OSS-DB

OSS-DB Gold
技術解説セミナー
2024/10/30 開催

DBパフォーマンスチューニング

本日の講師

株式会社 アシスト
小笠原 宏幸

LPI-JAPAN

講師紹介

■小笠原宏幸

- 株式会社アシスト
  ビジネスインフラ技術本部データベース技術統括部 所属

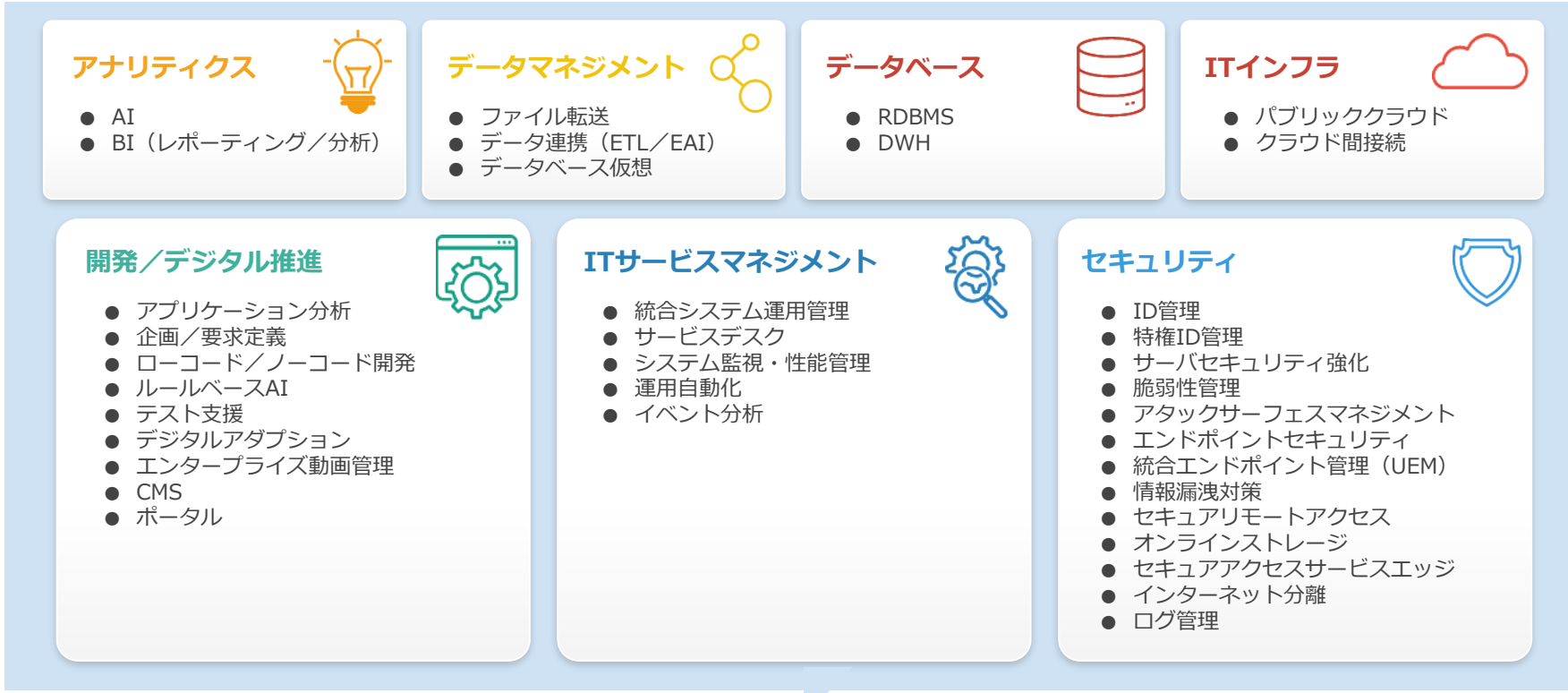
- PostgreSQL / EDB / Oracle DB などの DB 関連が主な担当業務
  - フィールドエンジニア
  - プリセールス
  - 教育講師

- 近年は、大手お客様に対して、PostgreSQL DBサーバ構築だけでなく、
  異RDBMSからPostgreSQLへの移行に関わる支援も実施。

アシスト紹介

■株式会社アシスト（https://www.ashisuto.co.jp/）

アシストは お客様に「ITソリューション」と「ソフトウェアサポート」
を提供する「パッケージ・インテグレーター」です。



アジェンダ

1. OSS-DB Gold 試験概要
2. 効果的な学習方法
3. DBパフォーマンスチューニング解説

OSS-DB／オープンソースデータベース技術者認定試験

■OSS-DBとは

オープンソースのデータベースソフトウェア「PostgreSQL」を扱うことができる技術力の認定です。
様々な分野でPostgreSQLの利用拡大が進む中でOSS-DBの認定を持つことは、自分のキャリアのアピールにもつながります。

✓OSS-DB Silverは導入や運用ができる技術力の証明

PostgreSQLについての基本的な知識を持ち、データベースの運用管理が行えるエンジニアとしての証明ができます

✓OSS-DB Goldは設計やコンサルティングができる技術力の証明

PostgreSQLについての深い知識を持ち、データベースの設計や開発のほか、
パフォーマンスチューニングやトラブルシューティングまで行えることが証明できます

✓対象のバージョンはPostgreSQL 12～14

「OSS-DB技術者認定」を取得するメリット

信頼されるエンジニアはどっち？

DB技術には「どんな要素があるのか」「それらがどのように関連しているのか」を体系的に押さえる

［例］
SQL、テーブル設計(論理/物理)、パラメータ設定、性能、バックアップ、監視、トランザクション、セキュリティ、バッファキャッシュ、…

DBのドキュメントや
書籍を読む

認定取得を通じて
体系的な理解度を確認する

OSS-DB Gold 試験概要

OSS-DB Goldは、2023年2月にV3.0がリリースされました。
Gold資格認定は、Silver資格を保有していることが前提条件になります。

資格認定条件	OSS-DB Silverを保有していること
出題数と時間	30問 / 90分 (合格目安は 70点以上)
試験実施方式	選択形式
試験実施形態	受験会場 / オンライン
試験バージョン	V3.0（PostgreSQL12～14が対象）
有効期間	認定日から5年間

出題範囲概要

出題分野	比重		重要度
G1 運用管理	30%	データベースサーバ構築	2
		運用管理用コマンド全般	4
		データベースの構造	2
		レプリケーション運用	1
G2 性能監視	30%	アクセス統計情報	3
		テーブル / カラム統計情報	2
		クエリ実行計画	3
		その他の性能監視	1
G3 パフォーマンスチューニング	20%	性能に関係するパラメータ	4
		チューニングの実施	2
G4 障害対応	20%	起こりうる障害のパターン	3
		破損クラスタ復旧	2
		レプリケーションの障害と復旧	1

詳細はLPI-Japan様のサイトにてご確認下さい
https://oss-db.jp/outline/gold#questionnaire_range_gold

効果的な学習方法

- ドキュメントによる基礎スキルの習得
- 実機による確認
- 模擬問題による試験対策

ドキュメントによる基礎スキルの習得

以下ドキュメントなどを用いて、用語や基本アーキテクチャに関する理解を深めます。

■マニュアル

一番基礎となるドキュメントです。
https://www.postgresql.jp/document/

■書籍

内部構造や運用管理を詳しく説明した書籍が複数出版されています。
マニュアルと異なり、図解を多く含んだ書籍もあり、理解が深まります。

■Webサイト

各種団体から試験対策や業務に役立つ情報が提供されています。

- Let's POSTGRES（日本PostgreSQLユーザ会）
  運用管理などを整理した技術的な文献・動画で学習できます。
  https://lets.postgresql.jp/index.php/
- PGECons（PostgreSQLエンタープライズ・コンソーシアム）
  技術ワーキンググループの成果物がアップロードされています。
  https://pgecons-sec-tech.github.io/tech-report/

