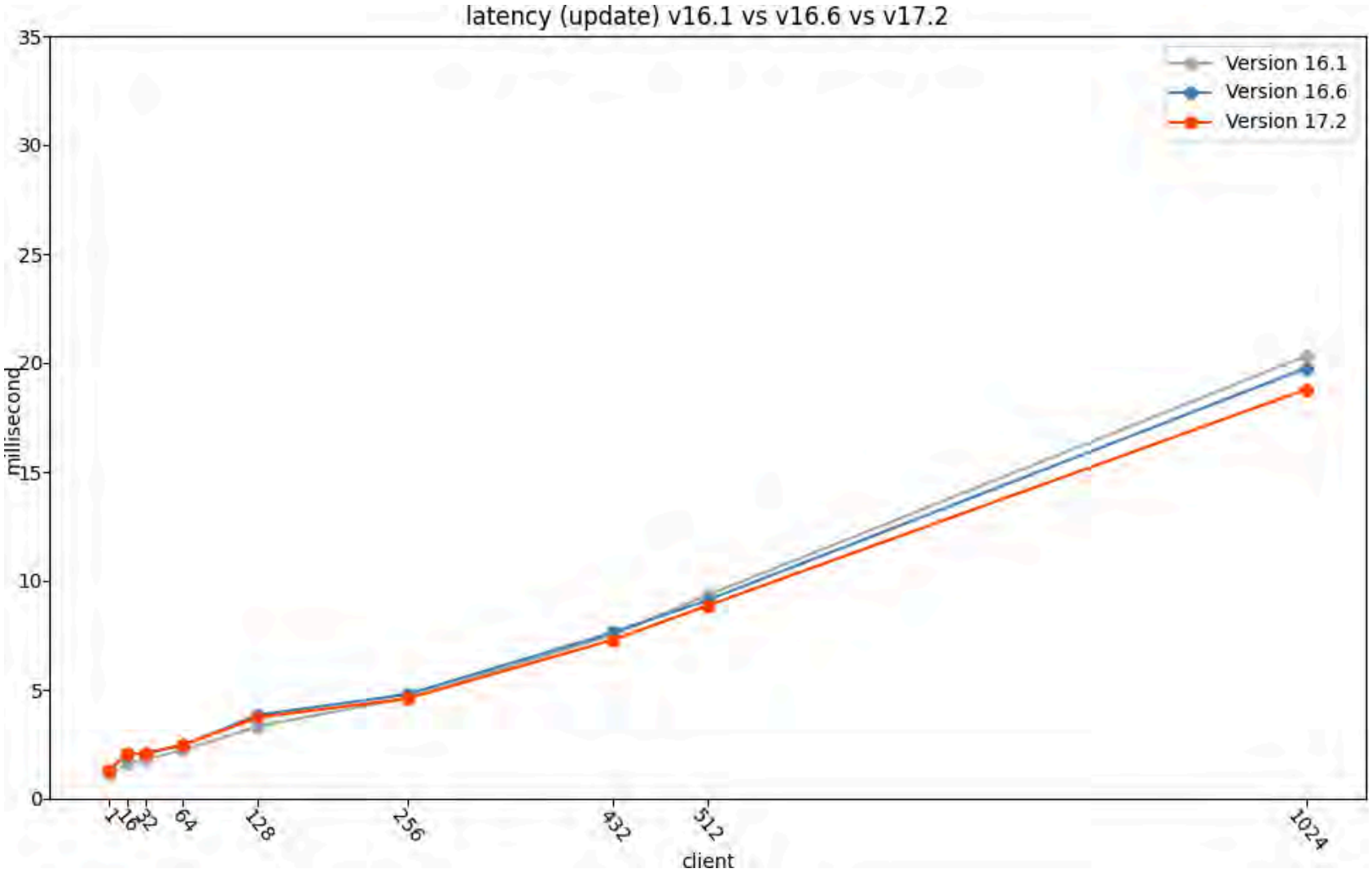
更新系TPS



- 432、512接続で頭打ち
- 高負荷(256～1024接続)時に最大約5%TPSが向上した
- 低～中負荷(1～128接続)時は平均すると前バージョン比は0%
- 全体平均で2%のTPS向上

