

PostgreSQL導入に向けての取り組み ～大規模システムへの適用を目指して～

NTTコムウェア株式会社 基盤技術本部

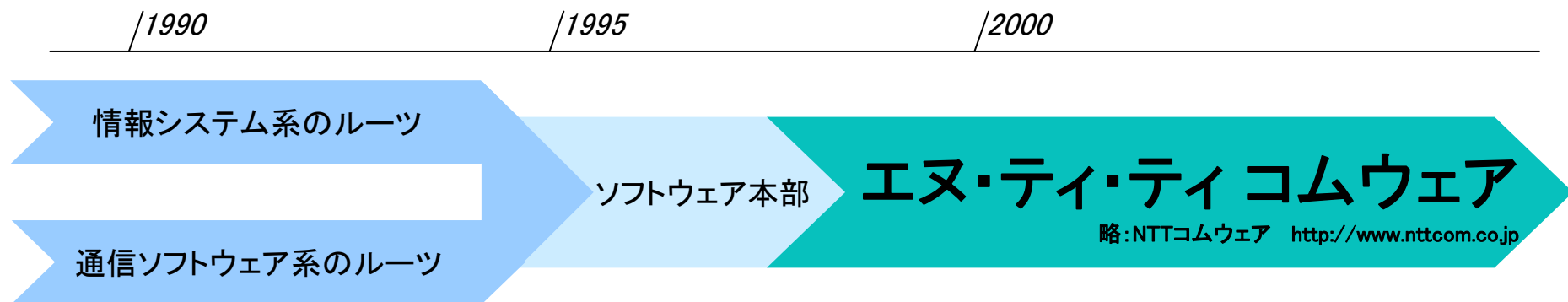
山内孝彦 吉田敏和

心をつなぐ、社会をつなぐ



1. NTTコムウェアの紹介

通信技術と情報技術の融合で幅広くシステム開発を実施！



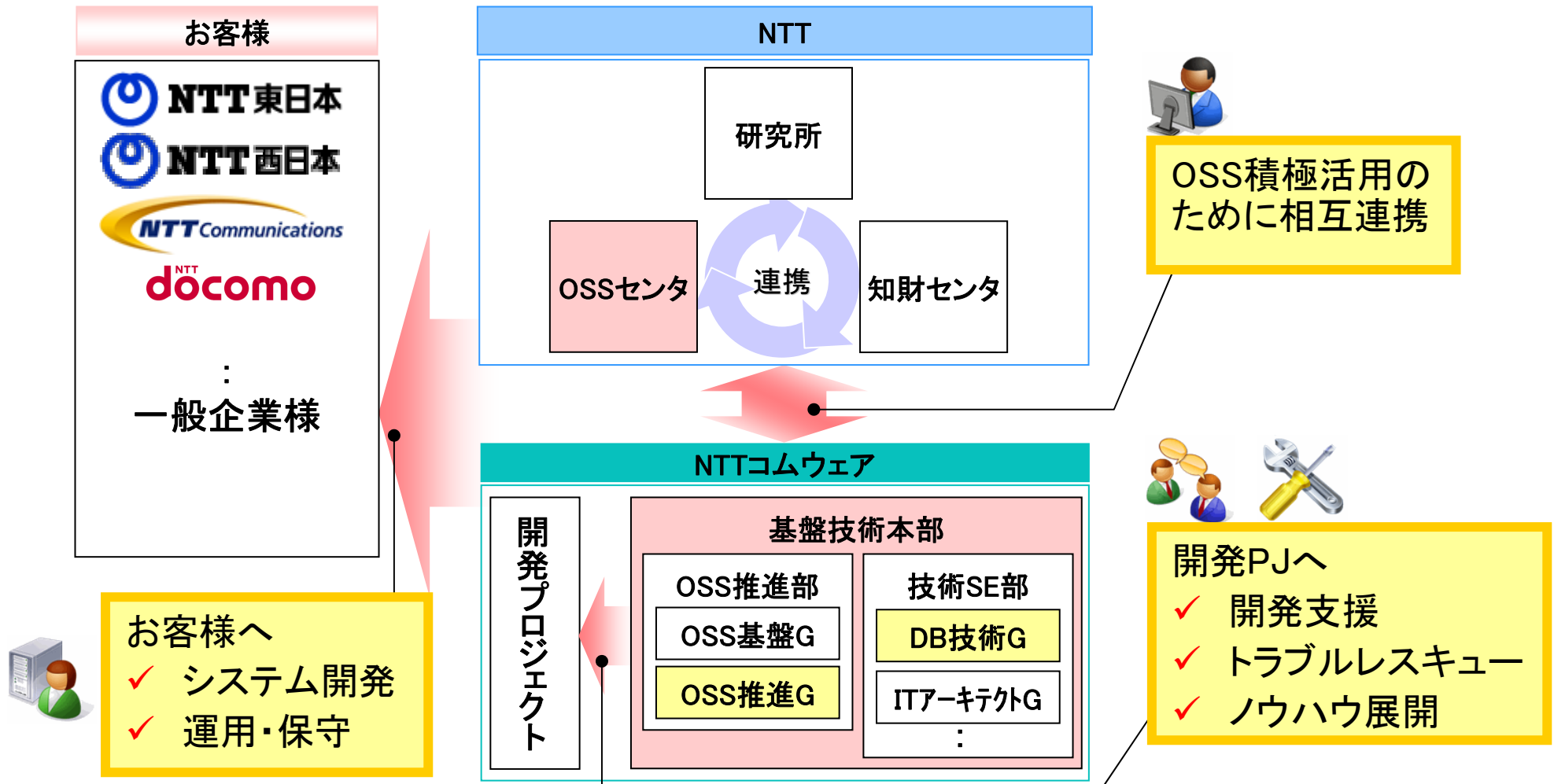
【コアコンピタンス】

- ✓ 次世代インフラ構築を支える総合力で、先進のIPネットワークソリューションを提供
- ✓ ユビキタス社会の実現に向けた、高付加価値ソリューションを実現
- ✓ 万全な運用・管理で、ミッションクリティカルな情報システムを支援

➔ NGNに関わる通信制御・アプリケーションサービス開発などによりNGNの拡大・普及を推進
もちろん、NTTグループ以外のお客様へのSIerとしても幅広く活動

1.1 自組織の役割(1/2)

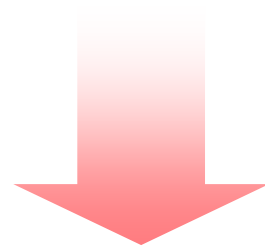
NTTグループ全体で「TCO削減」のためOSSを積極活用！
技術集約により技術力/開発力を向上させ開発PJへ技術支援を実施