実機による確認

手を動かして実際に確認することで、PostgreSQLへの理解が深まります。

■PostgreSQLの導入

導入手順は日本PostgreSQLユーザ会のサイトや、各個人のサイト/ブログなどに多数掲載されています。

- 日本PostgreSQLユーザ会：インストールガイド
  https://lets.postgresql.jp/map/install

■実機確認の必要性

- ドキュメントベースで理解したことを、実際に体感することで理解が促進されます。
- SQLのコマンドや実行結果に関する問題が一定量出題されます。
  このような問題は単純に「覚える」だけでは対応が難しく、応用が利くように実機確認で
  理解を深めることが重要です。

問題例）

以下コマンドでバックアップする際の説明として、適切なものを2つ選びなさい。
select pg_start_backup(current_timestamp::text,'true');

A）pg_start_backup関数の第3引数が指定されていないため、非排他モードが開始されている。
B）バックアップモードが開始され、直後にチェックポイントが発行される。
C）同一セッション内でpg_stop_backup関数を発行しなければならない。
D）第1引数はどんな文字列を指定しても問題ない。
E）複数のバックアップを並列で実行して取得することができる。

模擬問題による試験対策

試験に慣れるため、模擬問題による対策をお勧めします。

■LPI-Japan 公式問題集

LPI-Japan様のサイトにて公式問題集が公開されています。4カテゴリ併せて約60問あります。
https://oss-db.jp/measures/sample

■アシスト 試験対策問題集

- アシストにてOSS-DB Gold Ver.3.0に対応した問題集をリリースしました。
- カテゴリ別練習問題と模擬試験問題を含めた全146問を収録しています。
- 以下2形態で販売しています。
  - Amazon Silver問題集 / Gold問題集

Amazon OSS-DB Silver　Amazon OSS-DB Gold

  - Udemy Silver問題集 / Gold問題集

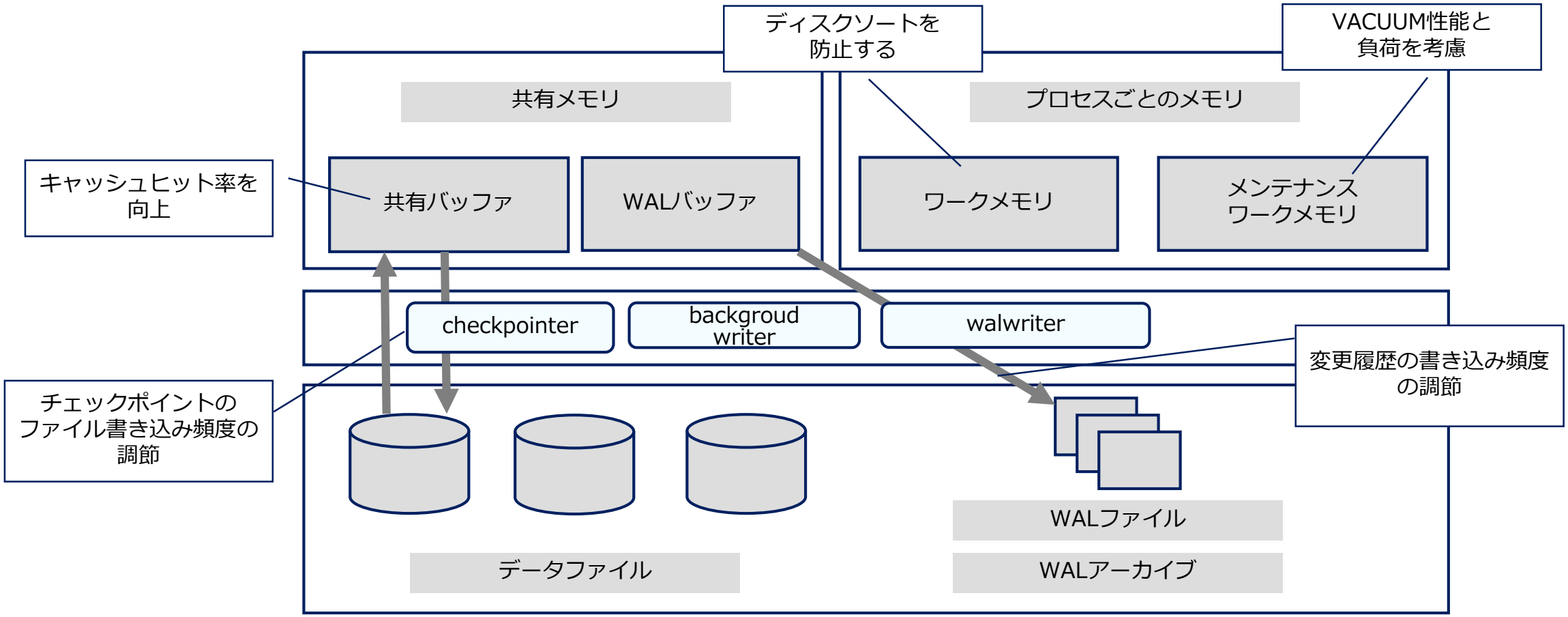
Udemy OSS-DB Silver　Udemy OSS-DB Gold

アシストメンバー 受験体験記

	メンバーA	メンバーB
学習期間	1.5か月（合計30時間）	2か月（合計40時間）
学習方法	1. マニュアル、書籍で基礎学習 2. LPI-Japan 公式問題集を解く 3. 問題集で間違えた箇所をマニュアルで確認し理解する ※2～3を繰り返し実施	1. アシスト試験対策問題集を解く 2. 間違えた箇所をマニュアルや実機操作で確認 3. LPI-Japan 公式問題集で最終確認
コメント	間違った問題を繰り返し確認するだけでなく、間違えた原因を理解するまで複数の文献を確認した。	アーキテクチャ部分は出来る限り図を描いて理解を深めるようにした。

