PostgreSQL
v.s.
大規模OLTP

2019年 4月 19日
OSSコンソーシアム データベース部会セミナー
SRA OSS, Inc. 日本支社 高塚 遥

講演者 と 本講演のテーマ

【講演者】

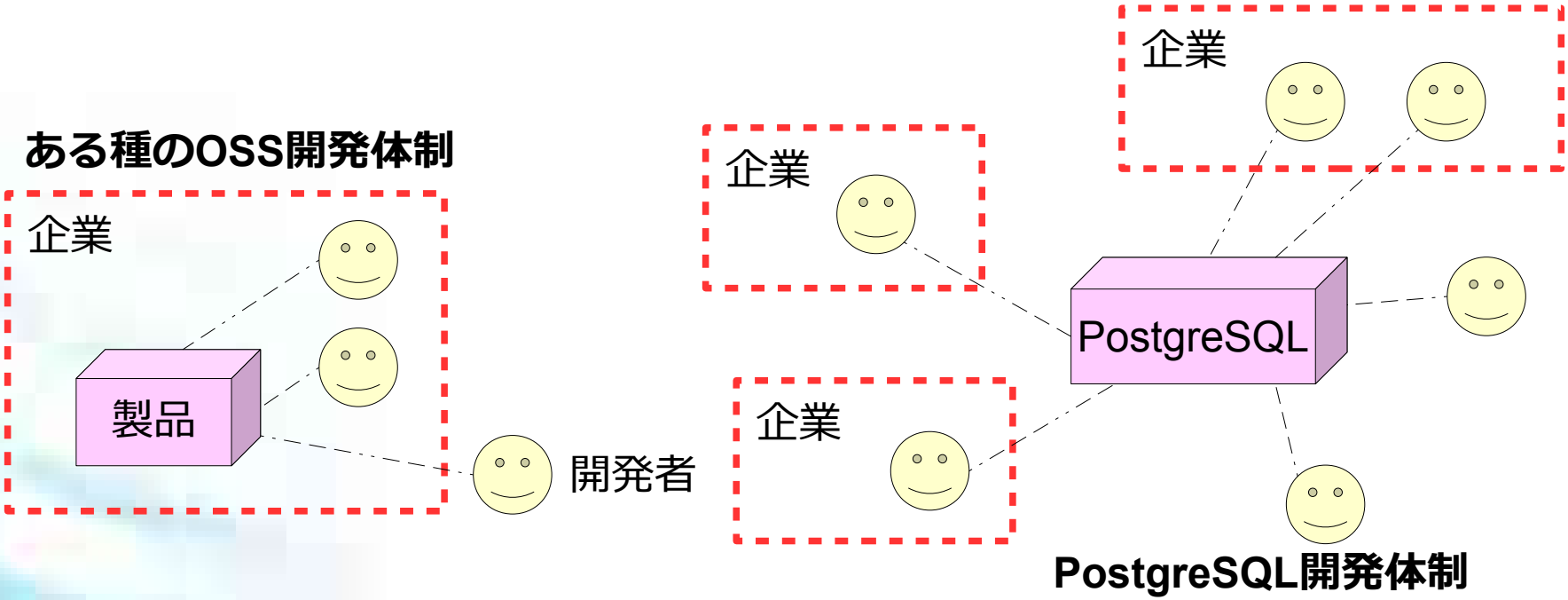
名前　　高塚 遥
所属　　SRA OSS, Inc. 日本支社
略歴　　PostgreSQLのセミナ講師、サポート、コンサルティング等に従事。
2016年より日本PostgreSQLユーザ会理事長を務める。
著書に「これから始める PostgreSQL 入門」（技術評論社）。

【TOC】

- PostgreSQL とは
- トランザクション処理のおさらい
- 大規模OLTPとの戦い
  - 今日のIT環境と PostgreSQLアーキテクチャ
  - スケールアップでの対応
  - スケールアウトでの対応