2. システムへ適用する際の着目点

PostgreSQLを活用し、
お客様へ高品質・高信頼なサービスをいかに安く提供できるかが重要！

- 商用製品と変わらず、システムの品質・信頼性に妥協は許されない
- OSSを採用するからには、TCO削減効果が求められる



お客様へPostgreSQLを適用する際の着目点

- 性能面
- 運用面
- コスト面

） について以降で紹介

2.1 性能面・運用面の着目点

適用検討の際に性能面で着目する点

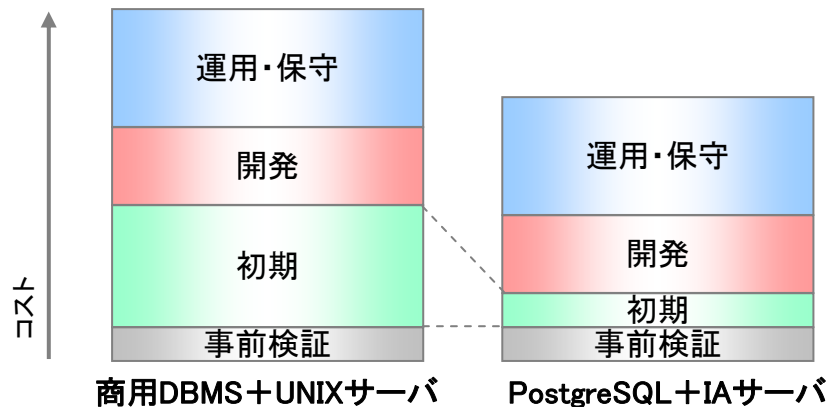
- 商用DBMSと比較して更新性能はどれほどか
 - ✓ 更新が苦手なDBMSと耳にしたことがあるけど実際は？
- 継続的に安定した性能でサービスを提供できるのか
 - ✓ I/O処理、不要領域回収処理などに対する性能安定性が気になる
- 大規模システムのスケールアップ要求を満たせるのか
 - ✓ サービス開始から5年以上のサービス拡大に対する拡張性がほしい
- 大量バッチ処理が与えられた時間内に完了できるか
 - ✓ 数千万のデータを扱う大量バッチを夜間の限られた時間内で処理したい

適用検討の際に運用面で着目する点

- 問題発生時に迅速な解析ができるか
 - ✓ ログ整理に労を費やしたり、解析情報の不足による再現待ちにはしたくない
- 運用開始後の実行計画変更のリスクを最小化できるか
 - ✓ 万が一発生した場合にお手上げでは問題
- DBメンテナンスによるサービスへの影響はどれほどか
 - ✓ 「CLUSTER」時にお客様へのサービスを停止する必要がある
- 監査に対応できるか
 - ✓ 性能への影響を最小限にし、必要な情報のみを記録したい

2.2 コスト面の着目点

■ライセンス費用だけでなくシステム全体としてのコストに着目



トータルコスト内訳のイメージ
(新規開発でTCO削減に成功した単純イメージ)



初期費用(ライセンス)が削減できると思いきや…
システム全体としてのコストが想定以上にかかって
しまいTCOが増大する可能性を意識する必要あり。

■PostgreSQL適用検討の際にコスト面で着目する点

■追記型を考慮したストレージコスト・設計コストはどれほどか

☑追記型であるが故にコストがかかる部分もあるのでは？

5.1章にて

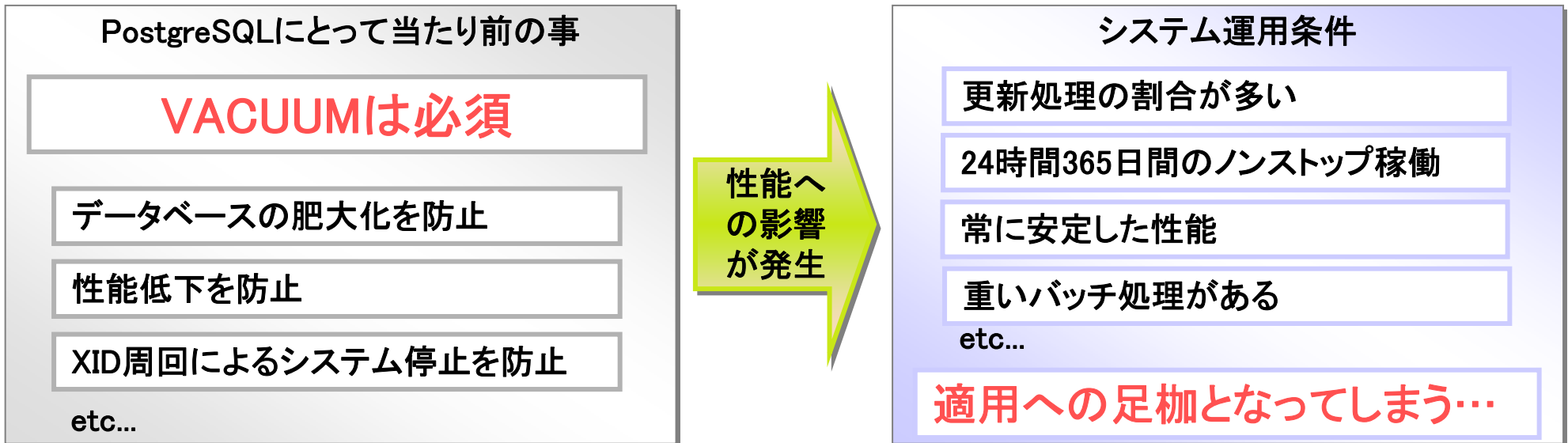
■商用DBMSからのマイグレーションコストはどれほどか

☑PostgreSQLに移行する方がコスト高になる場合もあるのでは？

5.1章にて

3.1 処理性能と安定性

■ PostgreSQLで処理性能と安定性を確保するためには…

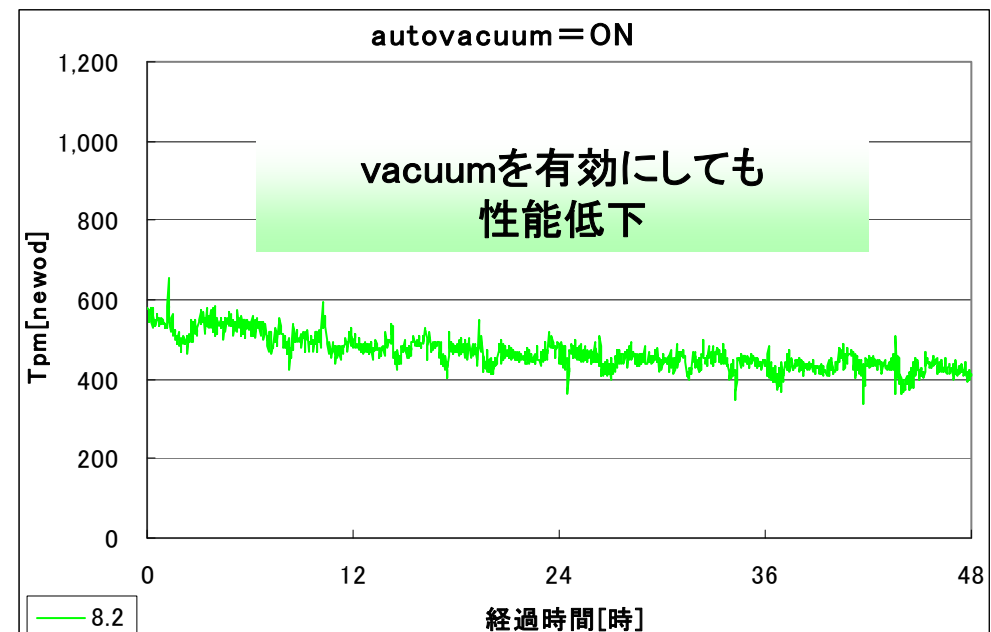
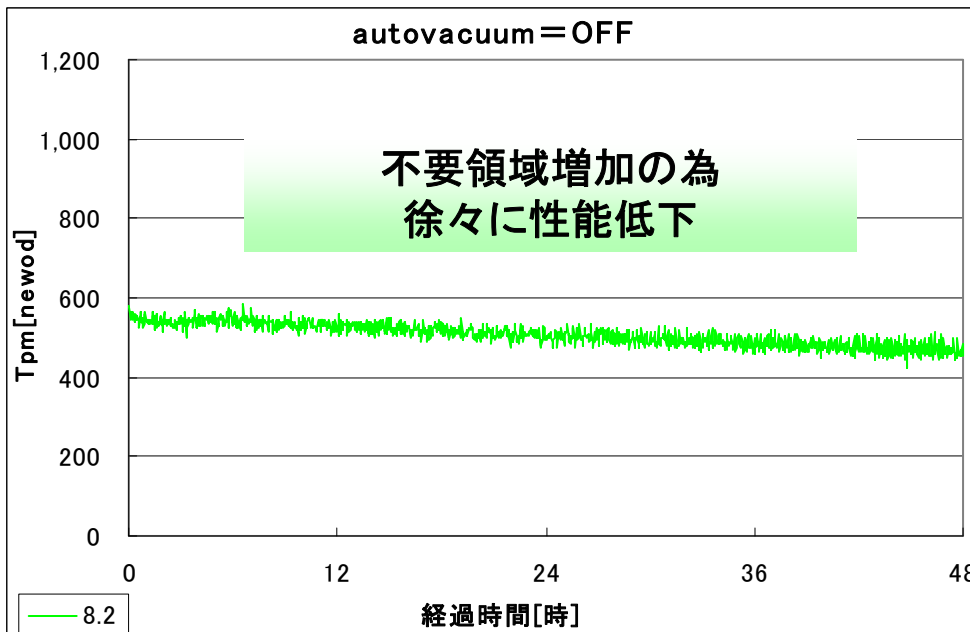