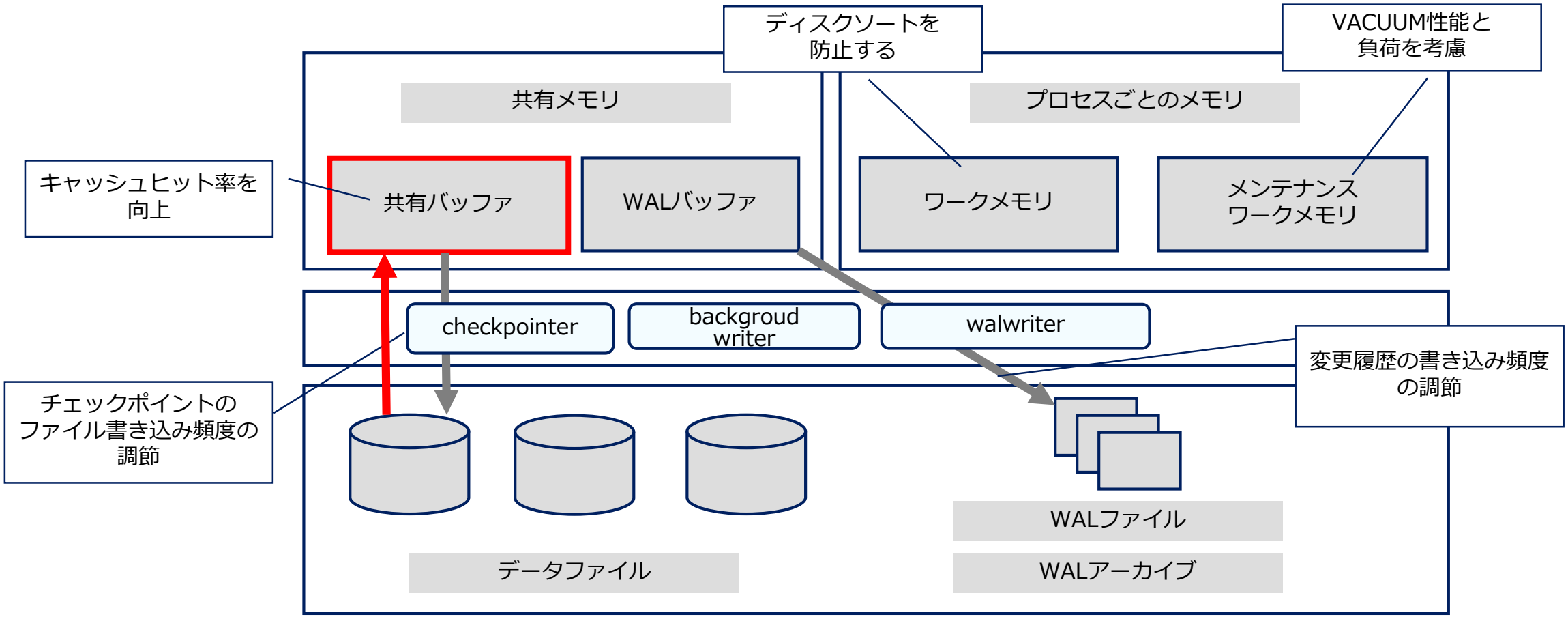
DBパフォーマンスチューニング解説

基本アーキテクチャと主なチューニングポイント



共有バッファとは

SQL処理のためにデータファイルから読込んだデータは共有バッファ上にキャッシュされます。
それを再利用することでディスクI/Oを削減できます。



共有バッファのパラメータ

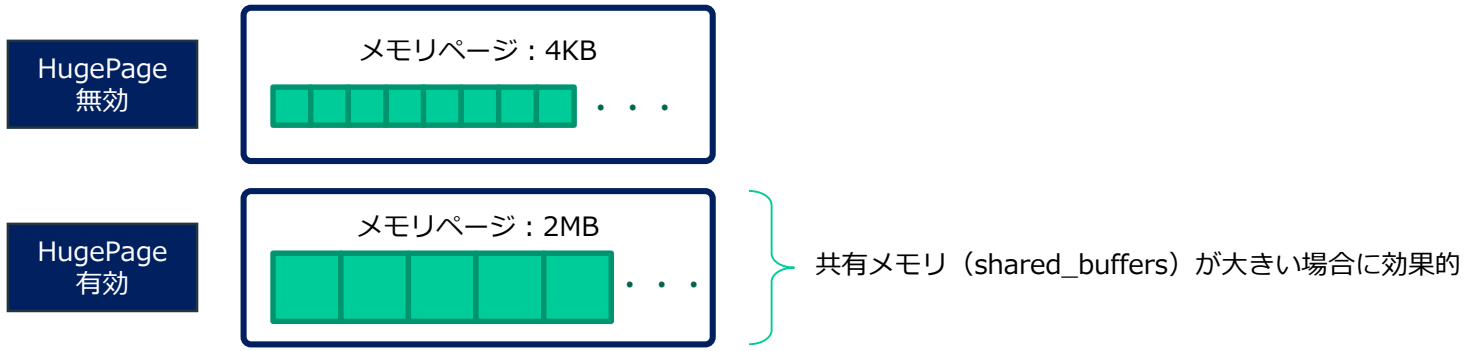
shared_buffersパラメータ　デフォルト：128MB

共有バッファのサイズを指定します。
大きく設定すると、キャッシュできるデータが増え、ディスクI/Oの削減に繋がります。
ヒット率が90%以上になるように調整します。

huge_pagesパラメータ　デフォルト：try | on | off

OSレベルでは、通常4KBのメモリページが使用されますが、より大きな2MBのメモリページを使用することで、
大規模な共有メモリ管理のオーバーヘッドを減らせる場合があります。これをHugePage（またはLargePage）といい、
Linuxであればカーネルパラメータvm.nr_hugepagesで有効化します。

huge_pagesパラメータは、PostgreSQLとしてHugePageを要求するかどうかを設定します。
デフォルトはtryであり、OSレベルでHugePageが有効化されていれば、HugePageを共有し、失敗した場合はデフォルトに戻します。
shared_buffersに数GB以上を割り当てるような場合に効果が期待できます。



共有バッファの分析

キャッシュヒット率を確認し、共有バッファが不足していないかを確認します。
pg_statio_user_tablesビューのheap_blks_hit列（ヒット数）とheap_blks_read列（ディスク読み込み数）をもとにヒット率を算出します。

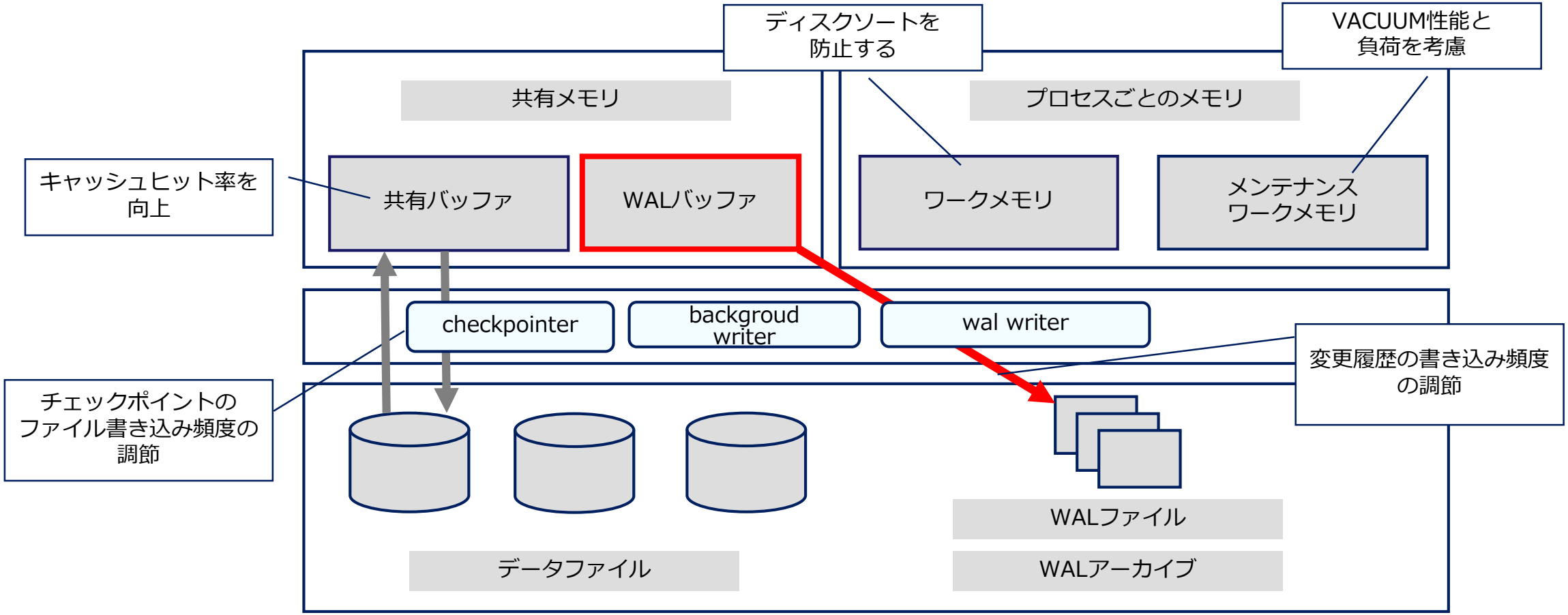
- pg_statio_user_tablesビューによるヒット率の確認

```
db1=# SELECT relname,
db1-# ROUND(heap_blks_hit*100/(heap_blks_hit+heap_blks_read),2) AS cache_hit_ratio
db1-# FROM pg_statio_user_tables WHERE heap_blks_read > 0 ORDER BY cache_hit_ratio;
      relname      | cache_hit_ratio
-------------------+-----------------
 pgbench_accounts |            9.00
 pgbench_branches |           76.00
        :                        :
```