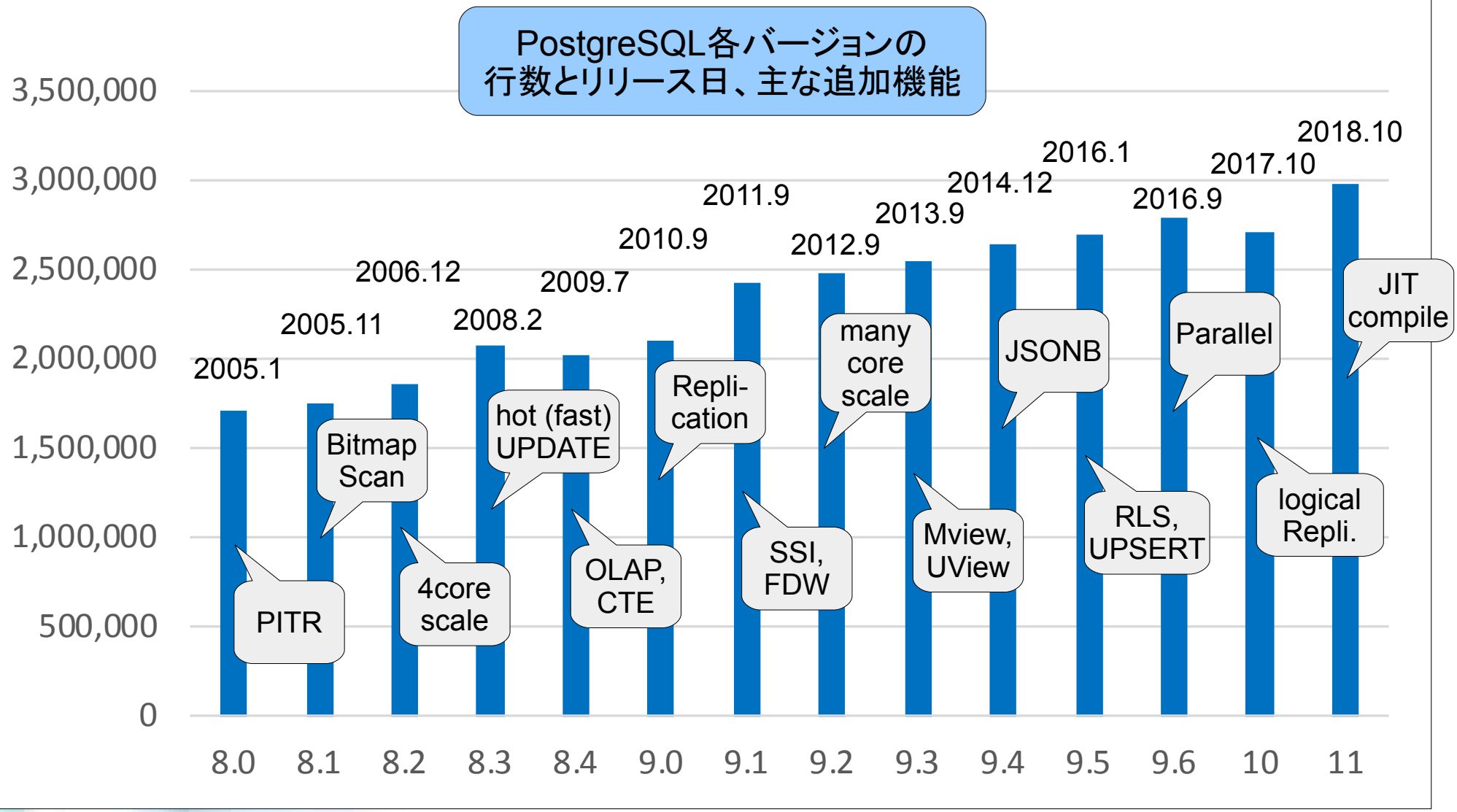
PostgreSQLとは (1)

- 多機能、高性能、かつオープンソースの
  リレーショナルデータベース管理システム（RDBMS）
- INGRES('70)、POSTGRES('80) 由来の歴史
- BSD型ライセンス
- 特定オーナー企業が無い



PostgreSQLとは（2）

- 安定的で管理されたリリース
- 機能面で着実に商用製品にキャッチアップ



トランザクション処理のおさらい (1)