PostgreSQLの場合は「追記型アーキテクチャ」の特性を考慮することが肝心！

検証によって
解決策を検討

3.1.1 処理性能と安定性ーver. 8.2ー

■ PostgreSQL 8.2の長時間運転安定性検証結果(1/2)



autovacuumは複数のテーブルに対して、パラレルにvacuumが実行できない。

→大きなテーブルのvacuumが始まる

→他のテーブルがvacuum待ちになる

→不要領域が溜まっていく

システム適用のために
検討すべきことは？

3.1.1 処理性能と安定性ーver. 8.2ー

■ PostgreSQL8.2の長時間運転安定性検証結果(2/2)

PostgreSQL8.2をシステムへ適用する際に必須とした検討内容を以下に示す。

検討1:TBL単位で適切なVACUUM実行時間帯を検討

■ autovacuumは使用しない

- 閑散期にVACUUMを実行する。不要領域増加による性能低下がシステムの許容範囲かを検討。

■ XID周回エラー防止

- 業務で更新をしないテーブルに対し、定期的(年毎とか)にvacuumを実行。

検討2:テーブルの断片化対策を検討

- 「CLUSTER+ANALYZE」実行のため、システム停止を伴う、定期メンテナンス(半期毎とか)を行う。

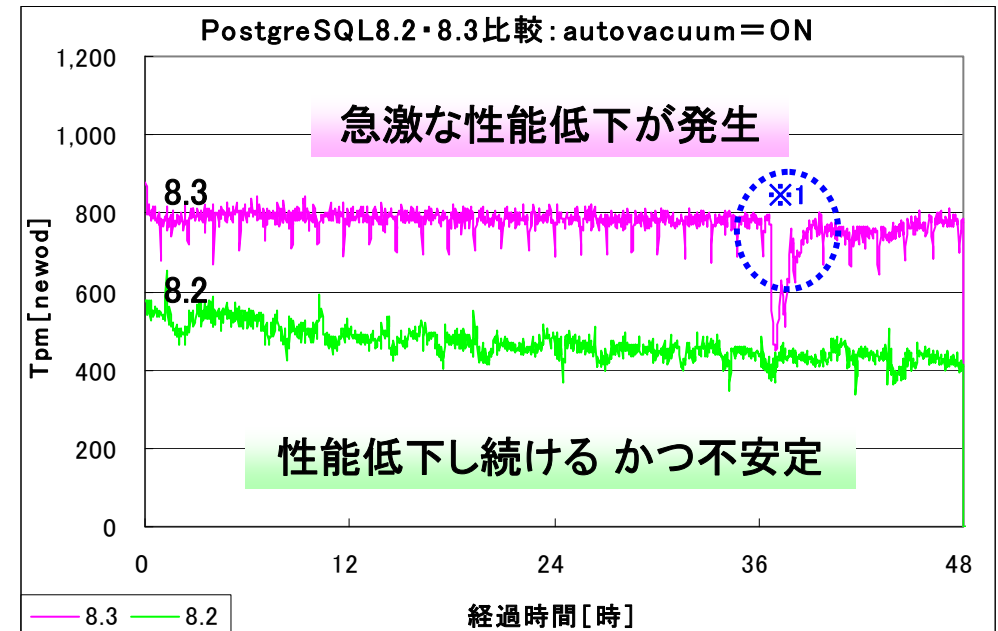
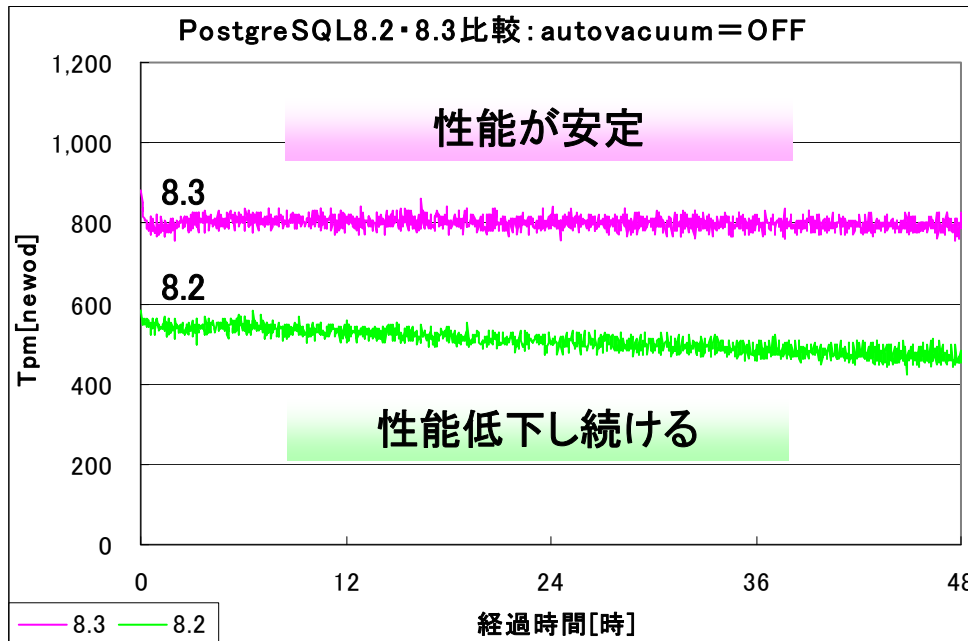


業務の閑散期は「バッチ処理(データバックアップや収集データの分析処理)」
が稼動する場合が多く、VACUUM実行時間帯の確保が困難。

PostgreSQL8.2でのシステム導入は見送られることが多かった。

3.1.2 処理性能と安定性ーver. 8.3ー

■ PostgreSQL8.3の長時間運転安定性検証結果(1/3)



