NTT Open Source Software Center

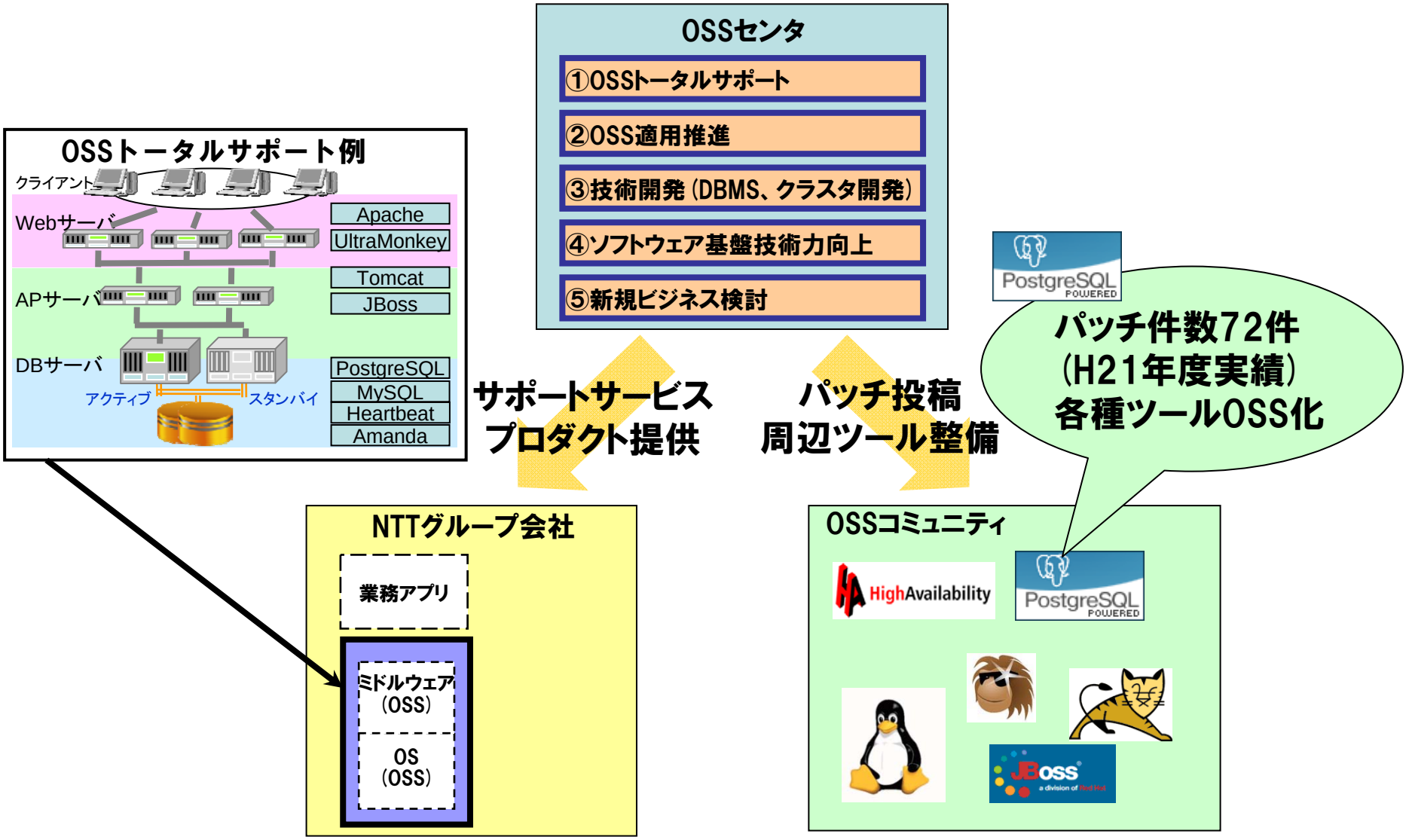
運用事例に学ぶ PostgreSQL

NTT OSSセンタ

勝俣智成
坂本昌彦
近藤光正

OSSセンタのご紹介 – 自己紹介を兼ねて –

NTT Open Source Software Center



アジェンダ

NTT Open Source Software Center

OSSセンタに寄せられたPostgreSQLの運用に関連する問合せの中から、
運用上有用と考える事例を紹介します。

I. VACUUM徹底理解
II. アーカイブログの扱い
III. チューニング

I. VACUUM 徹底理解

VACUUMはPostgreSQL特有のごみ掃除。自動VACUUM導入で
基本的には意識しなくなったとはいえ、問い合わせは多い。
事例を通じてVACUUMの詳細仕様を理解しよう。

VACUUMをめぐる問合せの分類

NTT Open Source Software Center

これまでのVACUUMの問合せを事象別にカテゴライズした。
本講演では、比較的問合せが多いものを紹介する。

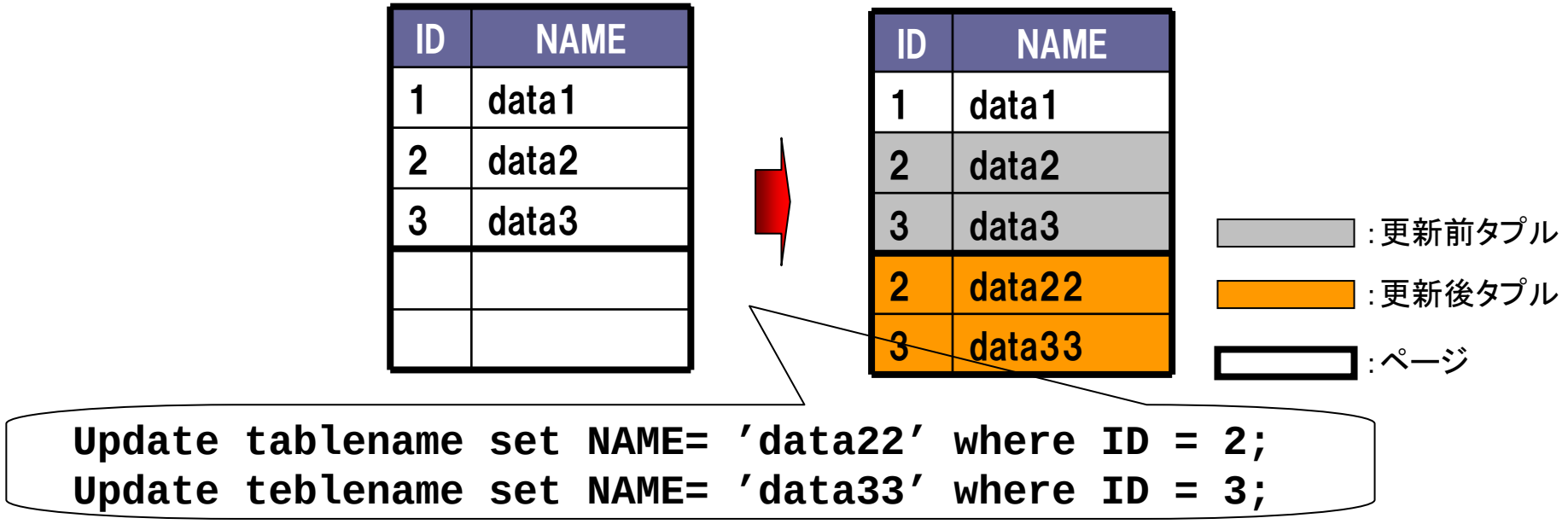
	問い合せ内容	原因・関連するPostgreSQLの仕様
ケース1	異常終了	自動VACUUMキャンセル機構
ケース2	終わらない(未応答)	PostgreSQL内部のロック
	終わらない(遅い)	背景負荷・FREEZE・ページ切詰
ケース3	効果なし	ロングトランザクション・FSM不足
	想定外の発動	XID周回防止機構
	発動しない	統計情報リセット・そもそも対象外