- データベース利用法
  - OLTP（オンライントランザクション処理）
    - 銀行業務、各種決済、市場取引、電子商取引、在庫管理、予約予定管理
      （+ オンラインゲーム）
    - 使用者の実際の取引を確定させ、取引の事実を記録する
    - データ更新が頻繁に発生する
    - 同時並行で多数の利用者
  - OLAP（オンライン分析処理）
    - 分析用途
    - 基本的に参照のみ
    - データ投入は規則的・計画的 （たいてい OLTPデータベースから持ってくる）
    - 少数の利用者 （「センサーデータの並行大量投入」はある）
  - その他
    - CMS、ディレクトリ、資産管理、全文検索
      （「分析」ではないけれども参照中心）
    - SNS
      （データ更新は多いけれども、実際の取引とリンクしない、不整合に寛容）

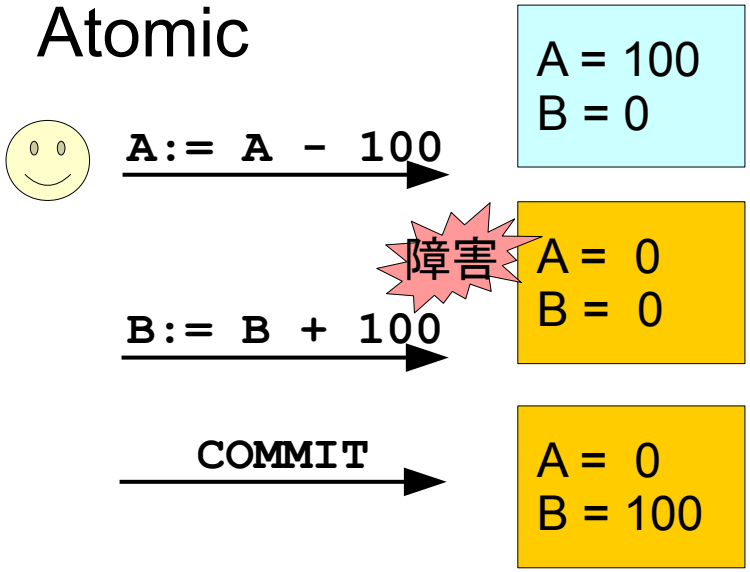
トランザクション処理のおさらい (2)

- トランザクション処理を行うシステムに要求されること
  ⇒ フルスペックな RDBMS が提供するもの

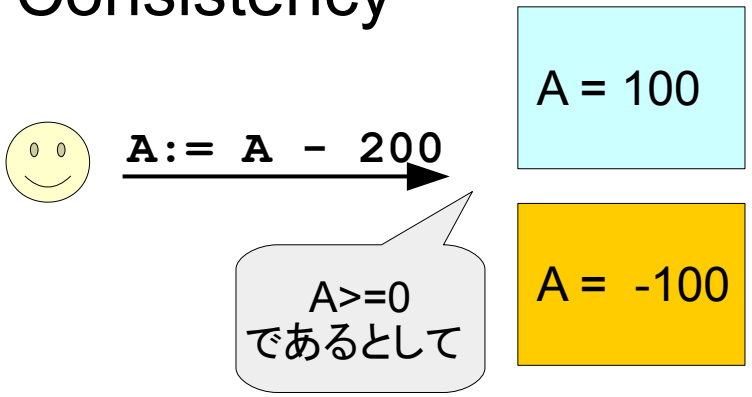
- ACID
  - Atomic 原子性
    - 一連の操作の結果が必ず「全部成功」か「全部失敗」のいずれかにできる
    - START TRANSACTION / COMMIT / ROLLBACK コマンド
  - Consistency 一貫性
    - データが正しい状態を保つ、見せる （「正しさ」は 論理設計する使用者が規定）
    - CONSTRAINT（制約）機能 、A と I、D の完備による不整合防止
  - Isolation 独立性
    - （並行する）他のトランザクションに影響されずに処理が実行できる
    - TRANSACTION ISOLATION LEVEL 対応、（明示、暗黙の）LOCK 機能
  - Durability 永続性
    - 確定したトランザクション結果が障害で失われない
    - 永続メディアへの書き込み、CHECKPOINT + WAL（Write Ahead Log）

トランザクション処理のおさらい (2-1)