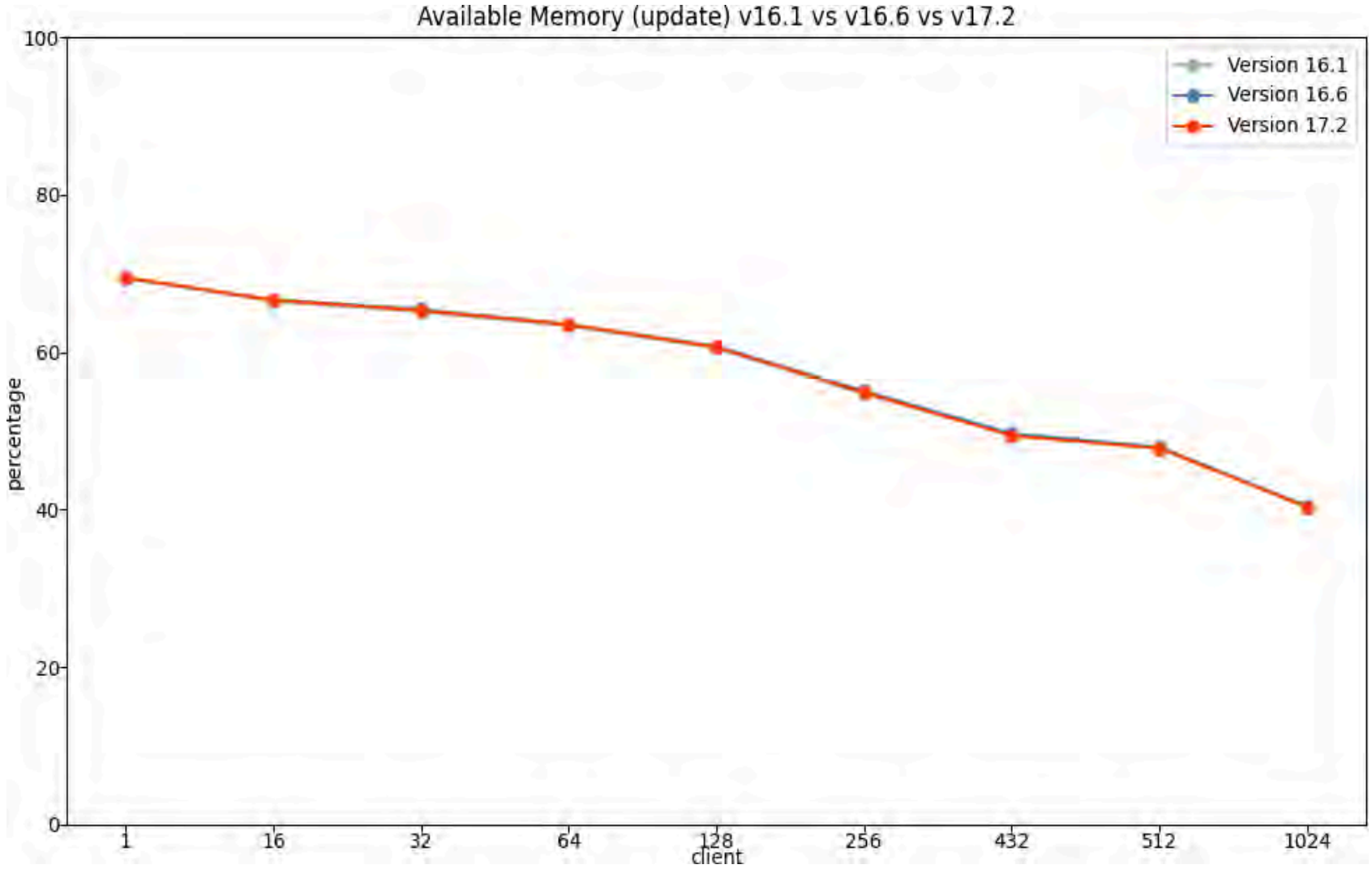
クライアント数	前バージョン比
1	1.00
16	0.99
32	1.00
64	0.99
128	1.02
256	1.04
432	1.04
512	1.02
1024	1.05