※1: vacuumの実行時間は126分



【autovacuum=off】: PostgreSQL8.3は性能が安定

→FILLFACTOR、HOTがとても有効に機能した為。

主にランダムなIDをキーに更新するベンチマークなので、ページ内に複数行の更新が同時に発生しづらい為、有効に機能した。



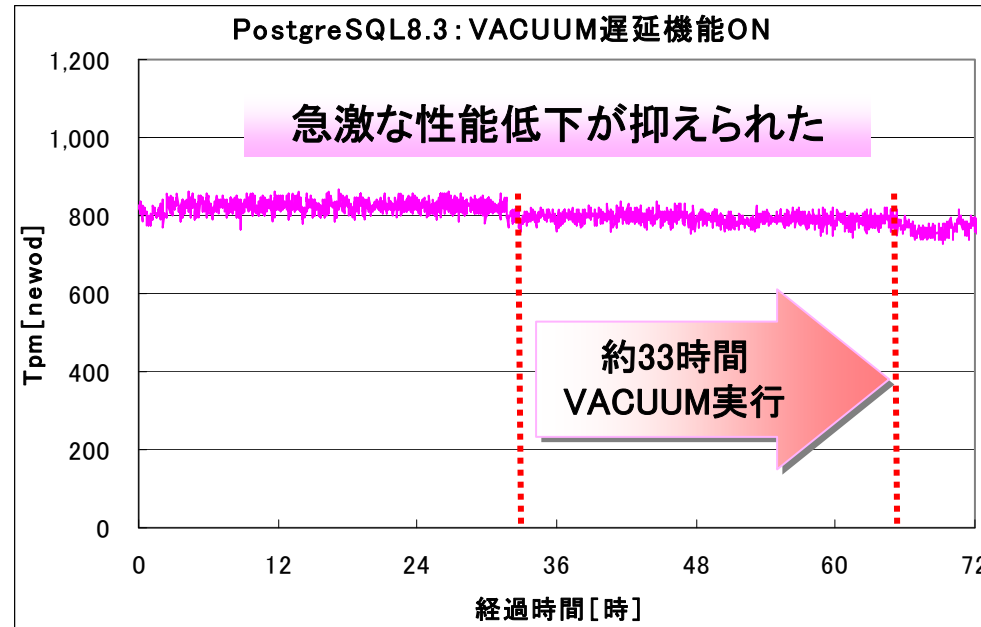
【autovacuum=on】: PostgreSQL8.3は急激な性能低下が発生

→大きなテーブル(order_line: 約30GB)のvacuumが発生し、I/O枯渇により急激に性能が低下した。

急激な性能低下を防ぐには...

3.1.2 処理性能と安定性ーver. 8.3ー

■ PostgreSQL8.3の長時間運転安定性検証結果(2/3)



- autovacuum+vacuum遅延機能の組み合わせ
- 急激な性能低下が抑えられる
- 定期的に点在する性能低下も抑えられる

性能が安定



- vacuumの所要時間が肥大化
- 今回の場合、約16倍*時間を要した

どんな弊害がある？

システム適用のために
検討すべきことは？

*VACUUM遅延機能=OFFの
VACUUM実行時間は126分間

3.1.2 処理性能と安定性ーver. 8.3ー

■ PostgreSQL8.3の長時間運転安定性検証結果(3/3)

PostgreSQL8.3をシステムへ適用する際に必須とした検討内容を以下に示す。

検討1: 大容量TBLへのVACUUM実行による急激な性能低下を防ぐ検討

■ autovacuum+vacuum遅延機能を推奨

- autovacuum実行中の業務への影響(レスポンスタイム)、バッチ処理時間への影響(スループット)を検証により確認しておく。

■ XID周回エラー防止

- autovacuum+vacuum遅延機能を使えば考慮はிரらない。

検討2: テーブルの断片化対策を検討

■ PostgreSQL8.2と同じ

検討3: FILLFACTORの領域検討

■ HOTを有効に機能させるためには必須

- 取りすぎると性能低下する。1ページに数レコード分の空きができるような値にする。



検討すべき項目量がさほど変わらないがスループットの向上 & 性能が安定。

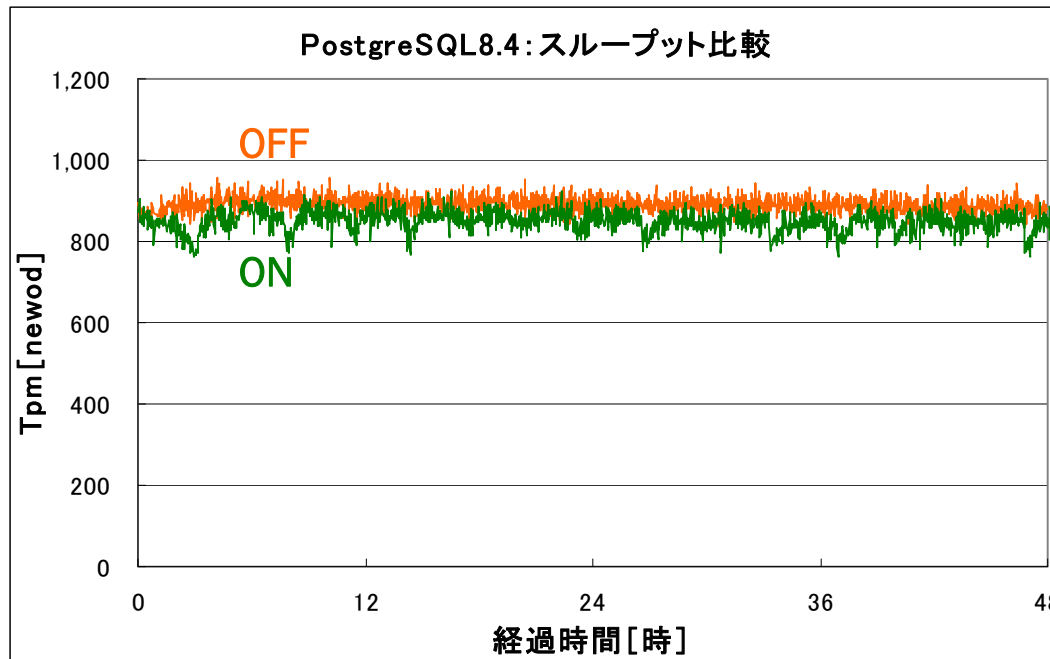
PostgreSQL8.3からシステム導入の検討対象となり、実導入も増え始めた。

(ただし、ディスク見積りに余裕を持たせる等の考慮が必要)

3.1.3 処理性能と安定性ーver. 8.4ー

■ PostgreSQL8.4長時間運転安定性検証結果

PostgreSQL8.4でautovacuumとVACUUM遅延機能をOFF同士、ON同士にして検証を行った結果を以下に示す。尚、検証環境で示したようにPostgreSQL8.4から環境が変わっている。