- Atomic



- Consistency

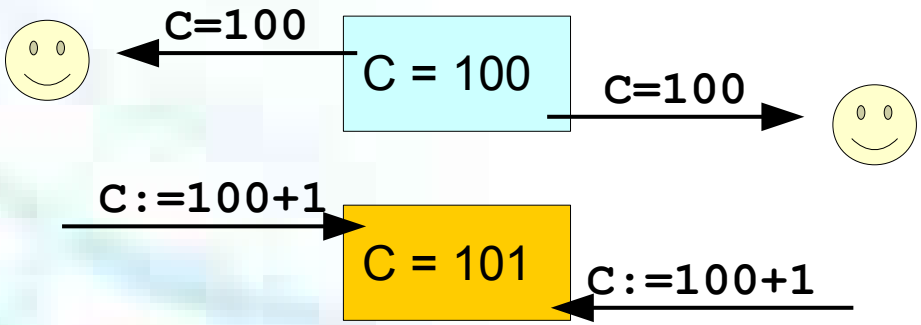


※ 購入履歴が追加されている
v.s.
出庫指示が追加されていない

等も Consistency違反。

非同期レプリケーションして
古いデータが見えてしまう等で
違反が生じる。

- Isolation



トランザクション処理のおさらい (3)

本質論では無い
現実の実装を区分

- SQL標準の TRANSACTION ISOLATION LEVEL

隔離レベル	発生する隔離失敗の動作	備考(PostgreSQLでの実装など)
READ UNCOMMITTED	Dirty Read（未コミットの変更内容の参照）	このモードは無い。READ COMMITTED と同じになる。
READ COMMITTED	Non-Repeatable Read（他TXのコミットにより同データを再度読むと異なる値）	デフォルトの隔離レベル。 多くのアプリがこのレベルを使っている。
REPEATABLE READ	Phantom Read（他TXのコミットで同条件で再度検索すると異なる結果）	SI (Snapshot Isolation) で実装。 よって実際には Phantom Read も起きない。
SERIALIZABLE	上記が発生しない。 Write Skew Anomaly（参照と書き込みの組み合わせで生じる異常） も発生しない。	SSI (Serializable SI)で実装。 暗黙の述語ロック機構が働く。 なお Oracle 実装は単なる SI。

- 高い隔離レベルでは、しばしば直列化失敗エラーが起きる
  - アプリケーション側で トランザクションのやり直しを行えば良い
    （DBアクセスフレームワークに標準で対応機能がほしい・・・）

トランザクション処理のおさらい (4)

- MVCC (MultiVersion Concurrency Control）による実現

物理格納データ:

テーブルのデータ

xmax	xmin	value
104	100	id=1 v='A'
	100	id=2 v='B'
108	102	id=3 v='C'
	104	id=1 v='A2'
	106	id=4 v='D'