更新系レイテンシ



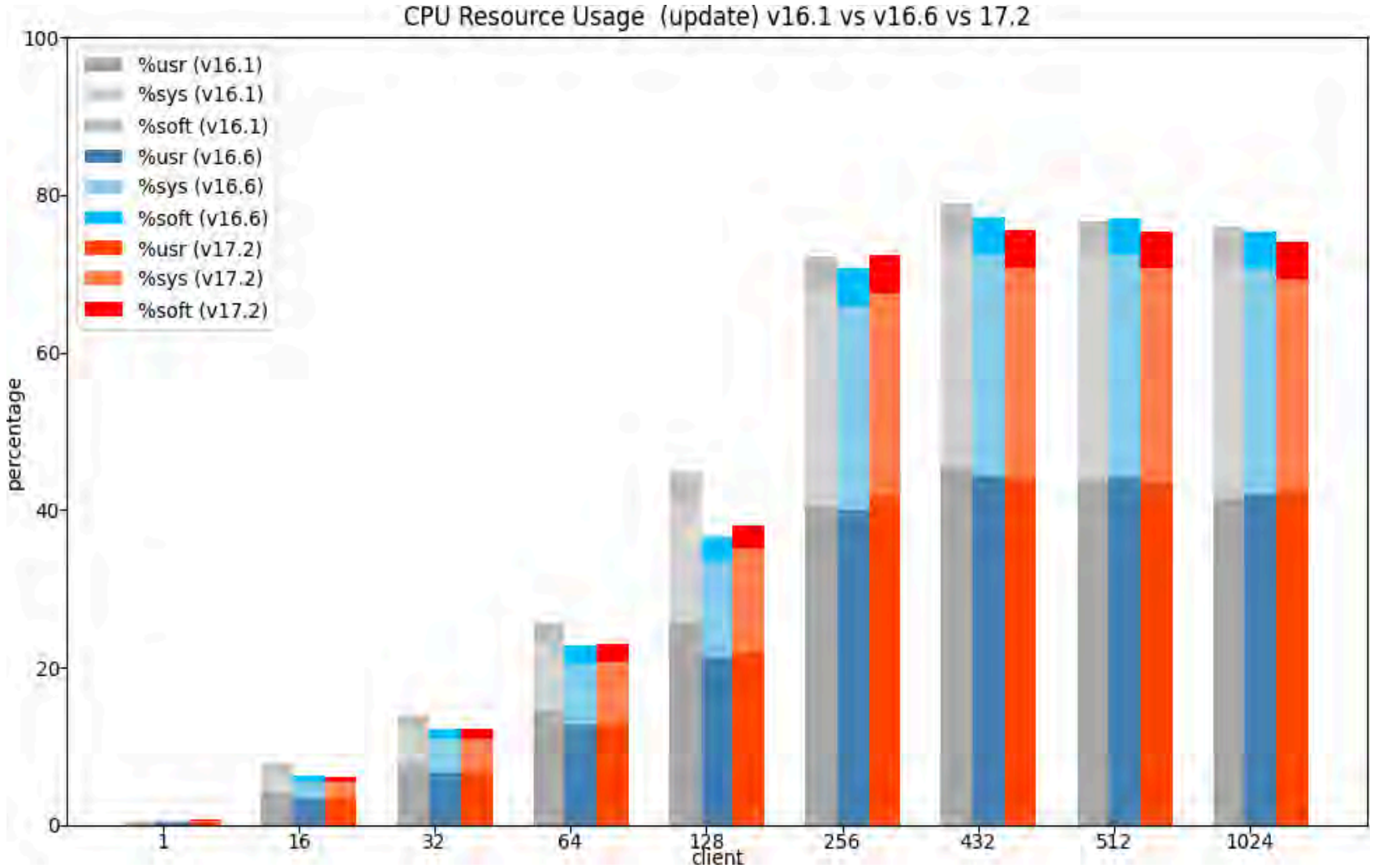
- TPSの結果を反映した形
- 全体平均で3%のレイテンシ短縮

更新系メモリ



- バージョン間で差はない

更新系CPU



- 17.2は16.6に比べて高負荷(432接続以上)時に2%CPU利用率が低くなっている
- ただし平均するとCPU利用率の差は0%

統計解析

参照系試験と更新系試験のTPSのどこに差があるか統計解析

取得したTPSのどこに差があるかを客観的に評価するためにR version 4.4.3を使って統計解析した。

等分散性検定

ルビーンの等分散性検定の結果、参照系・更新系共に分散の均一性に有意な差が認められた。

参照系

```
Levene's Test for Homogeneity of Variance (center = mean)
      Df F value    Pr(>F)
group 15  14.185 4.252e-10 ***
      32
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

更新系

```
Levene's Test for Homogeneity of Variance (center = mean)
      Df F value    Pr(>F)
group 17  4.8242 3.594e-05 ***
      36
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

データが分散の均一性の仮定を満たしておらず、かつ、各条件のサンプル数がN=3と少ないので、整列ランク変換(ART)(*)を行ってから二元配置分散分析を実施する。

info

(*) 整列ランク変換(ART)はdevelブランチのARToolを用いた。

統計解析

整列ランク変換(ART)を行って混合モデルの二元配置分散分析(*)

両試験とも交互作用が有意であった。

参照系

```
Analysis of Variance of Aligned Rank Transformed Data

Table Type: Analysis of Deviance Table (Type III Wald F tests with Kenward-Roger df)
Model: Mixed Effects (lmer)
Response: art(tps)

                    F Df Df.res     Pr(>F)
1 version      36.629  1      2   0.026231   *
2 num         179.729  7     14 1.2914e-12 ***
3 version:num  14.316  7     14 2.1197e-05 ***
---
Signif. codes:   0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

更新系

```
Analysis of Variance of Aligned Rank Transformed Data

Table Type: Analysis of Deviance Table (Type III Wald F tests with Kenward-Roger df)
Model: Mixed Effects (lmer)
Response: art(tps)

                    F Df Df.res     Pr(>F)
1 version      91.643  1      2   0.010736   *
2 num         285.092  8     16 8.9839e-16 ***
3 version:num  26.500  8     16 7.9998e-08 ***
---
Signif. codes:   0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

PostgreSQLバージョンとクライアント接続数の両方の要因が組み合わさって相加/相乗/相殺的にTPSに影響を与えている。

info

(*) ARToolの下位検定は被験者内計画に対応していない(Issue #37)。

統計解析

下位検定

交互作用が有意であったため、全条件の多重比較を実施し、どこの条件で両要因がTPSに影響を与えたか確認する。

参照系

- クライアント接続数64, 128でバージョン間に有意な差が認められた

```
contrast            estimate   SE   df t.ratio p.value
 16.6,1 - 17.2,1        -3.00 1.01 16.0  -2.959  0.2833
 16.6,4 - 17.2,4        -3.00 1.01 16.0  -2.959  0.2833
 16.6,8 - 17.2,8        -3.00 1.01 16.0  -2.959  0.2833
 16.6,16 - 17.2,16      -3.00 1.01 16.0  -2.959  0.2833
 16.6,32 - 17.2,32       3.00 1.01 16.0   2.959  0.2833
 16.6,64 - 17.2,64       5.67 1.01 16.0   5.589  0.0027 *
 16.6,128 - 17.2,128     8.67 1.01 16.0   8.548  <.0001 ***
 16.6,256 - 17.2,256     2.00 1.01 16.0   1.973  0.8094

P value adjustment: tukey method for comparing a family of 16 estimates
```