VACUUM基礎 1/2

NTT Open Source Software Center

追記型アーキテクチャ

- PostgreSQLは追記型
  - ・更新・削除をしても、元のデータが残る
  - ・誰からも参照されなくなると不要なデータとみなされる
  - ・DBサイズが大きくなっていく

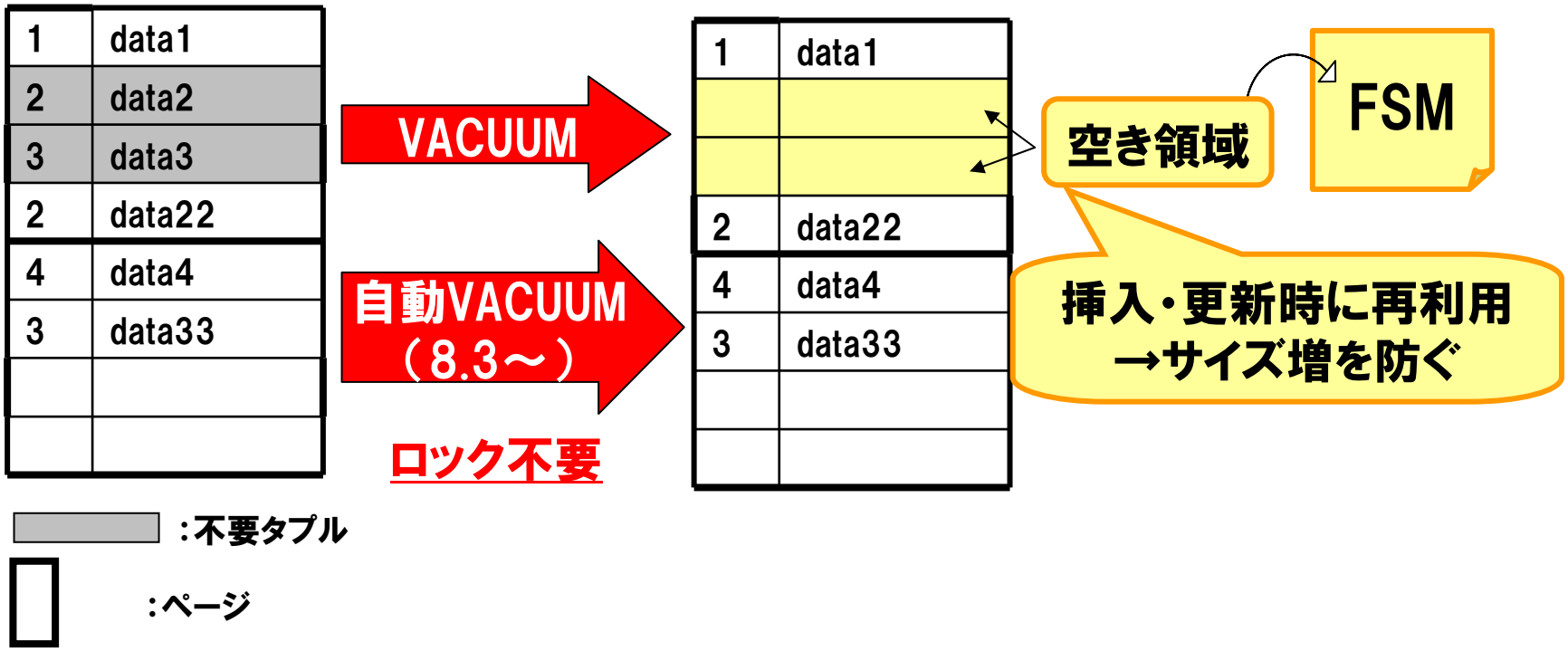


VACUUM基礎 2/2

NTT Open Source Software Center

追記型アーキテクチャと VACUUM

不要なデータの回収作業が必要。そのための処理が VACUUM



ケース1.自動VACUUMがキャンセル

NTT Open Source Software Center

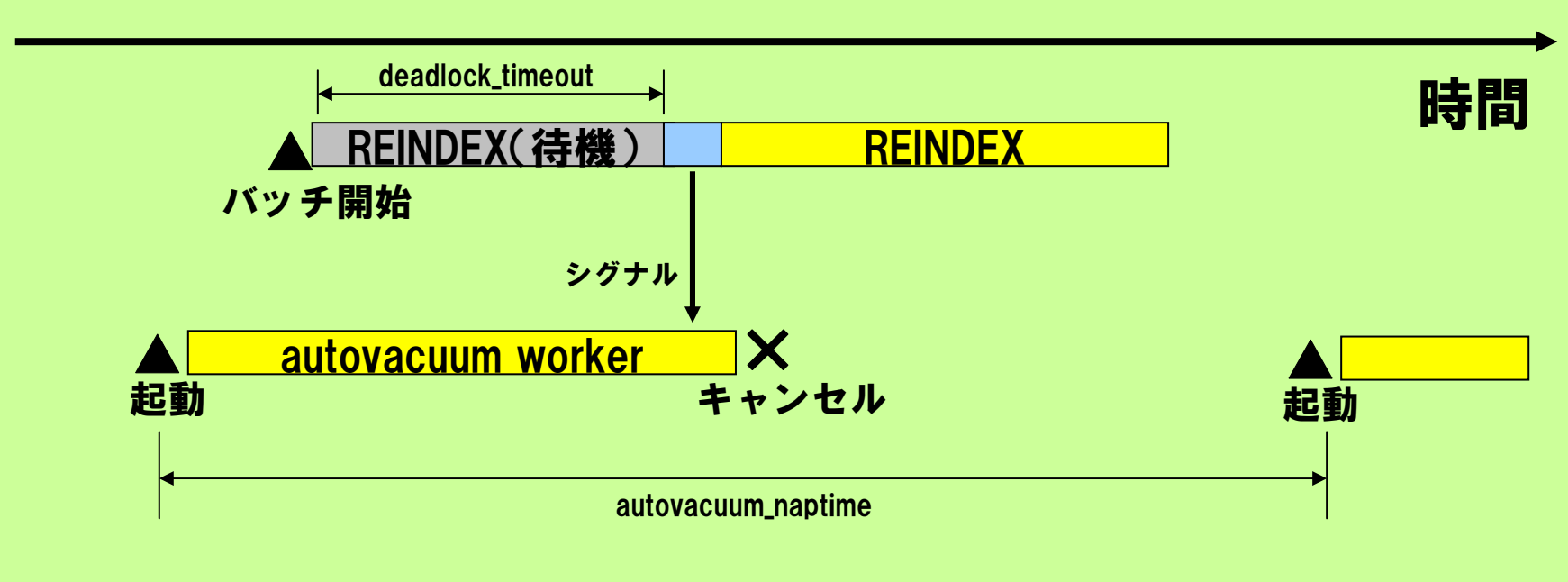
事象	ログにERRORレベルで自動VACUUMがキャンセルされた旨のログが出たが異常はないか？再実施はされるのか？ `2011-01-29 23:31:55 JST ERROR:  canceling autovacuum task`
原因/仕様	自動VACUUMは競合するSQL文によりキャンセルされる。
対策	自動VACUUMは再実施される。特に意識をしなくてもよい。