コミットログ

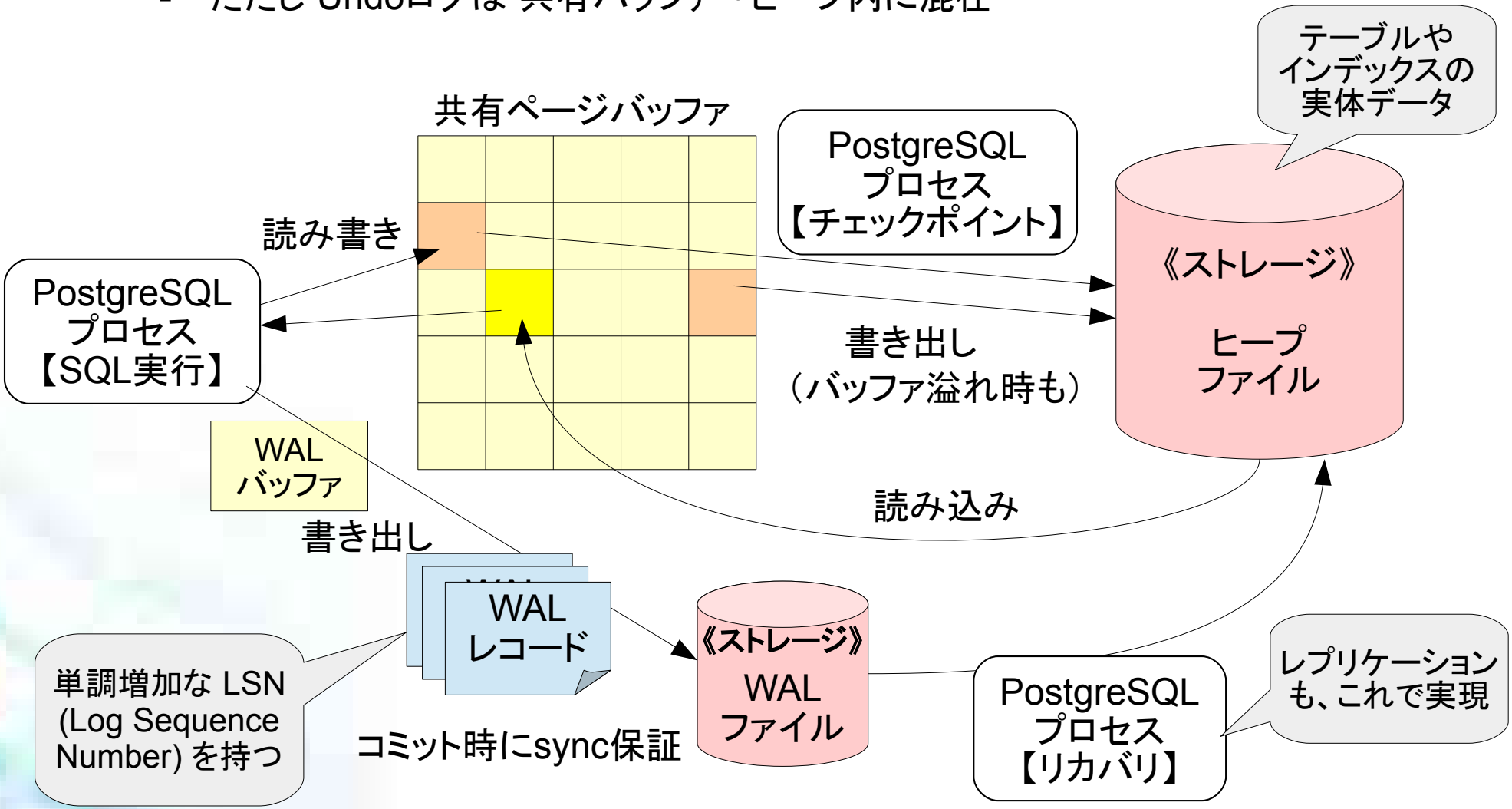
xid	status
100	コミット済
101	アボート済
102	コミット済
103	未コミット
:	:

後で Vacuum

```
～ 103番トランザクション ～
db=# SELECT * FROM t1;
 id | v
----+---
  1 | A
  2 | B
  3 | C
(3 rows)

～ 104番トランザクション ～
db=# UPDATE t1 SET v = 'A2'
       WHERE id = 1;

～ 106番トランザクション ～
db=# INSERT INTO t1 (id, v)
       VALUES (4, 'D');

～ 108番トランザクション ～
db=# DELETE FROM t1
       WHERE id = 3;
```

トランザクション処理のおさらい (5)

- チェックポイント / Write Ahead Log による Durability の実現
  - 伝統的な ARIESアーキテクチャの一種
    - ただし Undoログは 共有バッファ～ヒープ内に混在



大規模OLTPとの戦い (1)

- ますます高まる要求
  - IT業界で定番の枕詞

- 20年間のハードウェア成長の偏り
  - CPU　　クロックアップは飽和 / コア数、CPUキャッシュ、GPU
  - メモリ　　大容量化（含む低価格化）、 バスも高速化
  - ストレージ　　大容量化、SSD普及、 不揮発メモリ登場
  - ネットワーク　　高速化が継続中（～ 400GbE など）、とはいえ遅い

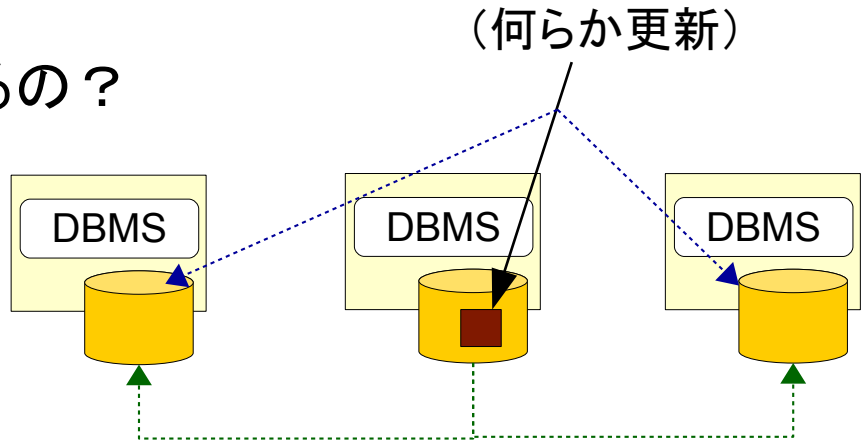
- 大規模OLTPでのボトルネックは？
  - 最速を求めると事実上オンメモリになる
    - ディスク格納指向のページを読み書きする無駄
  - 競合処理でコア数スケーラブルの限界
    - WAL書き出し周辺のボトルネック
    - それ以外にも同時実行での排他処理オーバヘッド