性能が若干安定していないのは以下の要因ではないかと考えられる。

- ・ vacuum_cost_delay等遅延機能のパラメータの値がまだ最適ではない
- ・ ANALYZE実行時の取得サンプリング数増加によるI/Oへの影響

- PostgreSQL8.3と同様にHOT機能が有効に働き、性能が安定。
- autovacuum+vacuum遅延機能を有効にしても同様に性能が安定。

PostgreSQL8.3と同様に実システムに導入しても問題ないと思われる。

3.2 スケーラビリティ

■大規模システムのスケールアップ要求を満たせるのかーCPUの観点ー

DBT-1(参照系モデル)^{*}でCPUスケーラビリティを計測した結果を以下に示す。

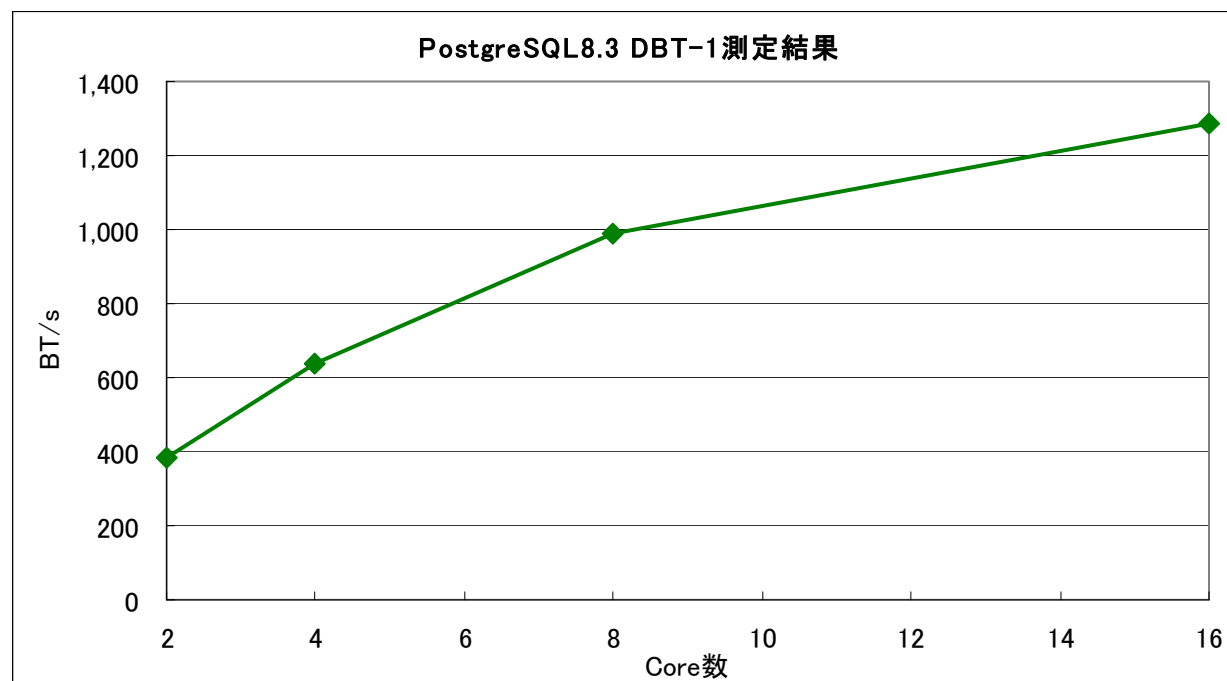


IAサーバで参照系トランザクションモデルの場合、16コアまではスケールする。



■IAサーバの場合、現時点ではOS・ハードウェアアーキテクチャの点から16コアを超えるとスケールしない。

■更新系トランザクションモデルの場合、CPUリソースを使い切る前にストレージのI/Oがボトルネックになり、スケールしない。



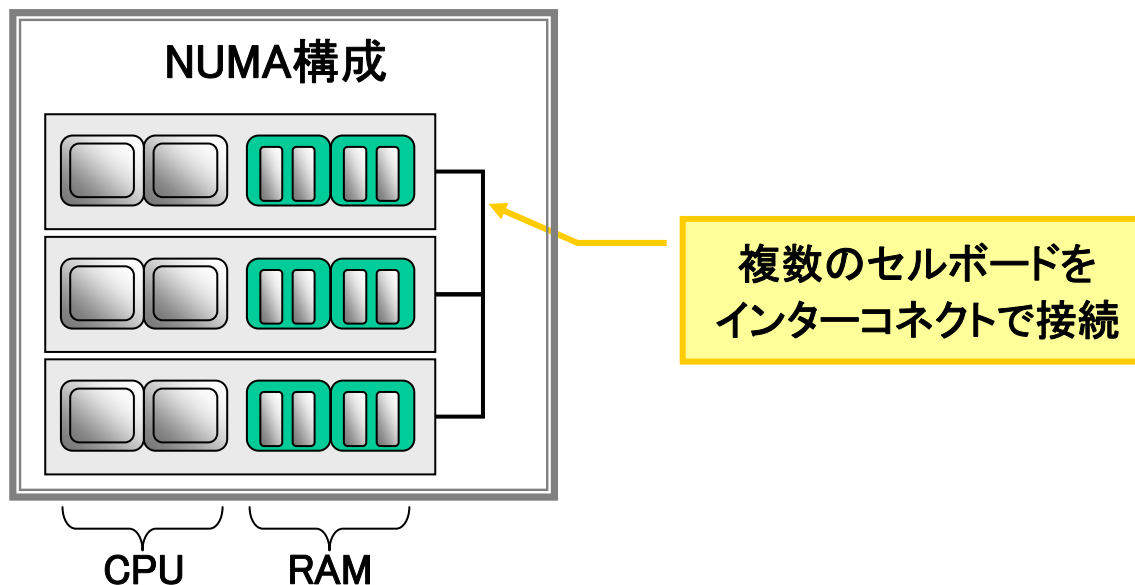
	2Core	4Core	8Core	16Core
CINT2006 Rates	33 (1.0)	62 (1.9)	119 (3.6)	209 (6.3)
PostgreSQL8.3	382.0 BT/s (1.0)	636.8 BT/s (1.7)	987.2 BT/s (2.6)	1284.7 BT/s (3.4)

データ提供元はNTT OSSセンタ、()内は2Coreの値を“1”としての比率

^{*}DBT-1は、OSDL(Open Source Development Labs)が開発したDatabase Test Suiteの1つで、TPC-W(Webコマースモデル)の仕様を部分的に取り入れたベンチマークツール

3.2 スケーラビリティ

- 大規模システムのスケールアップ要求を満たせるのかーノードの観点ー
複数ノード構成(NUMA構成)でのスケーラビリティについて以下に示す。



この検証ケースの場合は、インターコネクト経由のメモリアクセスによるオーバヘッドからラッチ競合が多発し有効なスケーラビリティを得られなかった。
→CPUを増やしてもメモリアクセスが遅延するような場合においては、スケーラビリティを得られない。