本事象が起こりやすい例：

- 夜間バッチ末尾でVACUUM コマンドを発行
- CREATE INDEX, REINDEX...などのDDLを発行
- 業務APが、通常業務の中で LOCK TABLE を発行

ケース1. 補足

自動VACUUMのキャンセル



REINDEXは、ロック取得待ち状態となり、一定時間後にデッドロック判定を行う。その中で、ロック待ち相手が自動VACUUMである場合にのみworkerにシグナルを送る。workerはシグナルを受け取るとキャンセルされるが、一定時間後にlauncherによる再検査があり、必要に応じて、再度VACUUMが実施される。

ケース2. VACUUMが終わらない

NTT Open Source Software Center

事象	VACUUMを実施したが(もしくは自動VACUUMが実施されたが)、何時間たっても終わらない。何が原因か？
原因/仕様	VACUUMは一時的にページへの排他ロックを取得する。これを阻害する要因※があるとVACUUMはロック取得待ちのままとなってしまう。
対策	カーソルを利用しているロングトランザクションの閉じ忘れ防止。

※該当ページを参照しているbackendがいる場合(pinがたっている場合)

VACUUMがとまる例
（行ロック取得待ち）

```
セッション1:
  BEGIN;
  DELETE FROM tab1 WHERE id = 1;
セッション2:
  DELETE FROM tab1 WHERE id =1;★ロック取得待ち
セッション3:
  VACUUM;
```

VACUUMがとまる例
（カーソルの使用）

```
セッション1:
  BEGIN;
  DECLARE cur CURSOR FOR SELECT * from tab1;
  FETCH 1 FROM cur;
セッション2:
  VACUUM;
```

ケース2. 補足

NTT Open Source Software Center

VACUUM（自動VACUUM）が停止している際の確認方法

- pg_stat_activity/pg_locksでは検出できない

```
postgres=# select * from pg_stat_activity ;
-[ RECORD 1 ]-+------------------------------
datid         | 11511
datname       | postgres
procpid       | 8264
usesysid      | 10
usename       | postgres
current_query | vacuum;
waiting       | f
xact_start    | 2011-02-02 19:33:54.111046+09
query_start   | 2011-02-02 19:33:54.111046+09
backend_start | 2011-02-02 19:24:09.76549+09
client_addr   |
client_port   | -1
```

- psでのプロセスの状態コードを取得

```
止まっている時
postgres 8264  0.2 0.3 161540 20992 ? Ss 19:24 0:02 postgres:postgres postgres[local] VACUUM
通常時
postgres 9245 11.9 0.7 164112 43636 ? Ds 19:48 0:02 postgres:postgres postgres[local] VACUUM
postgres 9245 16.5 0.4 164112 28692 ? Rs 19:48 0:01 postgres:postgres postgres[local] VACUUM
```

- gdbでバックトレースを取得

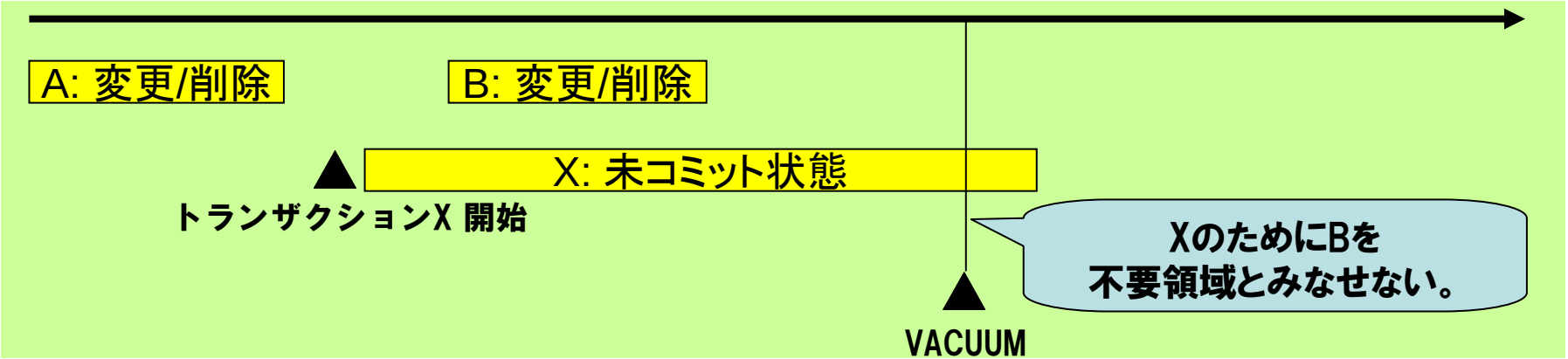
  LockBufferForCleanupでとまっていることを確認

ケース3. VACUUMは成功するけれど…

NTT Open Source Software Center

事象	VACUUMは定期的に行っているが（もしくは自動VACUUMをONにしているが）、テーブルサイズが大きくなって困っている。
原因/仕様	不要領域の判定は、VACUUM実施時点で動作している各トランザクションの可視性から判断される。ロングトランザクションがいるといつまでたっても不要領域とみなさない。
対策	ロングトランザクション・準備されたトランザクションが放置されていないかをチェック。存在時にはそれを解消。

不要領域の判定



ロングトランザクションの解消

```
postgres=# select pg_terminate_backend(9588);
 pg_terminate_backend
----------------------
 t
```

【参考】ケース4. VACUUMが遅い

NTT Open Source Software Center

事象	ある日実施したVACUUMだけ非常に時間がかかった。また大量のWALも吐かれているようだ。何の問題があるのだろうか？
原因/仕様	XID周回防止のため、一定期間変更されていない行の可視化処理（FREEZE処理）を行うため。
対策	log_min_freeze_ageを大きくして、明示的にVACUUM FREEZEできるためのメンテナンス時間を設け、その際に実施する。

事象	手動でVACUUMしたところ、自動VACUUMの何倍も時間がかかった。何が起きているのか？
原因/仕様	自動VACUUMでは、対象に付随したTOASTテーブルに対してはVACUUMを実施しない。一方、手動のVACUUMではTOASTテーブル・インデックスも対象となる。
対策	TOASTテーブルへのVACUUMは、適切な時に自動VACUUMが実施される。

事象	バッチで大量レコード削除後にVACUUMを実施しているが、毎回とても遅いように感じる。
原因/仕様	VACUUMはテーブル末尾に空のページがあるとファイルサイズを縮小する。その際に行われるページの切詰に伴う逆方向スキャンが遅い。
対策	大量レコード削除などが必要な場合、パーティショニング＋TRUNCATEで行うことを検討する。