更新系

- クライアント接続数256, 432, 512, 1024でバージョン間に有意な差が認められた

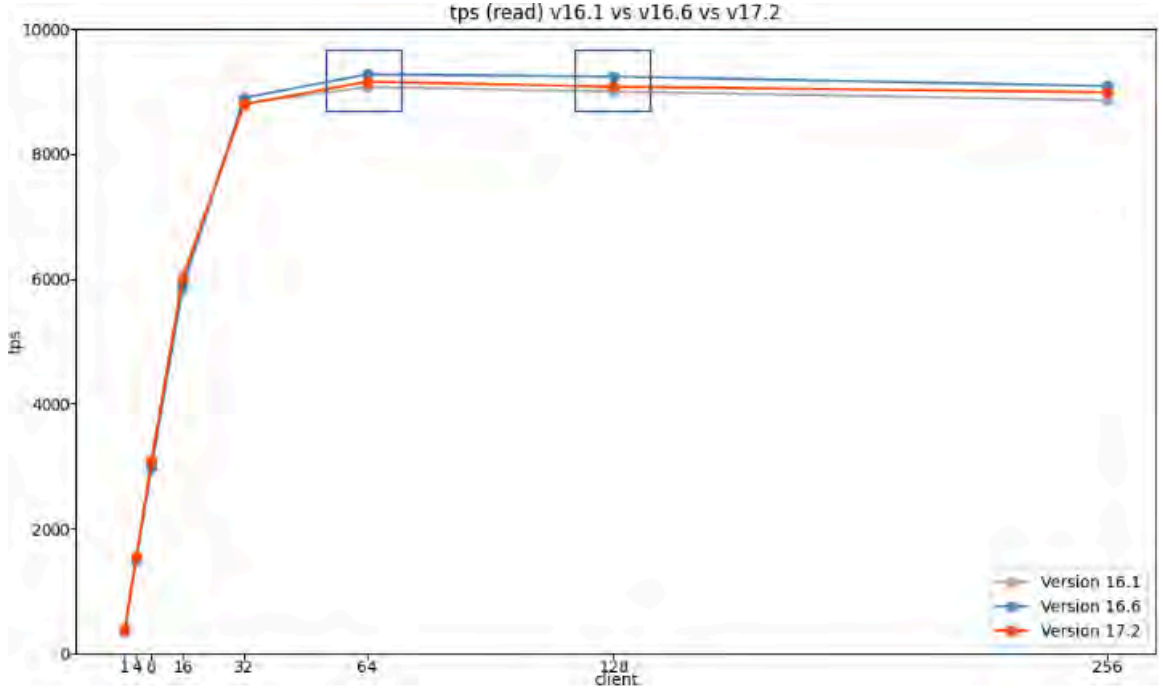
```
 contrast              estimate   SE   df t.ratio p.value
 16.6,1 - 17.2,1         -1.000 1.38 16.8  -0.725  1.0000
 16.6,16 - 17.2,16        0.333 1.38 16.8   0.242  1.0000
 16.6,32 - 17.2,32       -3.000 1.38 16.8  -2.175  0.7456
 16.6,64 - 17.2,64        0.333 1.38 16.8   0.242  1.0000
 16.6,128 - 17.2,128     -3.000 1.38 16.8  -2.175  0.7456
 16.6,256 - 17.2,256     -7.333 1.38 16.8  -5.317  0.0047 **
 16.6,432 - 17.2,432     -7.000 1.38 16.8  -5.075  0.0074 **
 16.6,512 - 17.2,512     -6.333 1.38 16.8  -4.592  0.0186 *
 16.6,1024 - 17.2,1024   -6.000 1.38 16.8  -4.350  0.0293 *

Degrees-of-freedom method: kenward-roger
P value adjustment: tukey method for comparing a family of 18 estimates
```