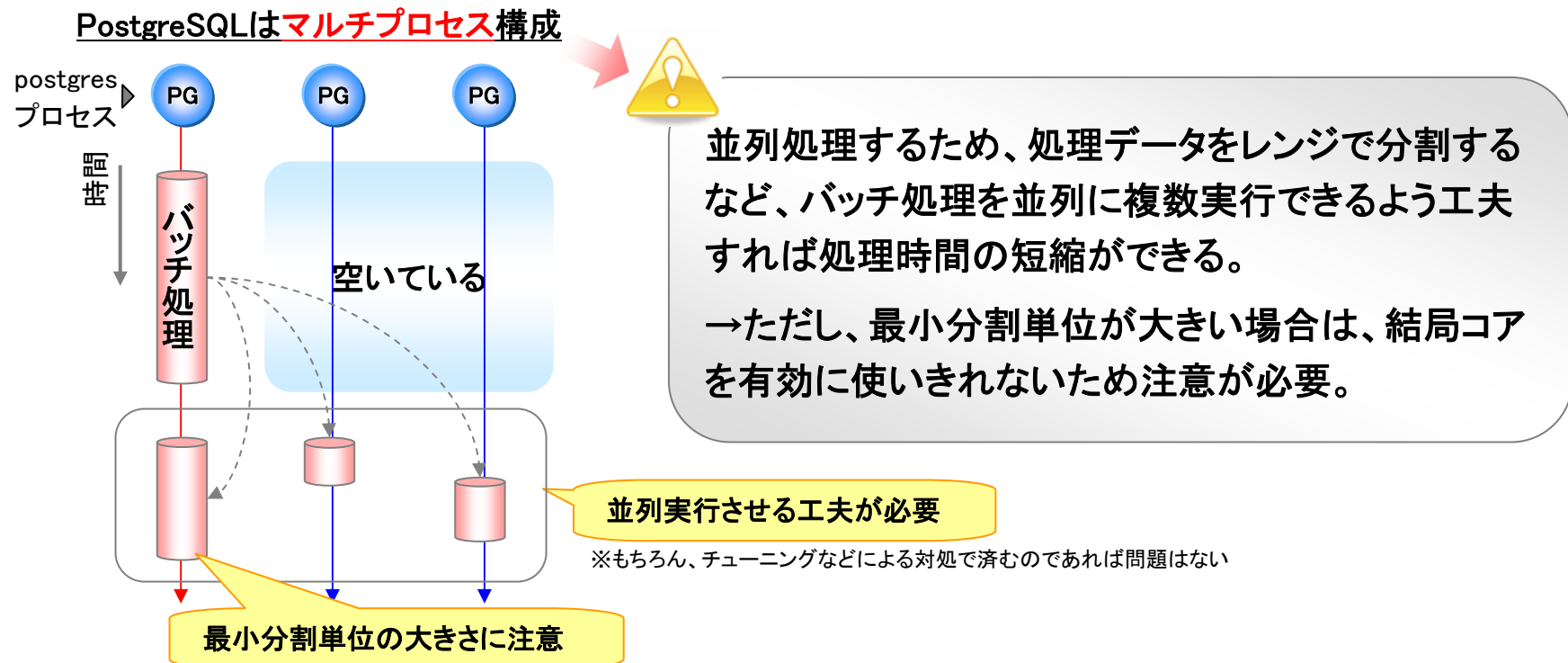
※チップレベルでNUMAを取り入れている場合など、インターコネクト経由のアクセスが高速な場合はラッチ競合は多発しないと考えられる。

3.3 バッチ処理性能

■大量バッチ処理が与えられた時間内に完了できるか

- ☑数千万の加入者を扱う大量バッチを夜間の限られた時間内で処理したい

弊社で提供しているシステムの特徴として、扱うデータ量が膨大であることが多い



4.1 性能ログ解析処理の改善

■問題発生時に迅速な解析ができるか

pg_statsinfo を使うと便利 (PostgreSQL8.3以上)

pg_statsinfoとは

- NTT OSSセンタ開発 (BSDライセンスのオープンソース)
- 統計情報のスナップショットを取得
- 解析時に見ることが多い情報を一元的に取得可能
- 大規模システムにPostgreSQLを適用する場合にはどうしても必要な機能！

DML実行回数、断片化状況、不要領域割合、
ロングランザクシオン情報、全スキャン回数等

例えばキャッシュヒットを見たい場合

詳しくは

今まで

```
SELECT
  ROUND((SUM(heap_blks_hit) /
    (SUM(heap_blks_read) + SUM(heap_blks_hit))
    * 100),2) AS "Cache Hit Ratio"
FROM
  pg_statio_user_tables;
```

Cache Hit Ratio

99.65

pg_statsinfo

DB Activity Information

```
-----
Database Size                : 52440684
Connection Average Number    : 1
Database Frozen Xid(Age)     : 5233
Commit Number                 : 5049
Rollback Number               : 0
Block Read                    : 921
Block Hit                     : 106000
Cache Hit Ratio               : 99.65
...
```

他にも多数情報取得可



- ・煩雑なSQLを作らなくても良い
- ・スナップショット機能で期間的な分析も容易になる
- ・情報取得漏れによる再現試験をする必要がなくなる

4.2 実行計画の制御

■ 運用開始後における実行計画変更のリスクを最小化できるか

- ☑ 万が一発生した場合にお手上げでは問題

対処1) SQL書き換えによる実行計画の制御

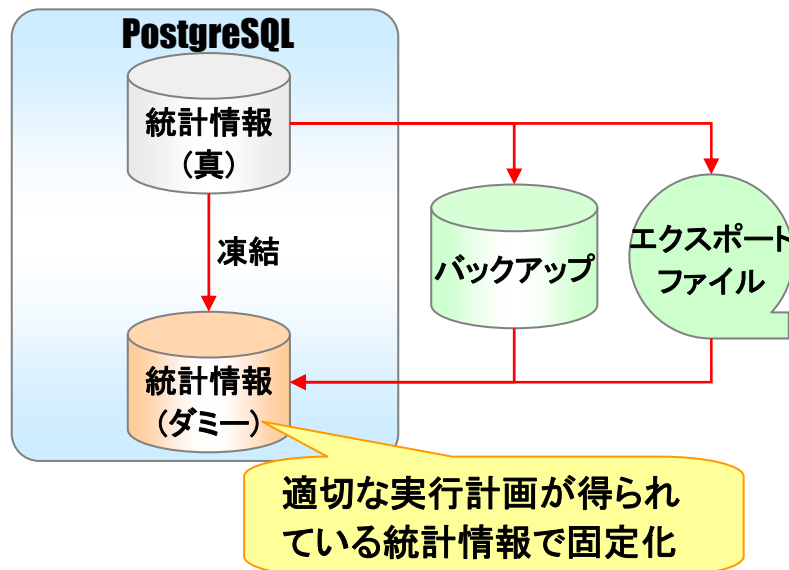
```
=# SELECT * FROM emp WHERE aid + 0 = 10;
```