Ⅱ.　アーカイブログの扱い

PostgreSQLのオンラインバックアップに関するトピックスをまとめる。
主にアーカイブログの扱いについて、事例を踏まえて解説する。

バックアップをめぐる問題の現状と分析

NTT Open Source Software Center

バックアップ・リストアに関する問い合わせをカテゴライズした。
本講演では、比較的問合せが多いものを紹介する。

	問い合せ内容	原因・関連するPostgreSQLの仕様
ケース1	エラーメッセージが出力	リストアの仕様
ケース2	オンラインバックアップから PostgreSQLを起動できない	必要なアーカイブログの不足
ケース3	アーカイブ領域のディスク容量が圧迫。どうすればよい？	アーカイブログ削除の未実施
ケース4	オンラインバックアップ取得の手順を教えてほしい	外部ツールの利用

オンラインバックアップ基礎 1/3

NTT Open Source Software Center

オンラインバックアップ/リストアの基礎

- WAL(トランザクションログ)によるクラッシュリカバリを応用
- 最新の状態だけでなく、任意の時点にリストアできる

オンライン中に取得したPGDATA(ベースバックアップ)とアーカイブログにより実現します。

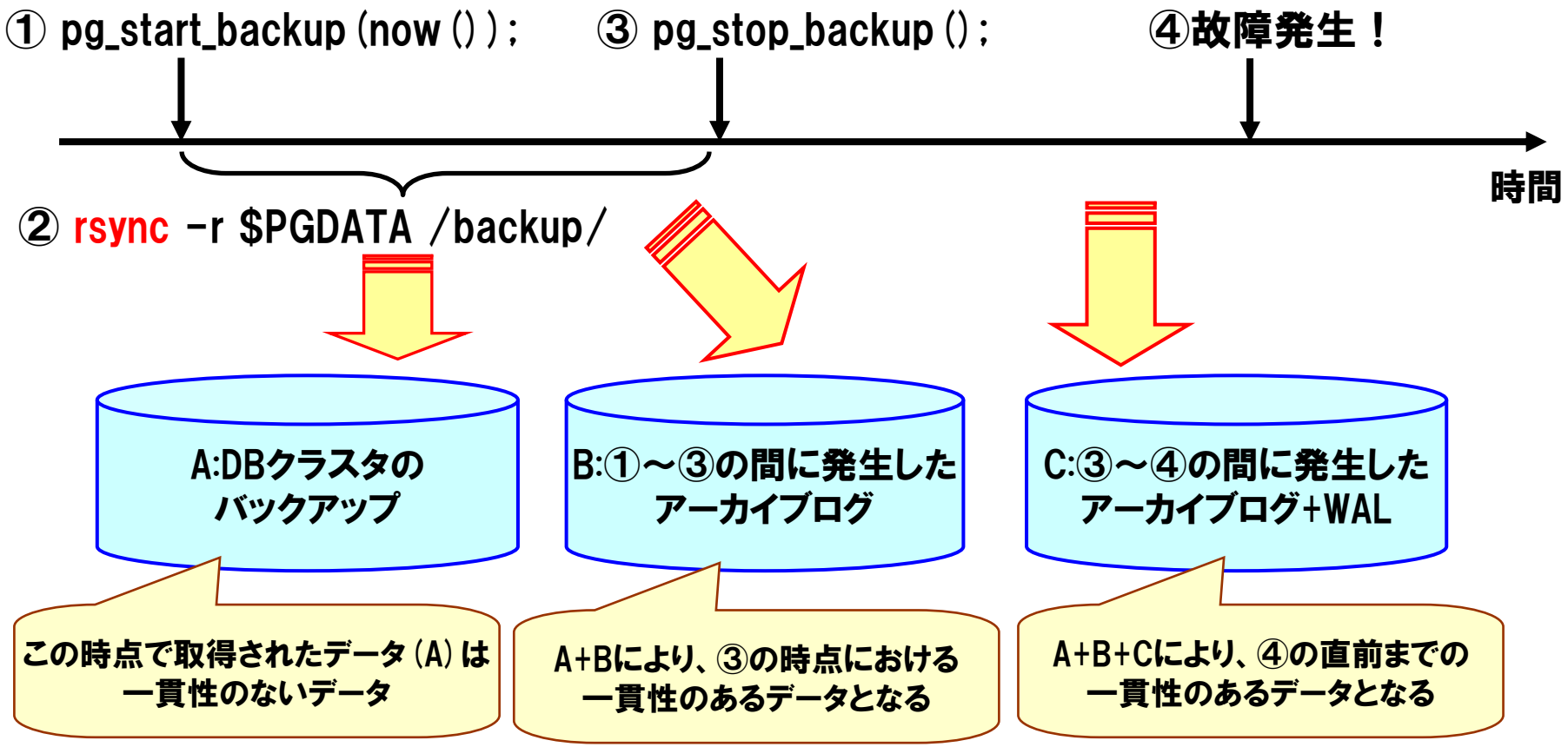
設定項目

- WALは定期的に再利用されるので、別途アーカイブログとして保存しておく。
  最低限必要な設定は以下の項目となる。
  - wal_level … PostgreSQL 9.0以降導入された項目
    「archive」か「hot_standby」に設定する
  - archive_mode …「on」に設定する
  - archive_command … アーカイブ領域への保存方法を記述する項目
    cpコマンドでOK
- リストアの設定は、recovery.confに設定する。
  - restore_command …「archive_command」に対応するコマンドを記述する