統計解析

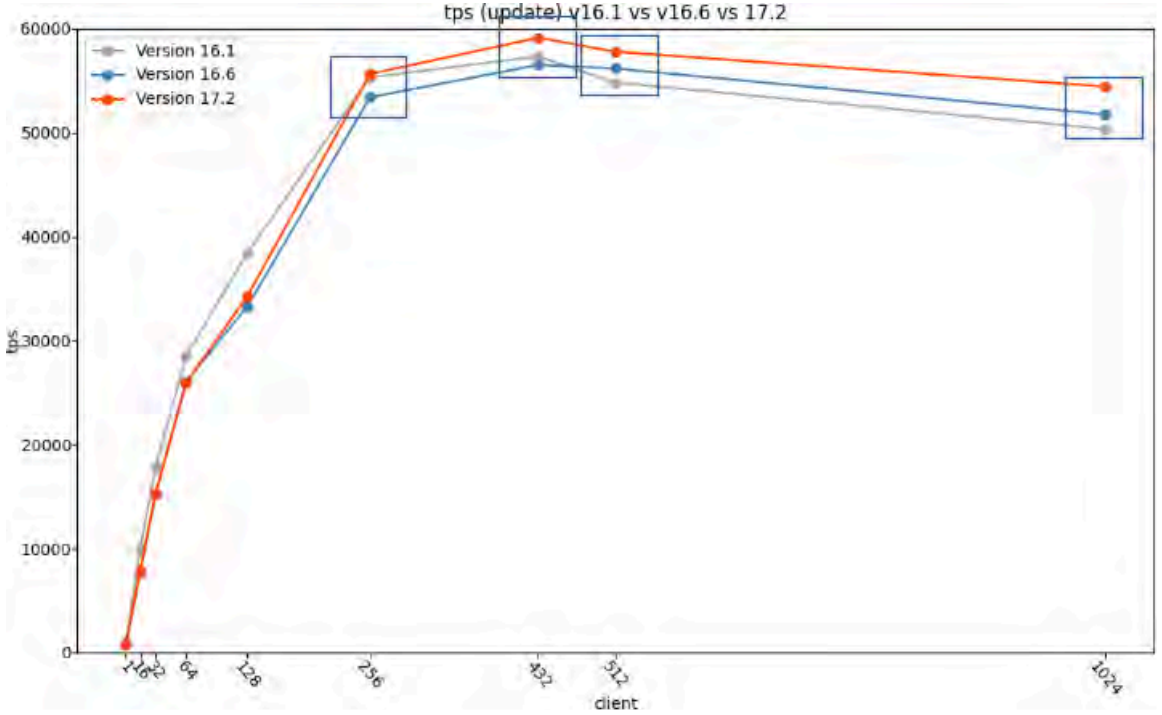
下位検定

枠で囲った箇所のみ、有意な差が認められた。

参照系



更新系



宿題事項の検討

更新系試験のCPU利用率のSoftIRQについて

- 2022年度から取り上げられた事象の宿題事項
- 更新系試験の全試行で、一貫して32個中8個のCPUにSoftIRQが発生している

info

SoftIRQ

ハードウェア割り込みが処理を完了できない場合や、即時に処理するとシステム全体のパフォーマンスが悪化する場合に行われるソフトウェア割り込みのこと。**ストレージI/Oの完了処理**(BLOCK_SOFTIRQ)や**ネットワークパケットの送受信**(NET_TX_SOFTIRQ, NET_RX_SOFTIRQ)などのイベントでSoftIRQが利用される。