Index Scan → Seq Scan

```
=# SET LOCAL enable_seqscan = off;
=# SELECT ...;
```

Seq Scan → Index Scan

対処2) 統計情報の凍結・バックアップ・リカバリ



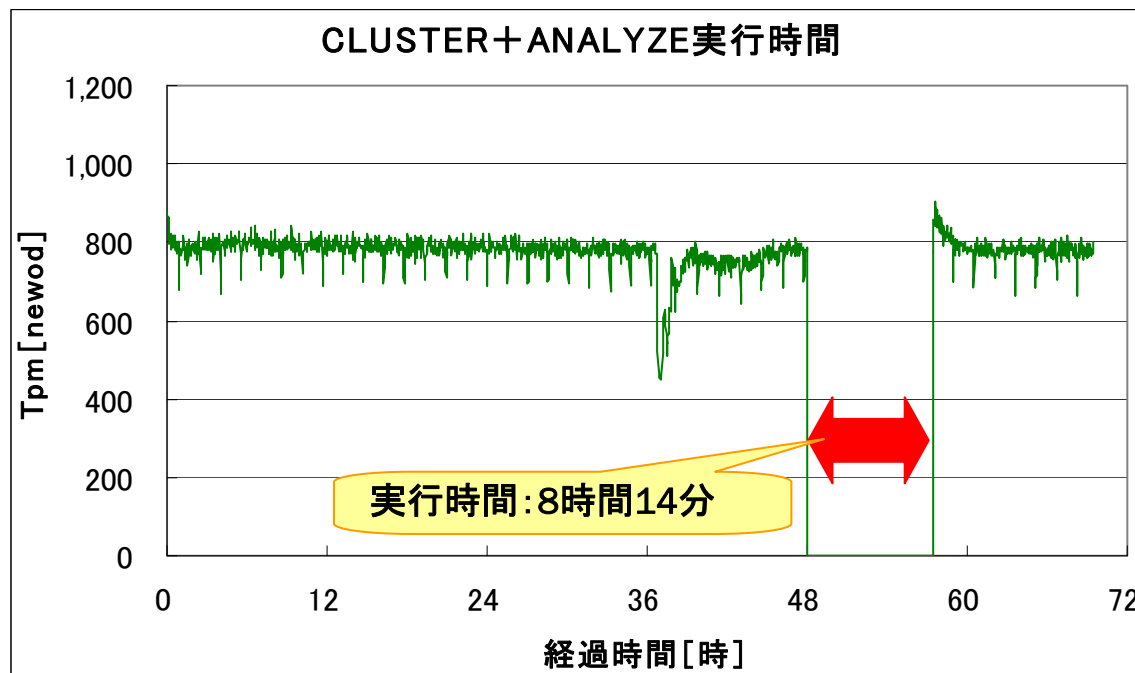
対処方法は、ケースバイケースであるが問題発生後に迅速に対応できるよう、予め対処方法について整理しておく必要がある。

→実行計画を制御する前に、SQLや設定値の見直しが可能な場合は、まずは見直しから検討すべきであることに注意。

4.3 DBメンテナンスの改善(1/2)

■DBメンテナンスによるサービスへの影響はどれほどか

長時間安定性検証を実行した後に CLUSTER と ANALYZE を実行した結果を以下に示す。



CLUSTERの特徴

- テーブル単位で排他ロック
- I/Oを著しく消費
- (最大テーブル+インデックス) 以上の空き容量が必要



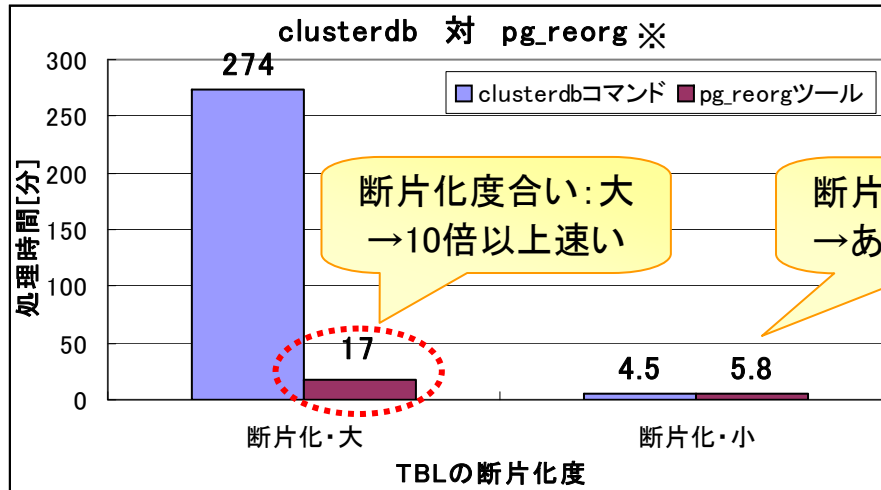
- システムの停止を伴う
- 空き容量を確保する設計が必要

そこで

4.3 DBメンテナンスの改善(2/2)

■pg_reorg の導入

PostgreSQL8.3 CLUSTER と pg_reorg の処理時間の比較検証結果を以下に示す。

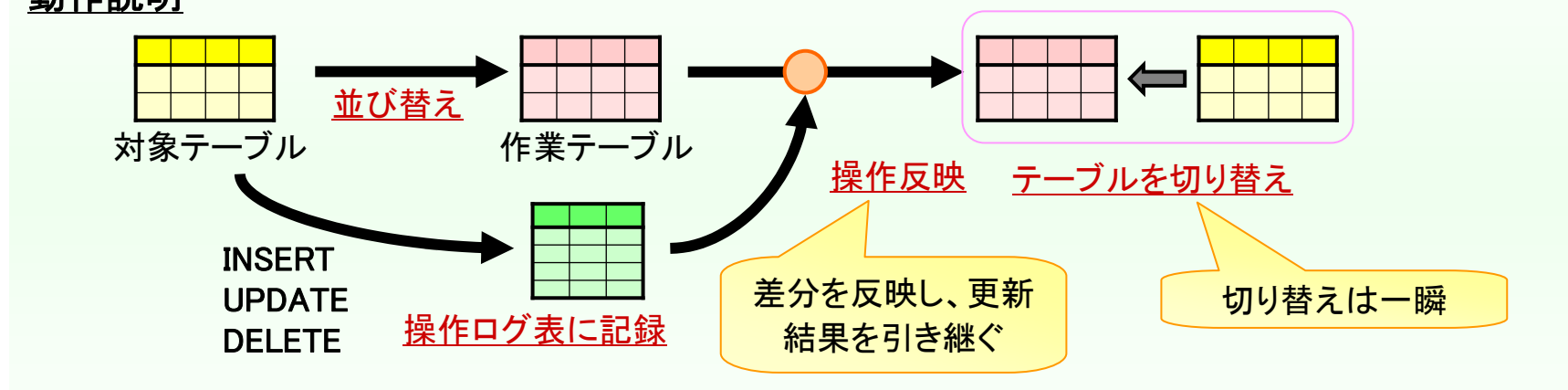


pg_reorgとは

- NTT OSSセンタ開発 (BSDライセンスのオープンソース)
- テーブルを再編成し、断片化を解消
- 再編成処理の間も参照/更新処理をブロックしないことが特徴 (言わばオンラインCLUSTER)

詳しくは [pg_reorg](#)

動作説明



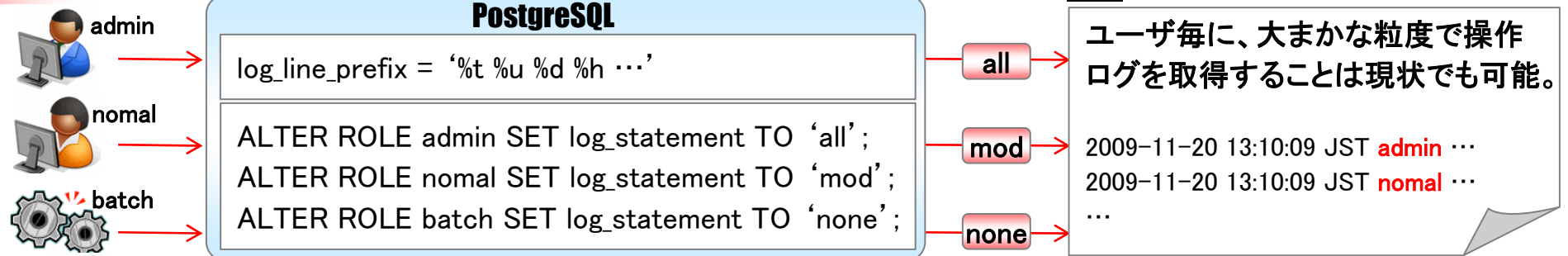
※NTT OSSセンタ提供 DBテーブル再編成機能の性能評価結果
(テーブルサイズ:2GB、内蔵ディスク1台のみ、無負荷時)