オンラインバックアップ基礎 2/3

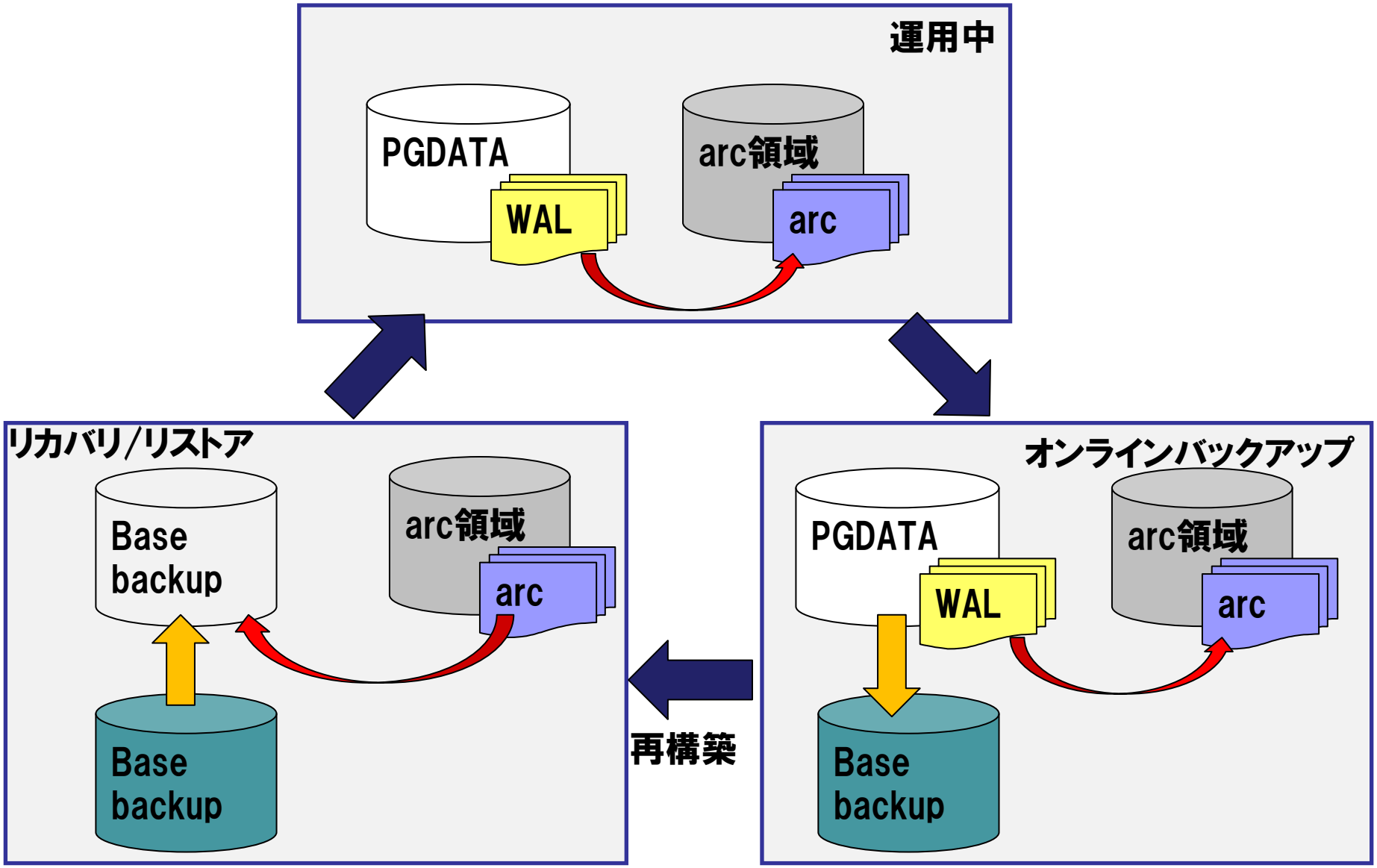
NTT Open Source Software Center

・オンラインバックアップ取得の流れ



オンラインバックアップ基礎 3/3

NTT Open Source Software Center



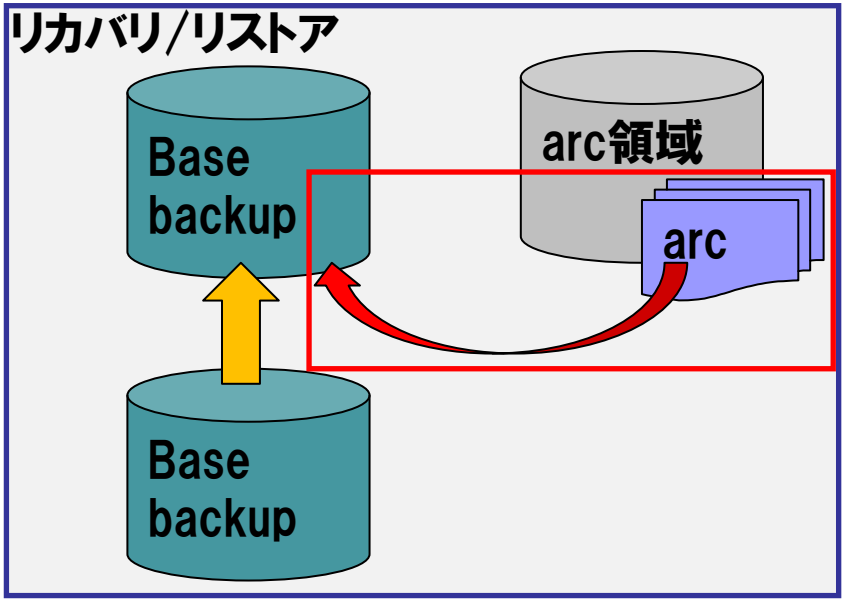
ケース1. リストア中にエラーメッセージ

NTT Open Source Software Center

事象	リストア時に以下のようなエラーメッセージが出力される。 リストアに失敗してるのか？ `No such file or directory`
原因/仕様	アーカイブログからのリストア時、コピーに失敗するまでファイルをコピーしようとする。また、本当に存在しないことを確かめるため、わざと失敗したりしている。
対策	無視しても問題ない。

処理の流れ

1. pg_controlを読んで、開始位置を決める
2. 開始位置を含むファイルをコピー
3. 成功しつづける限り、ファイルをコピー
4. 失敗したら、そこでコピーをやめる
5. 履歴ファイルを更新するため本当に現在の履歴が正しいか「ファイルがない」ことで確認
6. ないことを確認したら、ひとつ前の履歴ファイルをコピー＆追記更新して終了



ケース1. リストア中にエラーメッセージ

NTT Open Source Software Center

メッセージ例

```
LOG:  starting archive recovery
LOG:  restored log file "000000010000000000000003" from archive
LOG:  redo starts at 0/3000078
LOG:  consistent recovery state reached at 0/4000000
cp: cannot stat `/tmp/arc902/000000010000000000000004': No such file or directory
LOG:  could not open file "pg_xlog/000000010000000000000004“
                          (log file 0, segment 4): No such file or directory
LOG:  redo done at 0/30000A0
...
LOG:  archive recovery complete
LOG:  autovacuum launcher started
LOG:  database system is ready to accept connections
```

①

①:000000010000000000000004ファイルをアーカイブ領域をコピーしようとしてNG。
さらに、pg_xlogを探してみて見つからなかったらリカバリ処理を完了する。

ケース2. PostgreSQLが起動しない

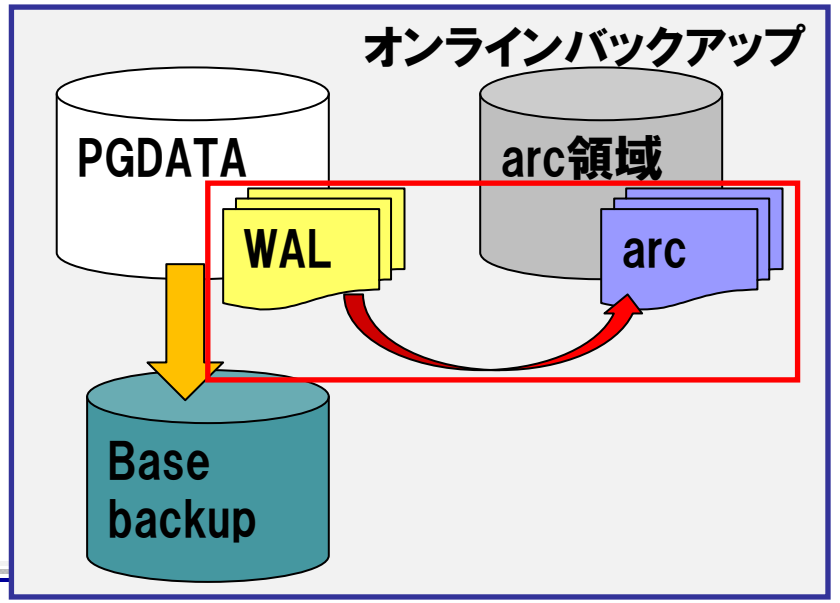
NTT Open Source Software Center

事象	オンラインバックアップで取得したバックアップを用いて、PostgreSQLを最新の状態にしようとしたが、起動できなかった。 `PANIC:  could not locate required checkpoint record`
原因/仕様	オンラインバックアップから最新の状態に復旧するには、ベースバックアップとアーカイブログと停止直前のWALが必要である。 アーカイブログとWALが歯抜けの状態になってはいけない。
対策	8.3以前では、pg_stop_backup () はアーカイブされるのを待たないため、別途アーカイブされたことを確認する必要がある。8.4以降では、確実にアーカイブされるので対策は不要。