キャッシュヒット率が **90**%を下回るようであれば、サーバの空きメモリ量に応じてshared_buffersの増加を検討します。

WALバッファとは

変更処理で発生した履歴を一時的に格納する領域です。
変更履歴は最終的にwal writerプロセスがWALファイルに書き込みます。

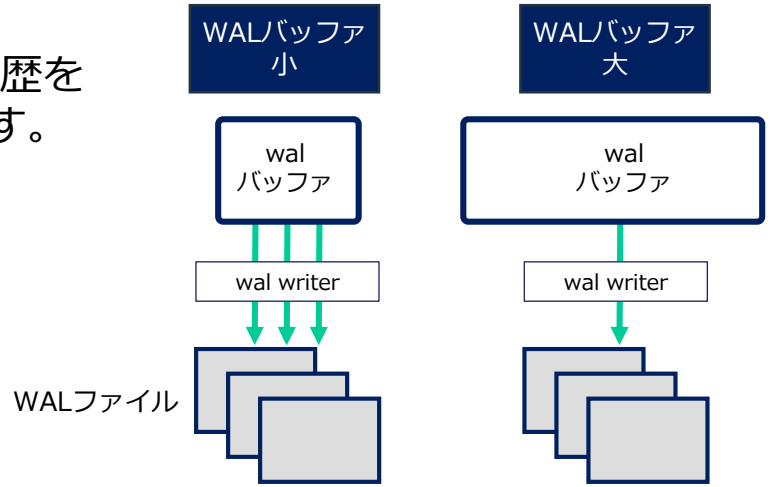


WALバッファのパラメータ

wal_buffersパラメータ

デフォルト：-1(shared_buffersの1/32)

WALバッファのサイズを指定します。
WALバッファが一杯になると、wal writerプロセスがWALファイルに変更履歴を書き込むため、過剰な書き込みが発生しないよう、適切なサイズに調整します。



wal_compressionパラメータ

デフォルト：off | on

WALの圧縮有無を指定します。デフォルトはoffです。
wal_compressionの有効化は、WALファイルへの書き込み減少に伴うディスクI/Oの低減や、領域の節約という観点で有効です。ただし、圧縮/解凍でCPUを余計に消費するため、開発環境などで影響を確認した上で実装します。
※PostgreSQLは他の有償RDBMSと比較してWAL生成量が多いため、ディスクI/Oが課題となっている環境では、本パラメータが有効である可能性があります。

WAL関連のパラメータで wal_keep_size がありますが、本パラメータはストリーミングレプリケーションで考慮するものです。
一度に大量のWALが生成されると、レプリケーション対象のWALが上書きされ消失する懸念がありますが、本パラメータでpg_walに保持するWALの最低サイズを設定できます。

WALバッファの分析

■WALファイルへの書き込み

以下のタイミングでwal writerがWALバッファからWALファイルに書き込みます。

- トランザクションがコミットされたとき
- WAL バッファが一杯になったとき
- wal_writer_delay パラメータに設定された時間が経過したとき（デフォルト：200ms）

■WALバッファの考慮事項

WALバッファのサイズが小さすぎると、コミット以外のタイミングでも書き込みが発生します。
その結果、ディスクI/Oが増加しパフォーマンスが低下する恐れがあります。

OSコマンド等の確認により、WALファイルが格納されるディスクに対するI/Oが多い場合、
WALバッファのサイズ等を調整し、ディスクI/Oの負荷を減らすようにします。

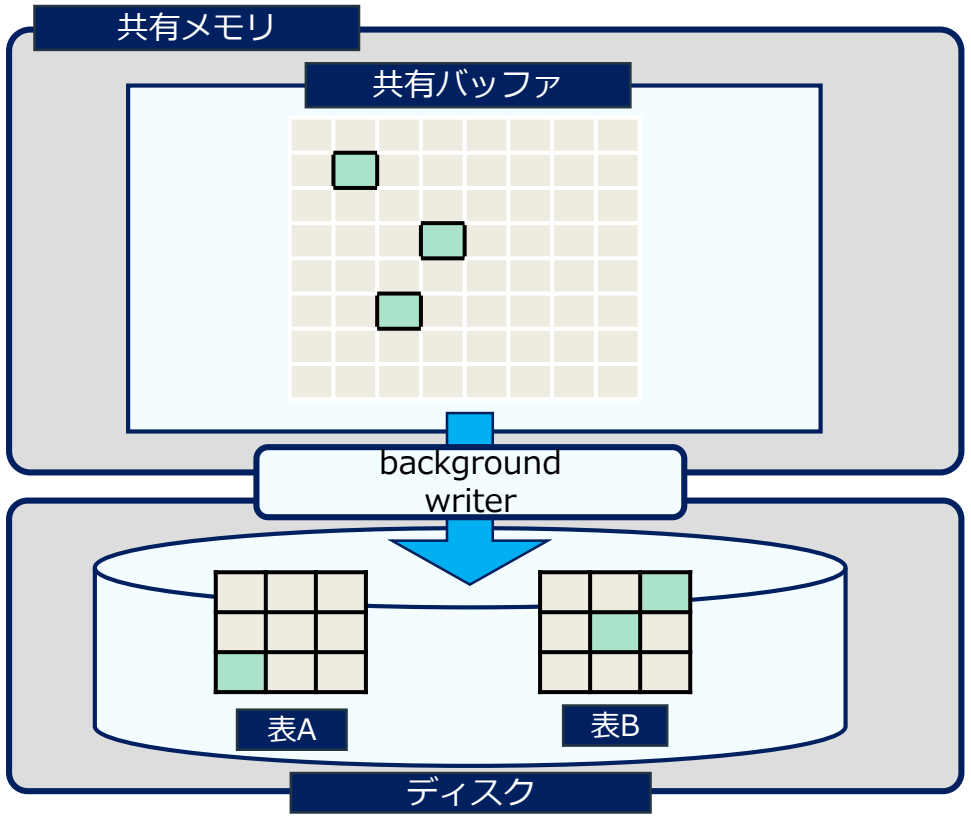
チェックポイントとは

チェックポイントとは、共有バッファ上の変更済みデータ（ダーティバッファ）をデータファイルに書き出し、
メモリとディスクの同期を取る処理のことです。

チェックポイントの間隔は、インスタンス障害からの回復時間とパフォーマンスに影響を与えます。
チェックポイントが頻発すると、ディスクI/O が多発しパフォーマンスが低下する恐れがあります。
そのため、パフォーマンスの観点ではチェックポイントが多発しないよう、チェックポイントの発生状況を確認し、
関連するパラメータでチェックポイントの発生間隔を調整します。

チェックポイントの発生間隔は
以下のパラメータで設定できます。

- 時間契機　　：　checkpoint_timeout
- サイズ契機　：　min_wal_size
  max_wal_size



チェックポイントのパラメータ①

デフォルト：5min

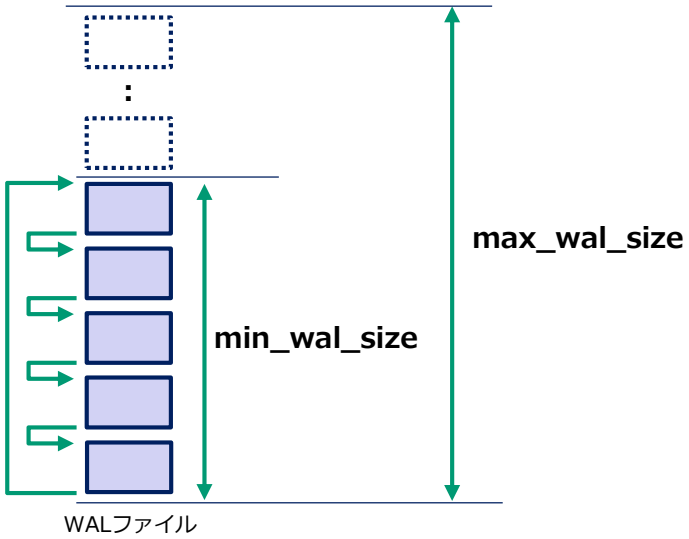
checkpoint_timeoutパラメータ　デフォルト：5min

チェックポイントの間隔を指定する、時間契機パラメータです。
一般的にはデフォルトの5分よりも大きな値に指定します。

min_wal_sizeパラメータ　デフォルト：80MB

max_wal_sizeパラメータ　デフォルト：1GB

WAL ファイルの合計サイズの下限と上限を指定する、サイズ契機パラメータです。
また、WALがこれらのサイズに達した際にチェックポイントが実行されます。
時間契機のチェックポイントが中心となるように、通常は、本パラメータを大きな値に調整します。