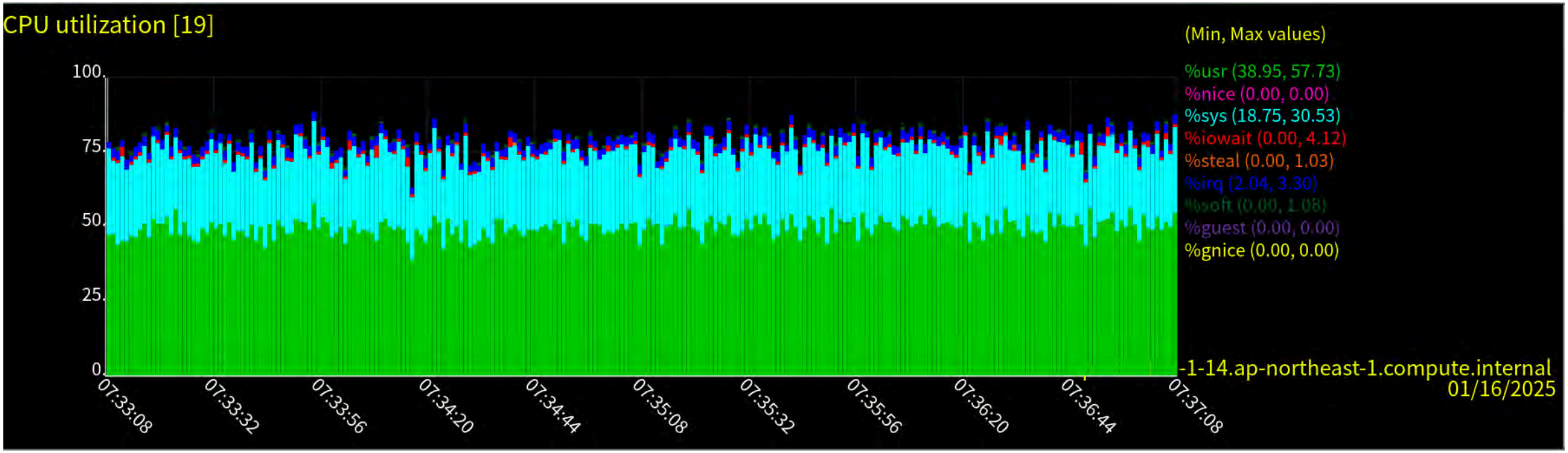
17.2の256接続の結果を代表に検討する。

- CPU毎のSoftIRQの発生傾向の確認
- SoftIRQを発生させるハードウェア割り込みの確認
- SoftIRQ発生原因の考察

更新系試験のCPU利用率のSoftIRQについて

- 32個中24個のCPUは以下のような傾向で、SoftIRQがほとんど発生していない

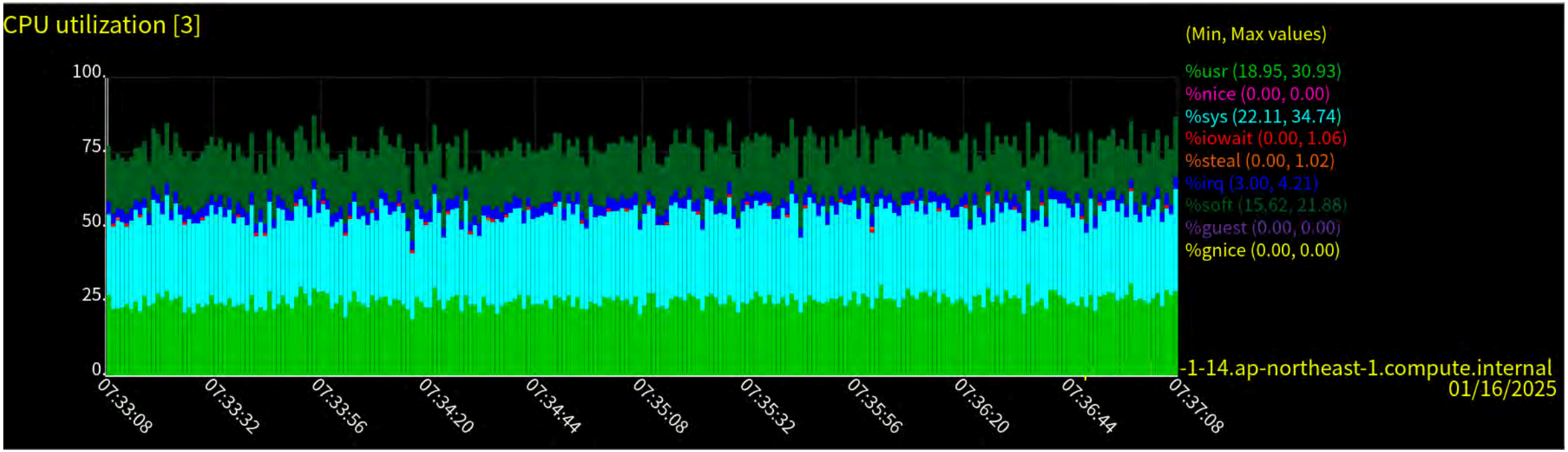
17.2, 256接続のCPU19



更新系試験のCPU利用率のSoftIRQについて

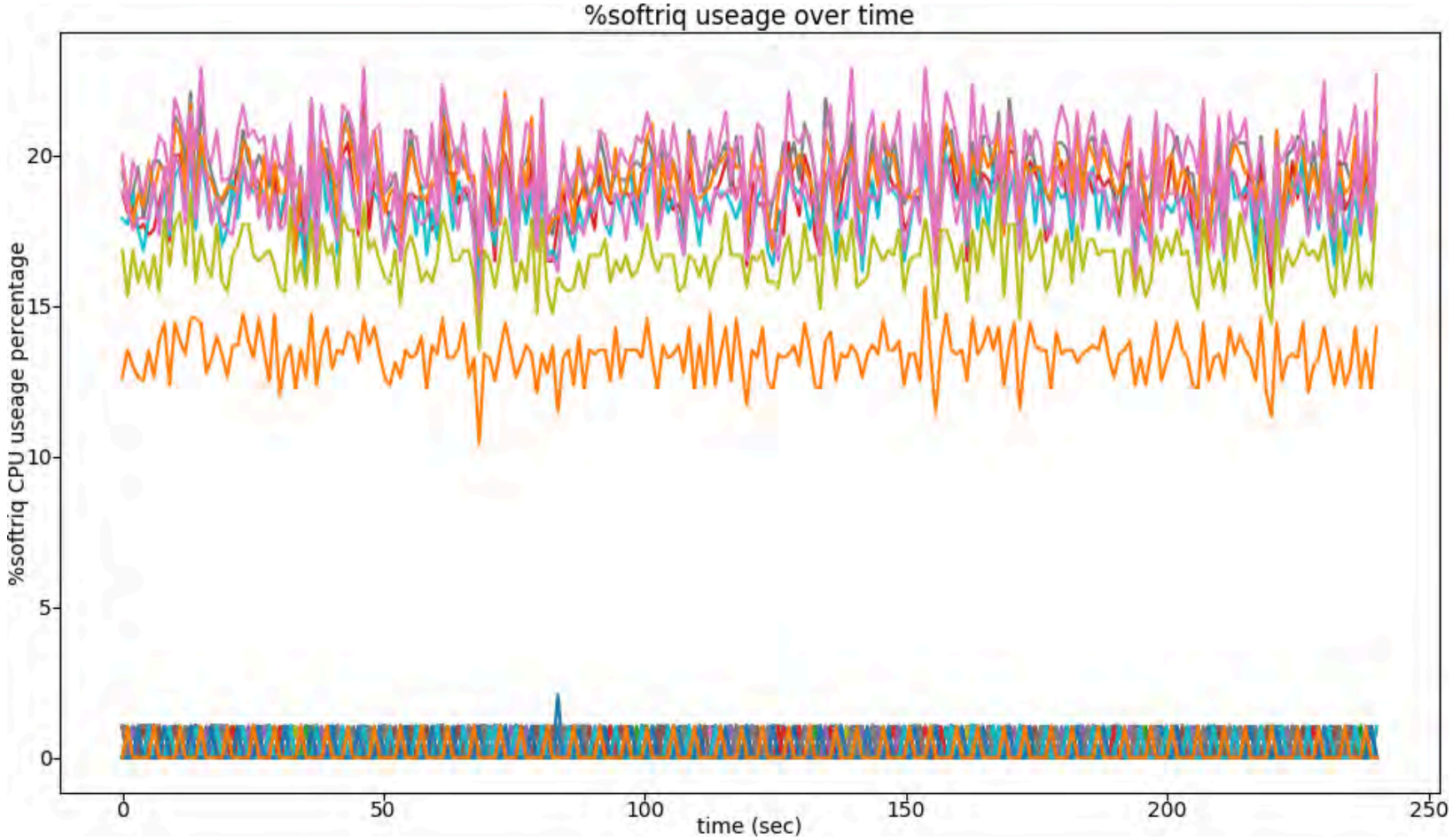
- 32個中8個のCPUは以下のような傾向で、SoftIRQが発生している

17.2, 256接続のCPU3



更新系試験のCPU利用率のSoftIRQについて

17.2の256接続試行中の%softだけをプロット。



- (凡例は省略しているが)CPU3, 7, 9, 11, 16, 21, 26, 28でのみ継続的にSoftIRQが発生している

更新系試験のCPU利用率のSoftIRQについて

SoftIRQがCPU3, 7, 9, 11, 16, 21, 26, 28で発生している理由

- m5a.8xlargeインスタンスはハイパースレッドで16CPUコアを32論理コアとしている
- 各スレッドにCPU0, CPU1, CPU2, ... のように番号が順番に振られる(*)

CPUコア	スレッド1	スレッド2	▷▷▷	CPUコア	スレッド1	スレッド2