PostgreSQL8.3のpg_stop_backup () の流れ

1. WALのスイッチを要求
2. backup_labelを読んで、開始位置取得
3. 履歴ファイルに、開始位置や終了位置、時刻等を書き出す

上記1の処理では、WALの切り替え要求のみ実施しているため、実際にアーカイブされるのはもっと後になる。pg_stop_backupがアーカイブより先に返却してしまうと必要なファイルが不足する。



ケース2. PostgreSQLが起動しない

NTT Open Source Software Center

- 必要なWALがアーカイブされたことを確認すること
  - スクリプトファイル等でバックアップを実施しているときは注意が必要。pg_stop_backupの直後にアーカイブ領域をコピーして、バックアップセットとしていないか確認すること。
- 8.3以前の場合には、以下のような確認を行う必要がある。
  - pg_xlogfile_name () 関数を用いて、アーカイブ領域をチェック

```
$ psql postgres -c "select pg_xlogfile_name(pg_stop_backup());"
     pg_xlogfile_name
--------------------------
 000000020000000000000002
(1 row)
$ psql postgres -c "select pg_xlogfile_name(pg_stop_backup());"
     pg_xlogfile_name
--------------------------
 000000020000000000000005
(1 row)
・・・
$ ls /tmp/arc902/000000020000000000000005*
/tmp/arc902/000000020000000000000005
$ ls /tmp/arc902/000000020000000000000002.*.backup
/tmp/arc902/000000020000000000000002.00000020.backup
```

pg_xlogfile_nameを利用し、どのファイルがアーカイブされればよいのかを把握する。

OSのコマンドで、アーカイブ領域に対象のファイルおよびバックアップ履歴ファイル(xxx.backup)が存在することを確認する。

ケース3. アーカイブ領域圧迫

NTT Open Source Software Center

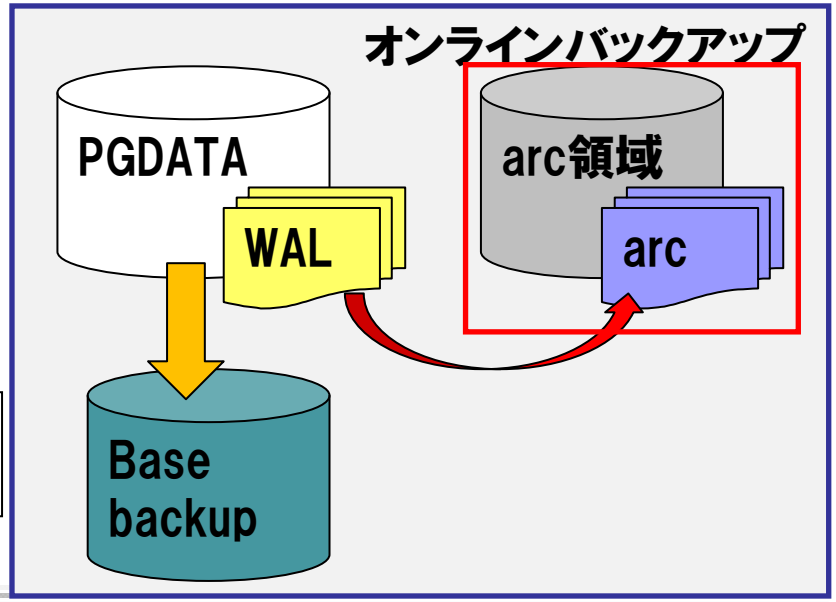
事象	アーカイブ領域のディスク使用量がどんどん増加している。どのファイルを削除すればよいか？
原因/仕様	PostgreSQL本体には、アーカイブログを削除する機能はない。
対策	どのファイルが不要かを確実に見分けて、定期的に不要なファイルを削除する。 不要なファイルの確認方法はいくつかあるので、要件に合わせて選択する。

アーカイブ領域がいっぱいになってしまうと、、、

- PostgreSQLはアーカイブに失敗しつづける
- このときPostgreSQLは、pg_xlog配下のWALを消さずに蓄えておいてくれる
- ただし、そのまま放置しつづけると、次第にpg_xlogの領域もいっぱいになり、PostgreSQLは利用できなくなる

```
PANIC:  could not write to file “xxxx”:
 No Space left on device
```

早めの対策を！



ケース3. アーカイブ領域圧迫

NTT Open Source Software Center

必要なファイルの確認手順

- ベースバックアップ配下にあるbackup_labelを確認し、「START WAL LOCATION:」より古いものを削除する
  - cat <base_backup>/backup_label
  - START WAL LOCATION: xx/xxx (file xxxxxxxxxxxxxxxxxxxx)
- 同様の情報は、アーカイブ領域にあるバックアップ履歴ファイル(xxxx.backup)にも記載される。ベースバックアップと対応するバックアップ履歴ファイルを確認し、古いものを削除する