マルチマシンで
スケールアウト

多CPUコアへの
更なる適応

大規模OLTPとの戦い (2)

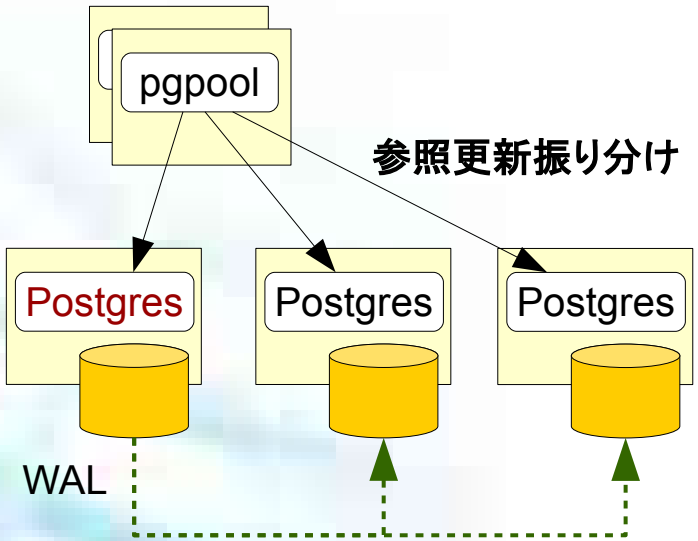
- データベースをスケールアウトするだって？
  - Webアプリケーションサーバなら簡単
  - DBはデータを持っている
    - レプリケーション？　シャーディング？
  - ACIDトランザクション処理はどうするの？
    - 2 Phase Commit
    - Paxos、Raft、etc・・・
    - 易しい話ではない
    - 分散用に設計されたDBMSでないと