- WALファイルは16MBであるため、デフォルトでは最小5個、最大64個
- 通常時、デフォルトではWALファイルは5個で循環利用
- 循環利用時、WALファイルを上書きできない場合（対象WALファイルのアーカイブが終わっていない場合など）、max_wal_sizeまでWALファイルを追加
- つまりWALファイルが一巡して上書きされる(消失する)する前に、チェックポイントによりダーティバッファをディスクにフラッシュする動作となる

チェックポイントのパラメータ ②

checkpoint_completion_targetパラメータ

デフォルト：0.9

チェックポイントの完了目標をチェックポイント間の総時間の割合として指定します。
チェックポイントにおけるダーティバッファの書き込み時間を分散させることで、I/Oをピークをなだらかにすることで
パフォーマンス改善が期待できます。

- 設定例と動作

checkpoint_timeout	Version	checkpoint_completion_target	動作
10分	～13	0.5	10分×0.5 となり、5分を掛けてチェックポイント処理を試行します
	14～	0.9	10分×0.9 となり、9分を掛けてチェックポイント処理を試行します

v13までの場合は、0.9に調整して、チェックポイント発生時のI/O負荷軽減を図ります。

チェックポイントの分析

■pg_stat_bgwriterビューによる確認

チェックポイントの発生状況をpg_stat_bgwriterビューで確認します。
チェックポイントは、時間契機（checkpoints_timed）による発動をベースにして、サイズ契機（checkpoints_req）を抑止することが理想です。理由は、時間契機の発動を30分程度に制御した方が良いためです。

- pg_stat_bgwriterビューによるヒット率の確認

```
db1=# ¥x
db1=# SELECT * FROM pg_stat_bgwriter;
-[ RECORD 1 ]---------+---------------
checkpoints_timed     | 170
checkpoints_req       | 12
      :                 :
```

■サーバログファイルへの追加ログによる確認

log_checkpointsパラメータをonとすることで（デフォルトはv14までoff、v15以降はon）、
チェックポイントで書き出されたバッファ数や所要時間などがサーバログファイルに記録されます。

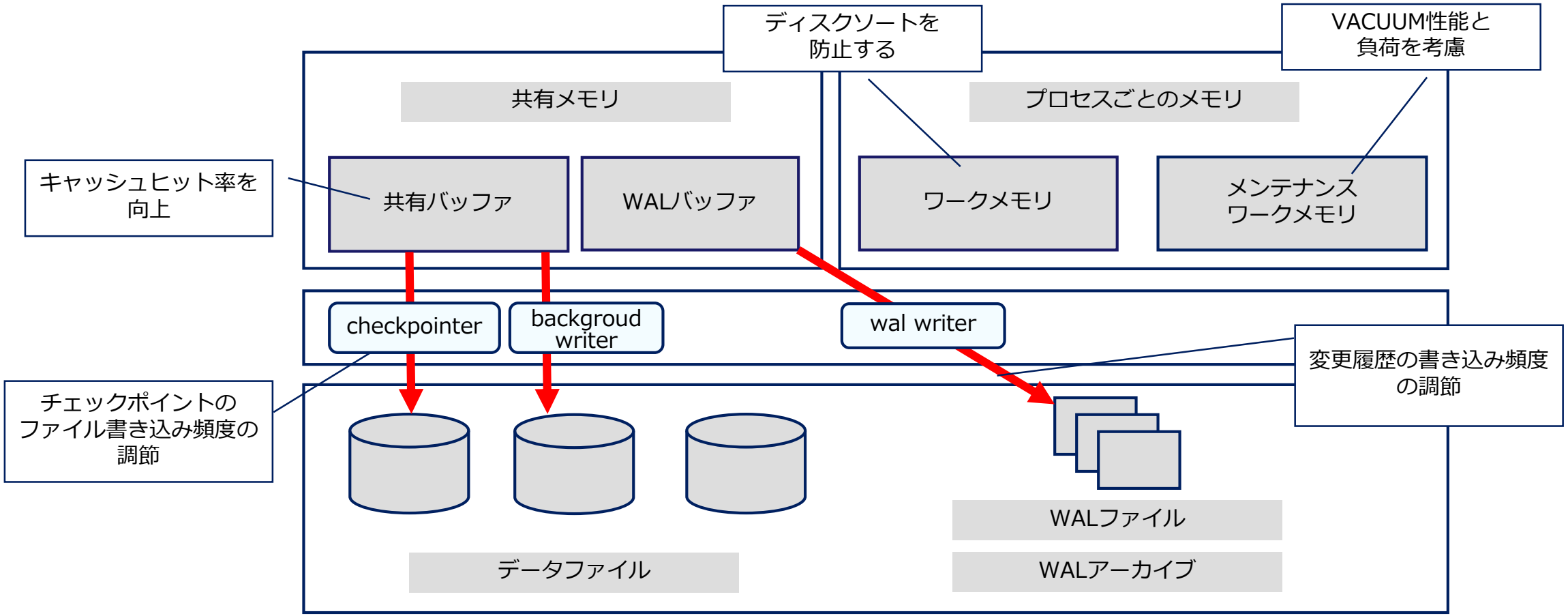
- log_checkpointsによる追加ログサンプル

```
2024-10-10 16:57:03.418 JST [2108] LOG:  checkpoint complete: wrote 48 buffers (0.3%); 0 WAL file(s) added ・・・
```

※一般的なOLTP環境では、チェックポイントは数十分間隔（例 30分程度)が好ましいと言われています。
　これを指針として適切かどうかを確認することをお勧めします。

ディスクI/O関連

共有バッファにあるダーティバッファはcheckpointer/background writer、WALバッファにある変更履歴は
wal writerがディスクに書き出します。
これらプロセスの書き込みタイミング等を適正化することでパフォーマンスの向上が期待できます。

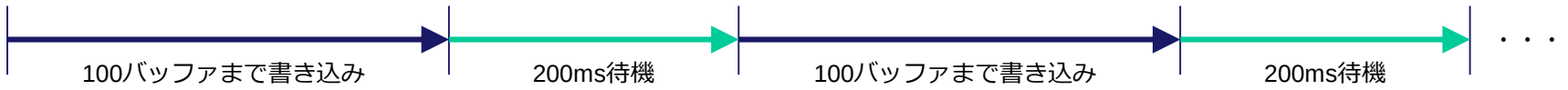


ディスクI/Oのパラメータ ①

bgwriter_delayパラメータ　デフォルト：200ms

background writerによるデータファイルへのダーティバッファの書き込みを待機する時間を指定します。

background writerがダーティバッファを書き込む際、書き込みが一定バッファに達すると（bgwriter_lru_maxpages：デフォルト100）、bgwriter_delayに指定した値（デフォルト200ms）を待機します。