4.4 監査への対応

■ 監査へ対応可能か

- ☑ 性能への影響を最小限にし、必要な情報のみを記録したい

現状



現状では、ログ取得の粒度が粗いため、蓄積されるログの量が大きくなってしまいう傾向にある。さらに、I/Oが増大して性能へ影響を及ぼす恐れもある。

ニーズ

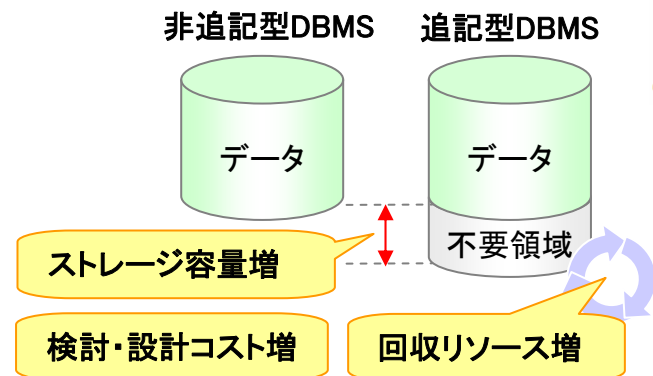


ユーザ毎の設定に加え、TBL毎・SQL毎などの細かな記録設定ができれば、監査要件の厳しいシステムにおいても適用が広がる！性能への影響は最小限に...

5. コスト面の検討結果

■ 追記型を考慮したストレージコスト・設計コストはどれほどか

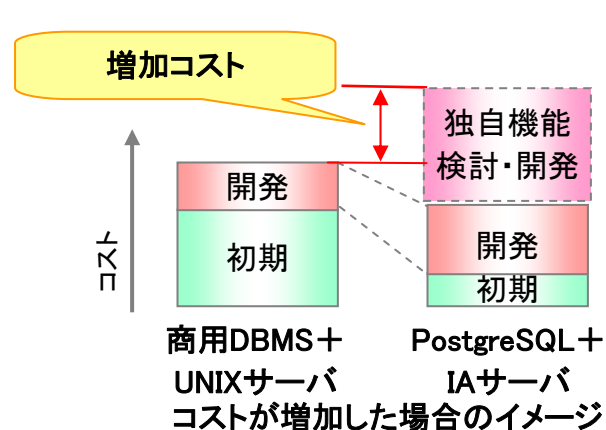
- ☑ 追記型であるが故にコストがかかる部分もある



不要領域が蓄積する分を考慮したストレージ容量の増加。また、VACUUM実行時のI/Oリソースや運用設計など、追記型に特化したコストが増加する場合があります。
→VACUUM頻度が少ない、蓄積データ量が多い、更新量が多い等の場合には特に注意が必要。

■ 商用DBMSからのマイグレーションコストはどれほどか

- ☑ 既存システムで使用している機能を検討・開発するコストに注意



PostgreSQLでは既存システムで使用している機能に相当する機能が無い場合、検討・開発が必要となる。
→マイグレーションの検討段階で、PostgreSQLでは実現不可能な機能の有無を調査し、実現するためにどのようなコストが必要なのか確認することが重要である。

6. まとめ

性能面


- PostgreSQL8.3: **HOT機能**、**autovacuum + vacuum遅延機能**を有効に活用できれば、性能アップ・安定した性能を期待できる。
- PostgreSQL8.4: 8.3と同様。

運用面

- **pg_statsinfo**: **問題発生時の対応がより迅速**に。
(性能への影響も殆ど無い為、弊社が扱うPostgreSQL導入システムへの適用を進めていく)
- **pg_reorg**: 弊社が扱うシステムはサービスダウンが許されないシステムが多いため、**pg_reorg**を有効に活用し、**サービスダウンタイムの縮小化**を図っていく。(そのための検証を今後実施予定)

コスト面

- **ストレージ**: **追記型アーキテクチャ**を意識したコスト見積もりが必要。
- **マイグレーション**: 商用DBMSの**独自機能**を**PostgreSQL上で実現するためのコスト見積もり**が重要。



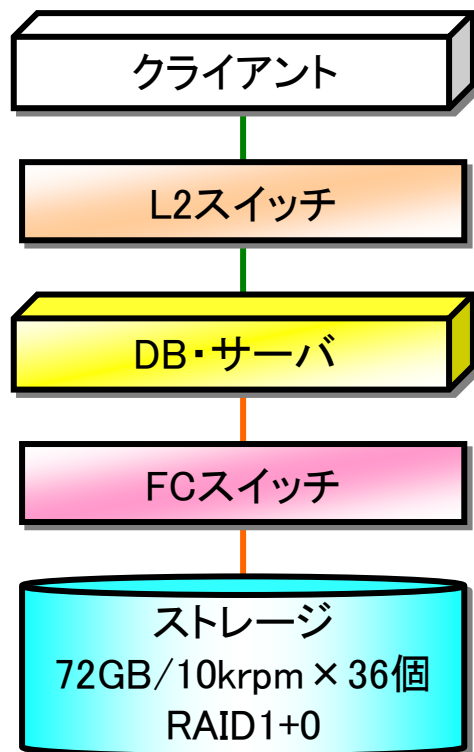
PostgreSQLが適用可能な
ケースが増えていく！

将来のPostgreSQLへの期待

- **vacuumの部分実行・一時停止機能**
→ バッチ実行スケジュールの確保、ロールバック処理の最小化
- **データファイルの破損・改ざんを検知**
→ より強固なデータベースに
- **監査機能の充実**
→ よりミッションクリティカルなシステムへ

参考. 検証環境

PostgreSQL8.2と8.3、および8.4の検証環境を以下に示す。



	PostgreSQL8.2 & 8.3	PostgreSQL8.4
クライアント	CPU: Intel Xeon 5160 3.0GHz 2P8C Mem: 14GB	CPU: AMD Opteron 8220SE 2.8GHz (デュアルコア) × 4 Mem: 16GB
DBサーバ	CPU: AMD Opteron 8220SE 2.8GHz (デュアルコア) × 4 Mem: 16GB	CPU: Intel Xeon X5460 3.16GHz 2P8C Mem: 16GB
OS	RedHat Enterprise Linux 4.4	RedHat Enterprise Linux 5.3

【検証用APについて】

- 今回使用したAPは、更新処理の性能を測定するべく、「TPC-C」の概念をもとに弊社で作成した独自ツールであるため、他の結果と性能値の比較を行うことは出来ない。
- 本資料では、「TPC-C like」と呼称。
- 本資料での性能値はあくまで参考値。

- 長時間一定の負荷を掛け続けて「性能」の安定性を確認する、長時間運転安定性検証を実施。

■ 検証条件

- 検証時間: 48時間 (負荷の掛け具合で何ヶ月分に相当するかを調整)
- autovacuumのON/OFF双方の検証を行い、VACUUMによる性能回復度を確認する。
- autovacuum_vacuum_scale_factor = 0.01
- autovacuum_vacuum_threshold = 0 ※不要領域のパーセンテージだけで監視を行う。