いずれの場合も、必要なファイルを消さないよう

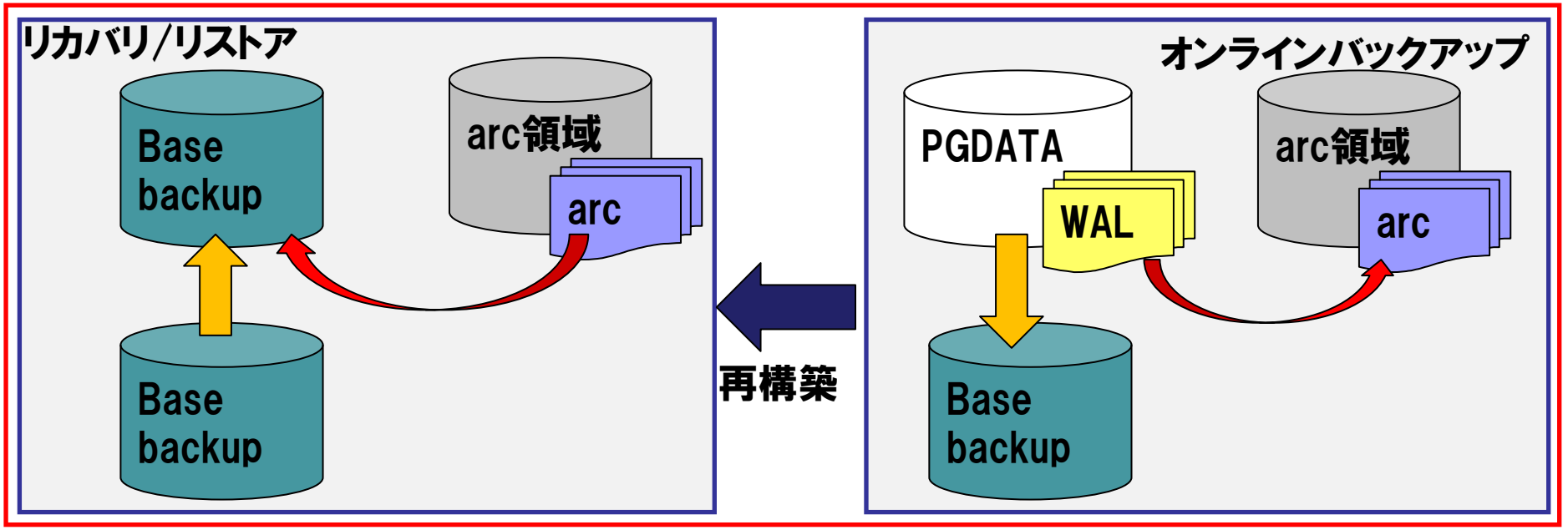
細心の注意を払って削除する。

- 対象のファイルは必要なファイルなので**それより前のファイルを削除**すること

ケース4. 具体的な手順

NTT Open Source Software Center

事象	オンラインバックアップ/リストアの手順が煩雑で分かりにくい。 具体的な手順を教えてほしい。
原因/仕様	クラッシュ・リカバリ/WALの機構を応用してバックアップ/リストアを実現しているので、全体的な手順は煩雑。
対策	注意点をきっちりおさえて、スクリプト等を作成する。 公開されているツールを利用するのもよい。



ケース4. 具体的な手順

NTT Open Source Software Center

- pg_rman
  - オンラインバックアップ/リストアの補助ツール
  - NTT OSSセンタで開発。GoogleCodeにて公開中
    http://code.google.com/p/pg-rman/
  - pg_dumpのように、いくつかのコマンドだけでオンラインバックアップ/リストアができるようになる
  - 再構築時に、テーブルスペースも自動的に再現可能

実行例:

```
$ pg_ctl start
$ pg_rman backup --backup-mode=full --with-serverlog
$ pg_rman validate
$ pg_ctl stop -m immediate
$ pg_rman restore
$ pg_ctl start
```

バックアップの実行と取得したファイルの検証

リストアも1コマンド

補足. 周辺ツールのご紹介

NTT Open Source Software Center

- pg_archivecleanup
  - ケース3の対策として有効なツール
  - 9.0以降、contribモジュールに付属
    http://www.postgresql.jp/document/current/html/pgarchivecleanup.html
  - スタンドアローンユーティリティとして機能させることが可能
  - また、recovery.confの「archive_cleanup_command」に設定することで、スタンバイ側で定期的にアーカイブ領域のクリーンアップが可能
    - この場合は、アーカイブ領域はレプリケーションのために利用されるだけで長期的な保管は意図してません。

実行例:

```
$ pg_archivecleanup -d <arc_dir> xxxx.yyyy.backup
```

バックアップ履歴ファイル(xxxx.yyyy.backup)を指定すると
<arc_dir>ディレクトリから、そのバックアップ履歴のバックアップを
リストアするのに必要なアーカイブログより前のファイルを全て削除する

Ⅲ.チューニング

内部挙動を理解しつつ、PostgreSQLをチューニングしてみよう！

チューニングをめぐる問題の現状と分析

NTT Open Source Software Center

これまでの問い合わせをカテゴライズした。
本講演では、比較的問合せが多いものを紹介する。

問い合せ内容	原因・関連するPostgreSQLの仕様
性能が出ない	推奨のチューニングパラメータを教えてください
バッファの効率的な利用について	一回目のアクセスが非常に遅い、2回目以降は早いが、対処策はないか？
バッチ処理について	バッチ処理に時間がかかる。良い対策はないか？

チューニングの基本

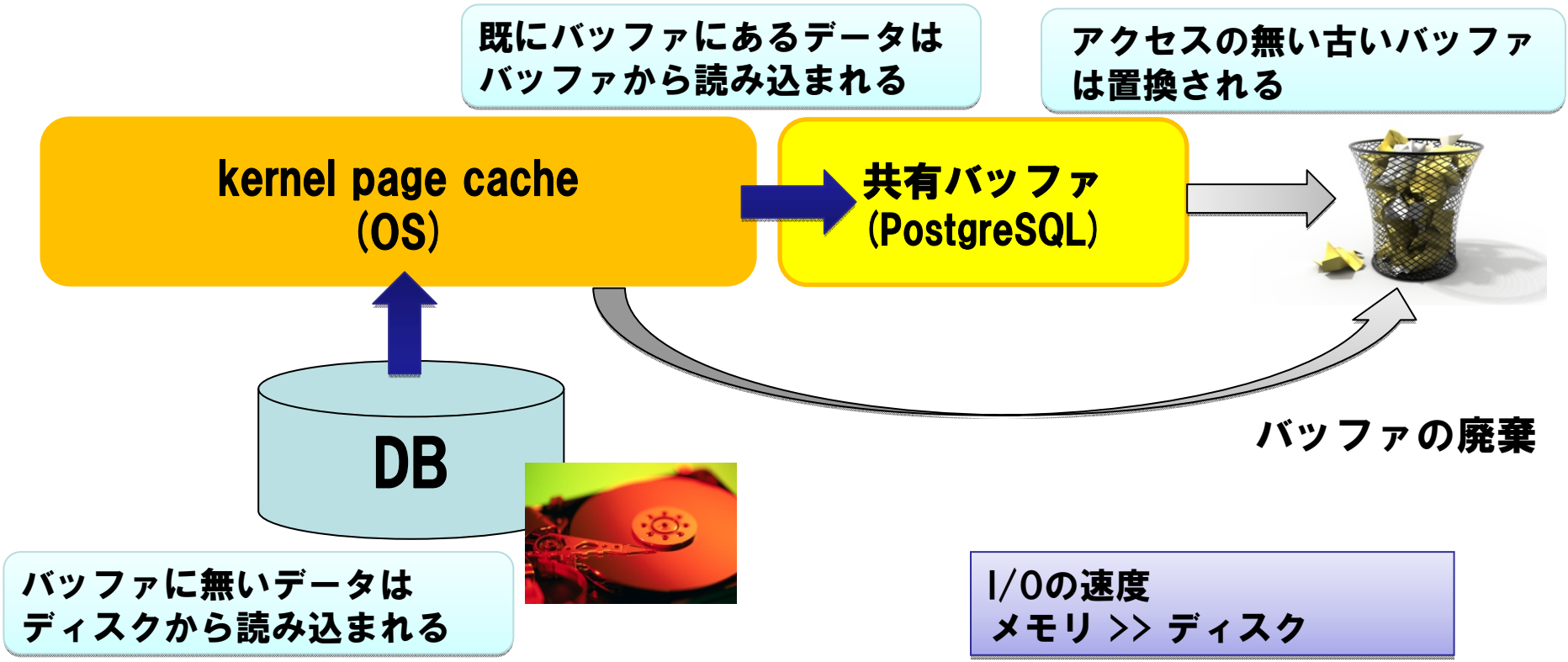
NTT Open Source Software Center

- shared_buffers ⇒ 搭載メモリの10%～20%
  - OSのページキャッシュと重複を少なくするため
- checkpoint_segments ⇒ 16～64
  - checkpointの回数を少なくすることで高速化
- wal_buffers ⇒ 4MB以上
  - トランザクション中のディスク書き込みを減らす
- effective_cache_size ⇒ 搭載メモリの50%程度
  - 適切な index scan が多くなる
  - random_page_costも2～3に設定すると更に良い
- work_mem
  - sortが多い場合は多めに設定する（要チューニング）
  - EXPLAIN ANALYZE で外部ソートが行われていないかをチェック