これにより、ディスクI/Oの負荷を分散させ、ピーク時のパフォーマンスの改善が期待できます。

ディスクI/Oのパラメータ ②

wal_sync_methodパラメータ

デフォルト：fdatasync | fsync | fsync_writethrough | open_datasync | open_sync

wal writerがWALファイルへの書き込みにおいて、内部的に使用する同期書き込み方式を指定します。

- fdatasync　　：fsync()を呼び出し、データ部分のみを強制書き出し
- fsync　　：fdatasync()を呼び出し、データ部分とメタデータを強制書き出し　※Linuxのデフォルト
- fsync_writethrough　　：fsync()を呼び出し、データ部分とメタデータをディスクキャッシュをバイパスして強制書き出し
- open_datasync　　：ファイルをO_DSYNCフラグで開き、データ部分のみを強制書き出し
- open_sync　　：ファイルをO_SYNCフラグで開き、データ部分とメタデータを強制書き出し

例えば、fsyncやfsync_writethroughはデータの整合性は高まりますが、パフォーマンスが犠牲になります。
逆にfdatasyncはパフォーマンスが良い代わりに、メタデータを同時に書き込まないため、
データの整合性に注意する必要があります。

また、プラットフォーム(OS)によって、パラメータ値のサポート/非サポートがある点を注意してください。

従って、本パラメータはパフォーマンスとデータの整合性 と プラットフォーム対応の有無等を踏まえて設定します。

※pg_test_fsyncユーティリティを使用すると、各設定値におけるパフォーマンステストが行えます。

ディスクI/Oのパラメータ ③

fsyncパラメータ

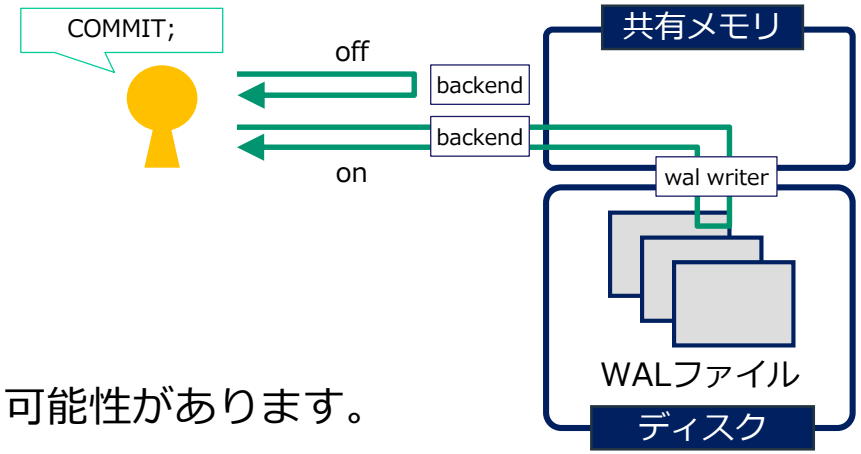
デフォルト：on | off

ダーティバッファやWALをディスクに書き込む際、fsyncシステムコール(データが実際にディスクに書き込まれたことを確認)を利用するかどうかを指定します。
デフォルトはonであり、書き込み中に障害が発生してもデータの整合性は保たれます。

synchronous_commitパラメータ

デフォルト：on | off | remote_apply | remote_write | local

WALファイルへのWALの書き込み完了を待たずにクライアントに
制御を戻すかどうかを指定します。
デフォルトはonで書き込み完了まで待機します。
offは非同期となり書き込み待機が無くなることで速度向上しますが、書き込み完了前に
DBが異常終了した場合、アプリケーションからのデータ復旧が必要です。
※論理バックアップをインポートして検証環境を構築する場合など、万が一データが失われても
すぐに復旧できるようなシーンで有効です。



上記 2つのパラメータを無効化している場合、非同期で応答が戻るため、
パフォーマンスの向上は期待できます。

一方、書き込み中に障害が発生すると、データが消失し整合性が取れなくなる可能性があります。

ディスクI/Oのパラメータ ④

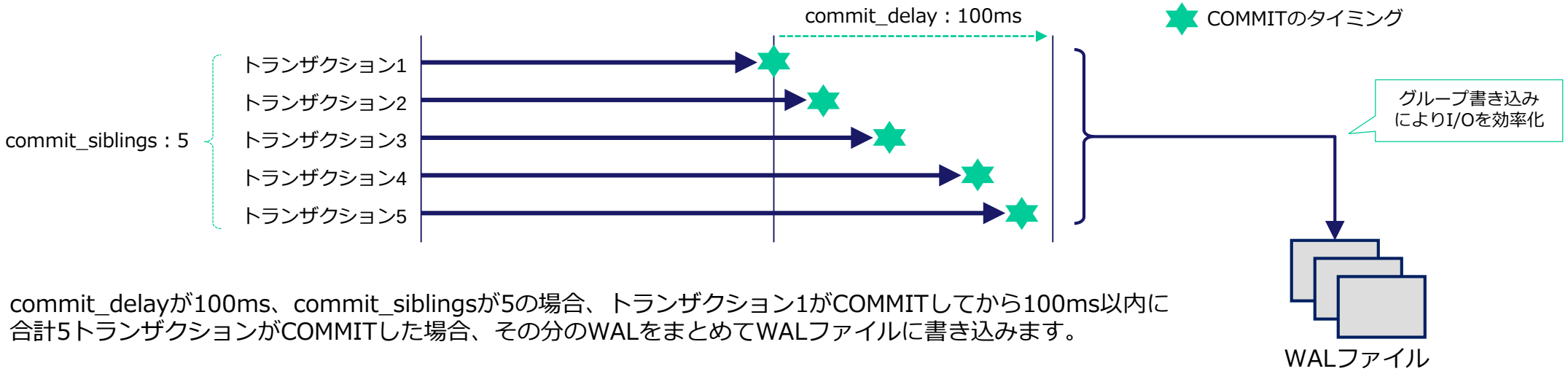
commit_delayパラメータ　デフォルト：0

WALファイルへのWALの書き込みを遅延させる時間を指定します。デフォルトは0で遅延しません。
遅延書き込みを有効にした場合、指定した時間が経過する間に複数のトランザクションでCOMMITが実行されると、
その分のWALをまとめて1回で書き込みます。

グループ書き込みにより、ディスクI/Oを効率化できます。
※遅延書き込みを行うには、fsyncがon、commit_siblingsで指定したトランザクションの数以上が存在する場合です。

commit_siblingsパラメータ　デフォルト：5

WALの遅延書き込みを有効にする、同時トランザクションの最小数です。デフォルトは5です。



commit_delayが100ms、commit_siblingsが5の場合、トランザクション1がCOMMITしてから100ms以内に
合計5トランザクションがCOMMITした場合、その分のWALをまとめてWALファイルに書き込みます。