コア0	CPU0	**CPU16**		コア8	CPU8	CPU24
コア1	CPU1	CPU17		**コア9**	**CPU9**	CPU25
コア2	CPU2	CPU18		コア10	CPU10	**CPU26**
コア3	**CPU3**	CPU19		**コア11**	**CPU11**	CPU27
コア4	CPU4	CPU20		**コア12**	CPU12	**CPU28**
コア5	CPU5	**CPU21**		コア13	CPU13	CPU29
コア6	CPU6	CPU22		コア14	CPU14	CPU30
コア7	**CPU7**	CPU23		コア15	CPU15	CPU31

各CPUコアあたり1スレッドだけを使ってSoftIRQを処理するよう割り当てられている。

割り当てのポリシーはirqbalanceサービスに依存する。

info

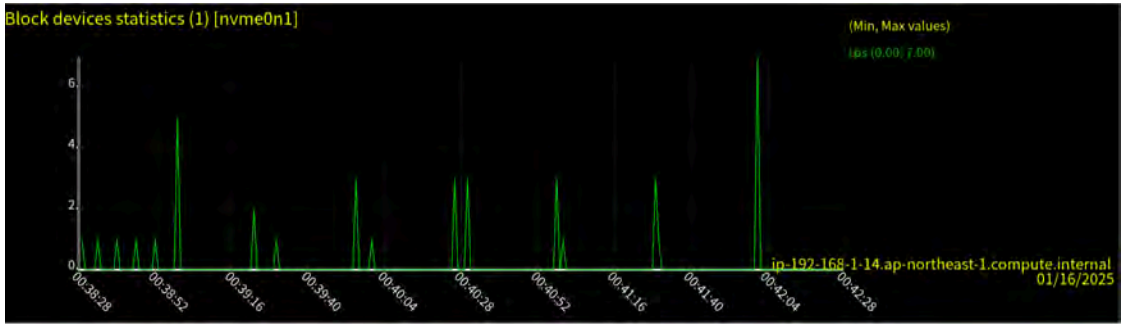
(*)Red Hat Enterprise Linux の同時マルチスレッド - Red Hat Customer Portal

更新系試験のCPU利用率のSoftIRQについて

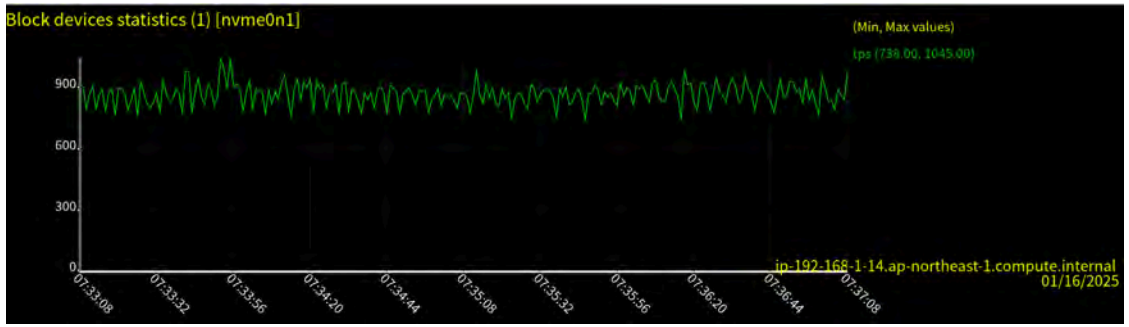
更新系試験で発生するハードウェア割り込み

参照系の17.2, 256接続と更新系の17.2, 256接続のストレージやネットワークの活動状況を比較すると顕著に更新系の方が高い。
sarの結果から、間接的にストレージI/Oの完了処理やネットワークパケットの送受信が活発であったと考えられる。