チューニングの基本 -バッファの仕組み-

NTT Open Source Software Center

・OSのバッファとPostgreSQLのバッファ



DBから読み込んだデータは、OSのキャッシュ領域を通じてPostgreSQLのバッファにも蓄積される　（バッファが重複する）

ケース１. 性能が出ない

NTT Open Source Software Center

事象	性能がでない。推奨するチューニングパラメータや高速化手法を教えて欲しい。
原因/仕様	初期のパラメータから変更していない。SQLに問題あり。 etc…
対策	先ほど紹介したチューニングパラメータを適用。 さらに、以下の内容を確認することで性能改善を図る。

- 適切にindexが張られているか？
  - EXPLAIN ANALYZE / EXPLAIN で実行計画を確認
  - 不要なindexが張られていないか確認
- HOTの活用
  - index 対象のカラムの更新はできるだけ避ける
- テーブル再編成を行う（CLUSTER, VACUUM FULL）
  - DB運用期間が長い場合は要チェック

ケース１．その他の主な対策

NTT Open Source Software Center

- Prepared Statementを利用する
  - SQLの構文解析結果や実行プランをキャッシュする機能
  - 典型的な OLTP や Web-DB では、CPU消費のうち25～50%がこのために消費されていた例も
  - JDBCやlibpqなどのクライアントIFを用いる場合は、そちらの機能から利用する

- 注意点
  - パーティショニングとの併用に難あり
    - 正確には、パーティションキーが変数化された場合に、constraint_exclusionが働かなくなります。返却される結果自体に矛盾はないが、候補テーブルの絞込みが遅れるため、かえって性能が劣化する場合あり。
  - 8.2以前での利用について
    - 8.3以降のPrepared StatementはANALYZE,VACUUM,DDLの変更の度に、実行プランを再作成するが、8.2以前では最初に生成した実行プランを使い続けるため、悪影響を及ぼす可能性あり。

ケース２.１回目のSQL処理だけ非常に遅い

NTT Open Source Software Center

事象	2回目のSQL処理は1秒以内だが、１回目のSQL処理が30秒と非常に遅い。サービス開始後に特に問題が出ている。
原因/仕様	DBが非常に大きかったこと、また、夜間のバックアップ処理とPGの再起動によりバッファがリフレッシュされていた。
対策	再起動後に頻繁に参照されるテーブルとインデックスをバッファに載せる。

- 特定のテーブルのみを事前にバッファに載せる※
  - SELECT * FROM テーブル名;
    - （注）Seq Scanの場合はインデックスはバッファに載りません。
- インデックスのみをバッファに載せる
  - contribのpgstattupleを使う
    - 例）SELECT * FROM pgstatindex（‘インデックス名’）
  - インデックスサイズ大で搭載メモリが少ない場合に特に有効

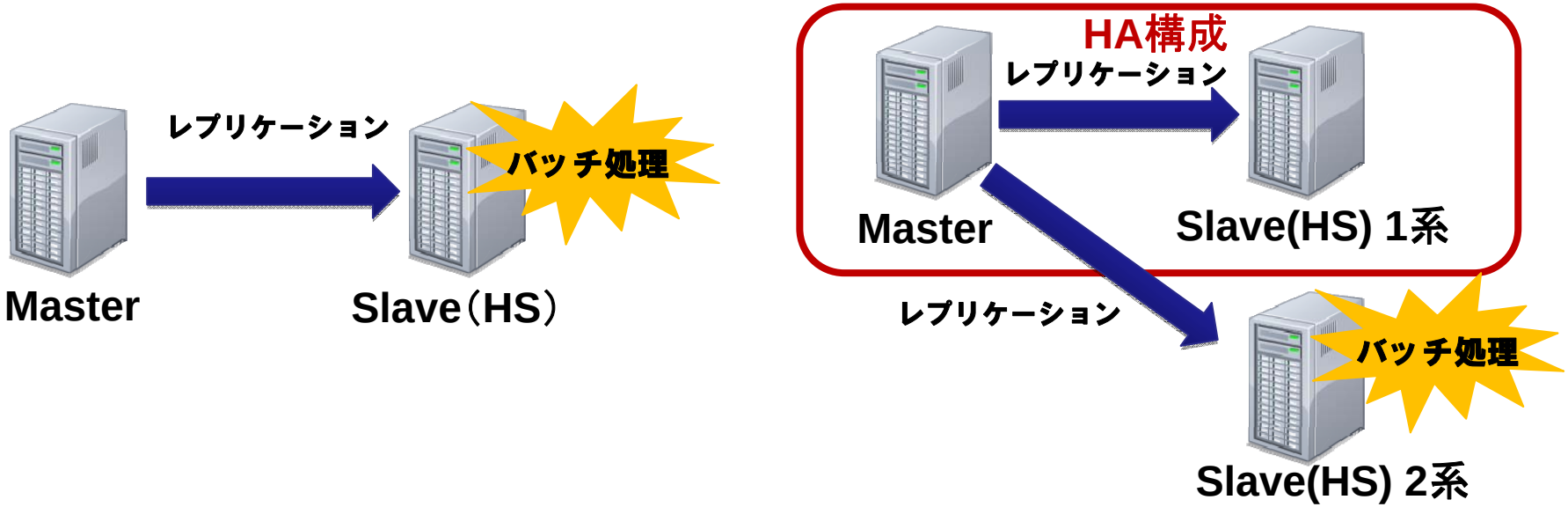
OSの

※PostgreSQLの共有バッファには全件は載りません。特定のテーブルデータで共有バッファが占有されるのを防ぐため、PostgreSQLがそれを防止します。

ケース３：バッチ処理を高速化したい

NTT Open Source Software Center

事象	夜間に現用系にてバッチ処理を実行しているが、処理が朝まで続きそう。このままではサービスへの影響がでそうだ。良い対策はないか？
原因/仕様	サービス開始後に予想以上のデータ量増加となり、バッチ処理の処理量が多くなった
対策	ケース１で紹介した内容の他にも、 9.0で追加のホットスタンバイ(HS)機能を利用する



HA構成の場合はマルチスレーブ構成にすると更に安全です