ディスクI/Oのチューニング ⑤

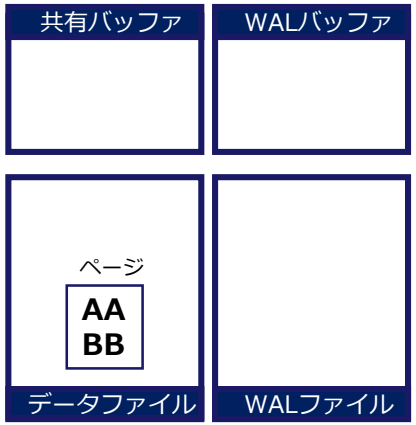
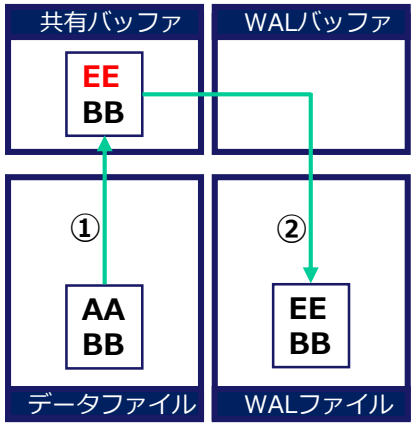
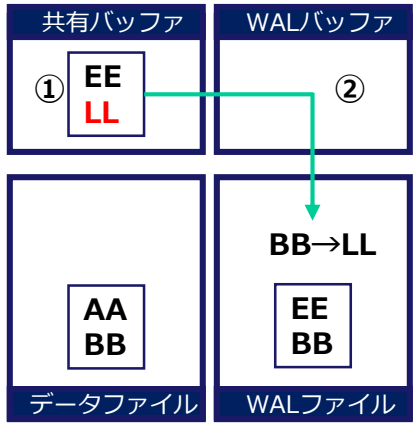
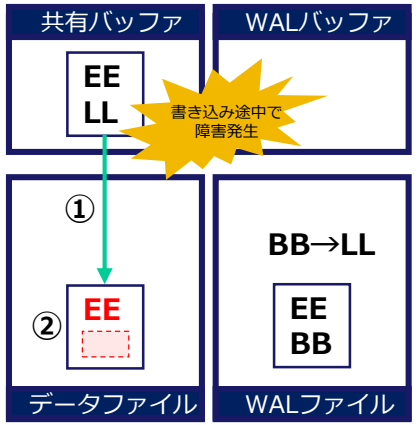
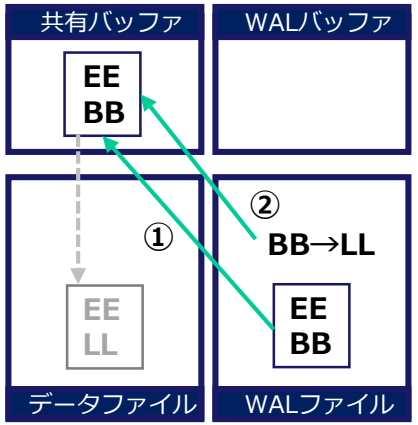
full_page_writesパラメータ

デフォルト：on | off

チェックポイントの後にそのページが最初に変更された過程で、そのページイメージをWALに書き込みます。
本機能により、チェックポイントによるページ書き込みの際、ページの一部が書き込まれたタイミングで障害が発生しても問題なく復旧できることを保証します。

本機能を「off」にすることで書き込むWALの量が減るため性能改善は期待できますが、クラッシュリカバリにおけるデータの整合性は保証されません。

- full_page_writes=on の時の動作

1．チェックポイント直後	2．UPDATE	3．UPDATE	4．チェックポイント中に障害	5．クラッシュリカバリ
				
	①ページを読込み、AA→EEに変更 ②変更履歴をWALバッファ経由でWALファイルに書き込む。 チェックポイント後の最初の変更のため、ページレベルとなる。	①ページ内のBB→LLに変更 ②該当ページは書き込み済みのため変更履歴のみをWALファイルに書き込む。	①チェックポイントが発生 ②ページ上部の「EE」を書き込んだ後、障害が発生し、下部が書き込まれないまま異常終了。 ページが中途半端な状態。	①サーバ再起動時にクラッシュリカバリを実施。中途半端なページではなく、WALファイル内のページイメージを基にする。 ②変更履歴を適用し、失われたページイメージを復旧。

ディスクI/Oの分析

PostgreSQLは、追記型アーキテクチャや変更履歴生成の特性などの理由で、
ディスクI/Oによるパフォーマンスへの影響に注意する必要があります。

例えば、iostatなどのOSコマンドでディスクの状態を監視し、DISK BusyやI/O waitsが
発生している場合は、これらのパラメータを調整します。

PostgreSQLで高パフォーマンスが求められるシステムの場合、I/O関連パラメータのチューニングだけでなく、ストレージ設計(ストレージ選択、RAID設計など)やデータベースファイルの配置設計(PostgreSQLのテーブルスペース機能)も含め考慮する必要があります。

プロセスごとのメモリとは

backendプロセスごとに、非共有のメモリが確保されます。以下の2つの領域から構成されます。

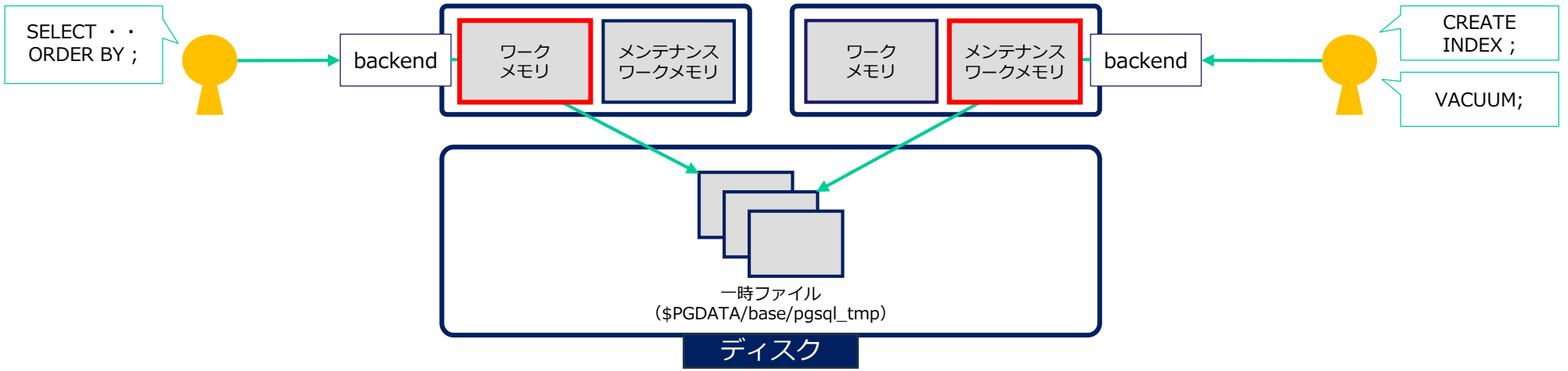
■ワークメモリ

ソート処理やハッシュ処理時に使用するメモリ領域です。

■メンテナンスワークメモリ

VACUUM、CREATE INDEXなどの内部的な保守操作で使用するメモリ領域です。

処理が各メモリ内で完了しない場合、ディスクソート（一時ファイルへの書き込み）が発生します。
このディスクソートの発生を出来る限り抑えるように調整することがチューニングのポイントです。



ワークメモリのパラメータ

work_memパラメータ　デフォルト：4MB

ソート処理などの backendプロセスのワークメモリのサイズを指定します。
ディスクソートが発生している場合、ディスクソートの発生量を踏まえて適切なサイズに設定します。
メモリ使用量としては、work_mem × max_connections(最大接続数)を考慮の上、調整する必要があります。
また、特定のバッチ処理の場合、セッション または トランザクション単位でパラメータを設定します。

hash_mem_multiplierパラメータ　デフォルト：v14まで1.0、v15以降2.0

ハッシュ処理（ハッシュ結合やハッシュ集計）で使用されるメモリ量を制御します。
work_memの値に対する倍率として指定します。
ハッシュ処理で扱うデータ量がある程度見積りできる場合、適切に設定するとハッシュ処理のパフォーマンスを向上できます。