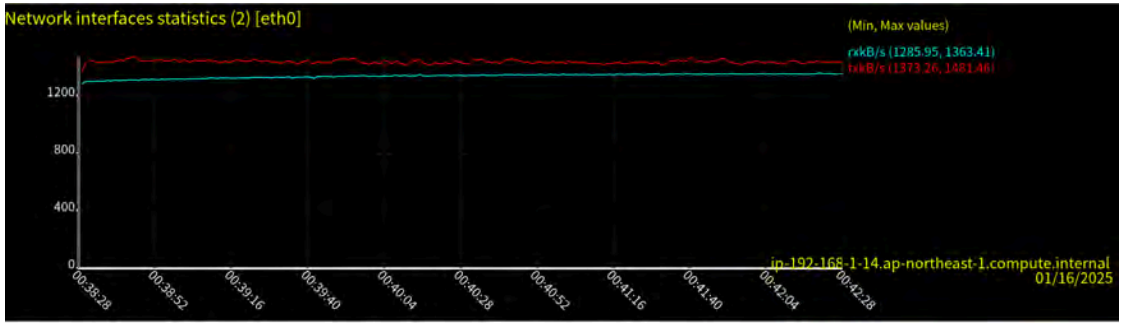
参照系のストレージsar



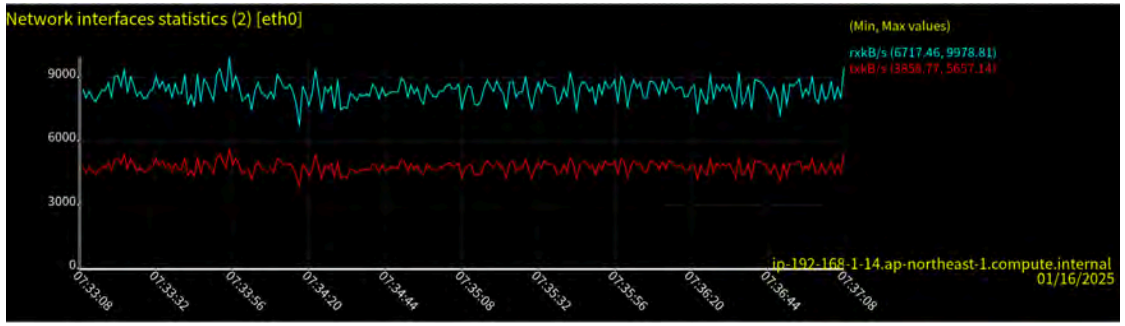
更新系のストレージsar



参照系のネットワークsar



更新系のネットワークsar



更新系試験のCPU利用率のSoftIRQについて

更新系試験でSoftIRQが発生する原因

更新系試験のワークロードはストレージI/Oの完了処理とネットワークパケットの送受信を多数発生させている。

- **ストレージI/Oの完了処理**
  更新系試験なのでストレージの書き込み要求の発行頻度が非常に高くなる。
- **ネットワークパケットの送受信**
  本検証ではEBSをアタッチしていた。
  EBSボリュームはネットワーク越しにアタッチされるのでネットワーク越しのストレージ書き込みであった。

大量の小さな書き込みI/Oがストレージとネットワークのハードウェア割り込みを発生させ、SoftIRQを引き起こしたと推察される。

- SoftIRQ処理は各CPUスレッド毎のksoftirqdが担当し、ksoftirqdはSoftIRQ処理を他へ移すことはない
- そのため、SoftIRQ処理は特定CPUに張り付きやすい
- システム全体のスループットと応答性の最適化を目的としたirqbalanceの動作上、許容・想定されたものである

まとめ

PostgreSQL 16.6 vs 17.2の試験結果

参照性能

- 低負荷(1～16接続)時のTPSが前バージョンと比較してわずかに改善されていたが、統計的な有意差は無し
- 高負荷(64, 128接続)時は前バージョンと比較してTPSが2%の性能低下が発生していた
  - リリースノートから本現象につながる変更は発見できていないが、何らかのソースコード上の変更がわずかながら影響を与えた可能性は考えられる

更新性能

- 高負荷(256接続以上)時に前バージョンと比較してTPSが最大5%(平均3.7%)改善していた
  - PostgreSQL 17ではWAL排他処理の改善(*)が行われており、高負荷な更新処理が集中する状況下で、WAL書き込みによるロック競合が減少し、スループットが向上したと推察される

宿題事項

- 更新系試験の特定CPUへのSoftIRQ集中はirqbalanceの動作とワークロード特性によるものであり、異常ではなくLinux・AWSの設計上起こり得る挙動

info

(*)PostgreSQL 17 に関する技術情報

PGECons

PostgreSQL Enterprise Consortium