- 設定例と動作

work_mem	hash_mem_multiplier	動作
4MB	1.0	ハッシュ処理に最大4MBを使用
	2.0	ハッシュ処理に最大8MBを使用

メンテナンスワークメモリのパラメータ

maintenance_work_memパラメータ

デフォルト：64MB

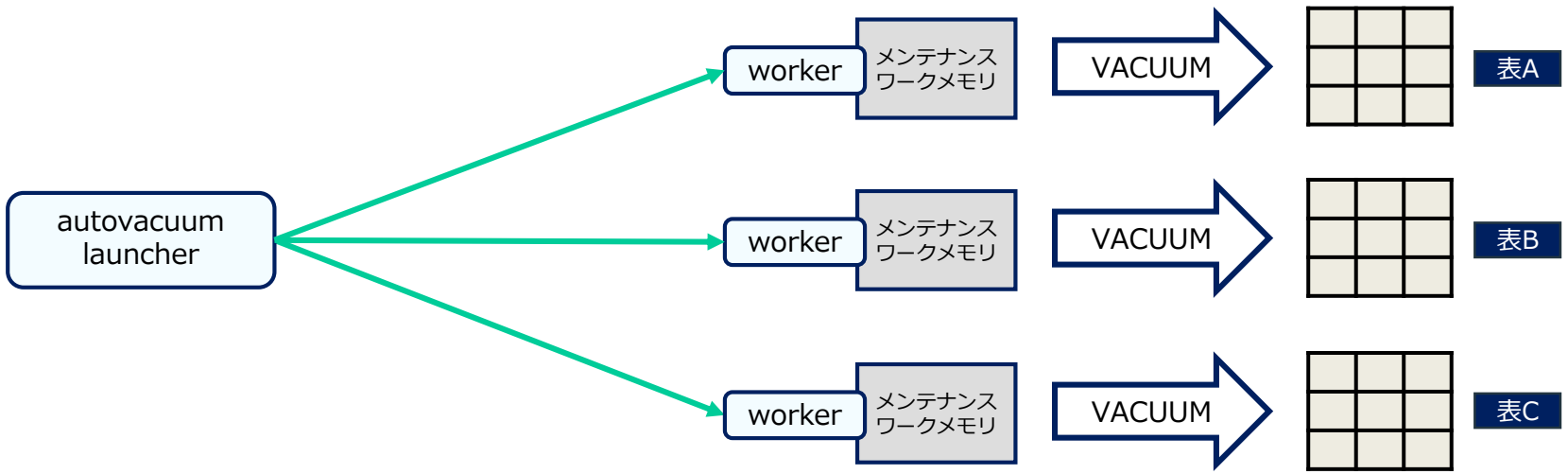
メンテナンスワークメモリのサイズを指定します。

autovacuum_work_memパラメータ

デフォルト：-1（maintenance_work_memと同じ）

自動VACUUM用のワーカプロセスが使用するメモリサイズを指定します。
デフォルトは-1で無効になっており、maintenance_work_memの値が使用されます。

自動VACUUMでは、autovacuum_work_mem× autovacuum_max_workers（デフォルト3）分のメモリが消費されるため、メモリが枯渇しないよう調整する必要があります。



プロセスごとのメモリの分析

■ディスクソートの発生状況の確認

log_temp_filesパラメータでディスクソートが発生した処理の詳細をサーバログファイルに出力できます。

- -1　　：出力しない　※デフォルト
- 0　　：すべてのディスクソートの情報
- 任意のサイズ　：指定サイズ以上のディスクソートの情報

● log_temp_filesによる追加ログサンプル

一時ファイルのサイズ

```
[2024-10-10 13:31:13.721 JST] postgres db1 26334[3] POS-00000 LOG: temporary file: path "base/pgsql_tmp/pgsql_tmp26334.0", size 45481984
[2024-10-10 13:31:13.721 JST] postgres db1 26334[4] POS-00000 STATEMENT: SELECT * FROM pgdb1_accounts ORDER BY bid;
```

該当のSQL

※対象のSQLが明確である場合、EXPLAIN ANALYZEでもディスクソートの詳細を確認できます。

■自動VACUUMの状況の確認

log_autovacuum_min_durationで、長時間かかっている自動VACUUM処理をサーバログファイルに出力できます。

- -1　　：出力しない　※デフォルト
- 0　　：すべての自動VACUUMの情報
- 任意の時間(ms)：指定時間以上を要した自動VACUUMの情報

● log_autovacuum_min_durationによる追加ログサンプル

```
[2024-10-21 13:11:23.721 JST] postgres db1 26334[3] POS-00000 LOG:  automatic vacuum of table "test.pgbench_accounts": index scans: 1
        pages: 14640 removed, 0 remain, 14640 scanned (100.00% of total)
        ・・・
        system usage: CPU: user: 1.09 s, system: 0.18 s, elapsed: 78.59 s
```

自動VACUUMの経過時間

模擬問題

サンプル問題 ①

□ **PostgreSQLのパラメータの説明として、適切でないものを 2 つ選びなさい。**

□ A） checkpoint_timeoutパラメータは、チェックポイントの間隔を決定する時間契機パラメータである。

□ B） work_memパラメータは、自動 VACUUM で使用するメモリーサイズを設定する。

□ C） checkpoint_completion_targetパラメータは、チェックポイント処理を完了するまでの時間目安を設定する。

□ D） チェックポイントによる負荷を考慮し、チェックポイントは可能な限り時間契機で実行されるように調整すべきである。

□ E） wal_buffersパラメータは、WALバッファに割り当てるサイズを設定する。

サンプル問題 ①：回答

正解：B

A）　checkpoint_timeoutパラメータは、チェックポイントの間隔を決定する時間契機パラメータである。

正しい説明のため、不正解です。

B）　work_mem パラメータは、自動 VACUUM で使用するメモリーサイズを設定する。

正解です。

VACUUM などのメンテナンス処理で使用するメモリーサイズは maintenance_work_mem で設定します。

C）　checkpoint_completion_targetパラメータは、チェックポイント処理を完了するまでの時間目安を設定する。

正しい説明のため、不正解です。

D）　チェックポイントによる負荷を考慮し、チェックポイントは可能な限り時間契機で実行されるように調整すべきである。

正しい説明のため、不正解です。

サイズ契機パラメータ（max_wal_size/min_wal_size）によるチェックポイントの発生が多い場合、これらのパラメータの値をより大きくすることなどを検討します。

E）　wal_buffers パラメータは、WAL バッファに割り当てるサイズを設定する。

正しい説明のため、不正解です。

サンプル問題 ②

□ **ハッシュ処理の際に使用されるメモリー領域の説明として、適切なものを 2 つ選びなさい。**

- □ A） hash_mem が使用される。
- □ B） 使用するメモリーの最大サイズは work_mem×hash_mem_multiplier で計算できる。
- □ C） work_mem が使用される。
- □ D） work_mem を超えるサイズのメモリーが使用されることはない。
- □ E） hash_mem_multiplier のデフォルト値は 0.5 である。

サンプル問題 ②：回答

正解：B、C

A）　hash_mem が使用される。

不正解です。

hash_mem というパラメータは存在しません。

B）　使用するメモリーの最大サイズは work_mem×hash_mem_multiplier で計算できる。

正解です。

C）　work_mem が使用される。

正解です。

ハッシュ処理には work_mem が使用されます。

D）　work_mem を超えるサイズのメモリーが使用されることはない。

不正解です。

使用するメモリーの最大サイズは work_mem × hash_mem_multiplier で計算されるため、

hash_mem_multiplier パラメータの値によっては work_mem パラメータで指定したサイズ以上のメモリーが

使用される可能性があります。

E）　hash_mem_multiplier のデフォルト値は 0.5 である。

不正解です。

hash_mem_multiplier パラメータのデフォルト値は v14 までは 1.0、v15 以降は 2.0 です。

参加特典

本セミナーの受講者限定で、Udemy OSS-DB Silver/Goldの問題集を、
それぞれ**52**% Off(4,200円→2,000円)特別価格にてご購入いただけます。

【Udemy】OSS-DB Silver Ver3.0 試験対策問題集

Udemy
OSS-DB Silver

【Udemy】OSS-DB Gold Ver3.0 試験対策問題集

Udemy
OSS-DB Gold

■ **注意事項**
上記のアイコンをクリックして専用Webページにアクセスして下さい。
※他からアクセスした場合、特別価格が適用されません。

■ **有効期間**
本日から **5**日間 です。
（2024年10月30日(水) 19:15 ～ 2024年11月04日(月) 19:15）