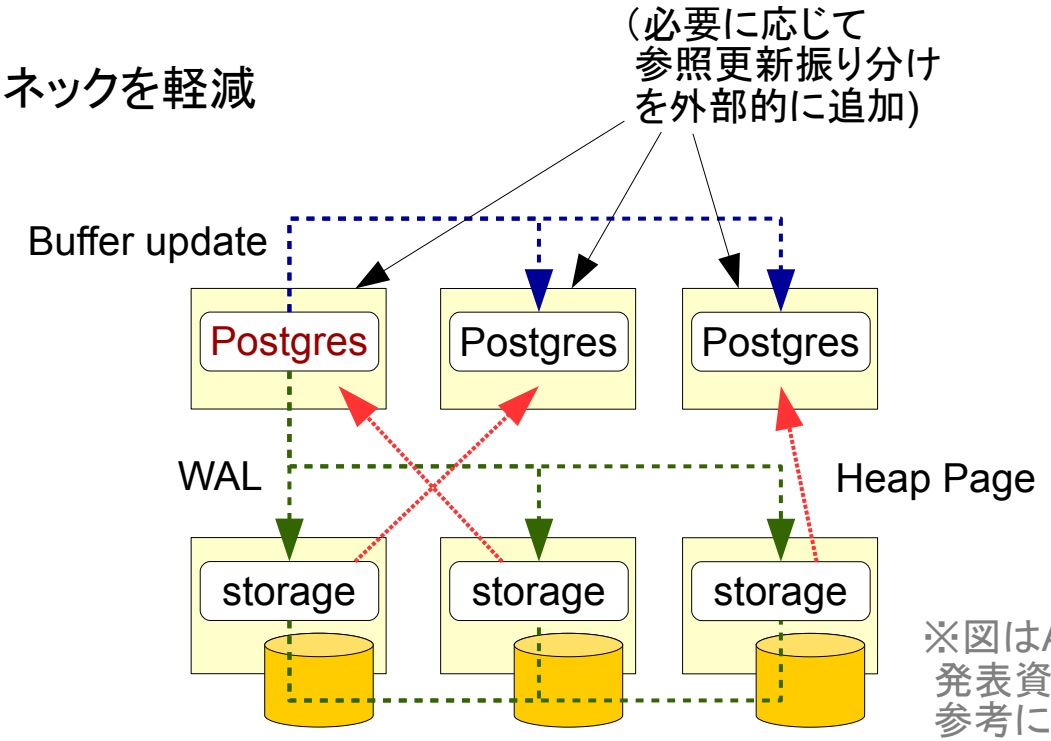


- 今ある PostgreSQLのクラスタ機能
  - 物理レプリケーション　インスタンス単位 WALを伝播
  - ロジカルレプリケーション　テーブル単位 WALに付加した行情報を伝播
  - 同期モード　非同期、write同期、apply同期、 クォーラム可
  - トポロジ　シングルマスタ＝マルチスレーブ、 カスケード可
  - 各種の外部プロジェクトで拡張されたクラスタソリューションが提供されている

大規模OLTPとの戦い (3)

- PostgreSQL の OLTP むけ高性能クラスタは？
  - レプリケーションと参照負荷分散
    - ストレージ同期保証off + クォーラム同期レプリケーション も可能
    - 更新処理への効果は限定的、またはマイナス
  - AWS Aurora PostgreSQL
    - WALレコードでストレージノードに書き込み
    - 信頼性を実現しつつ、
      WAL書き込み周辺のボトルネックを軽減

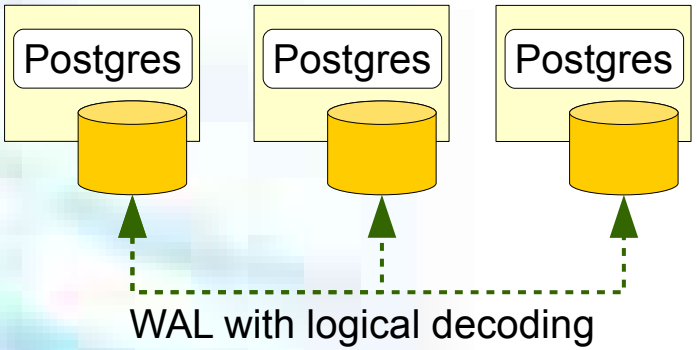


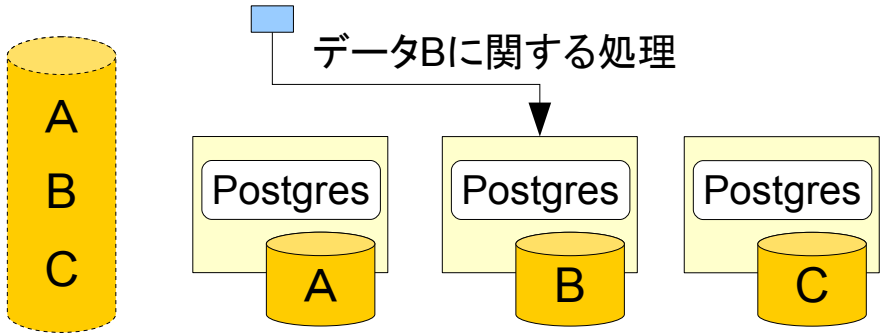


※図はAWS
発表資料を
参考に作成

大規模OLTPとの戦い (4)

- BDR（Bi-Directional Replication、マルチマスタ）クラスタ
  - 商用製品による拡張
  - ロジカルレプリケーションに基づく ／ 整合性保証は緩い
  - 書き込み性能向上目的に適しているわけではない
- 透過的なシャーディングクラスタソリューション
  - Postgres-XL ／ Citus ／ 殆どの製品は OLAPむけ
- アプリケーションレベルでのシャーディング／DB分割／KVS併用
  - 現アーキテクチャで対応可能な規模に分割する
    - 必要に応じて FDW（外部データラッパ）やテーブル単位ロジカルレプリケーションを活用
  - 一部のデータをスケーラブルな KVS に逃がす、あるいは、キャッシュとして使う
  - PostgreSQLでの大規模OLTP対応に現実的に使われているのはこれ





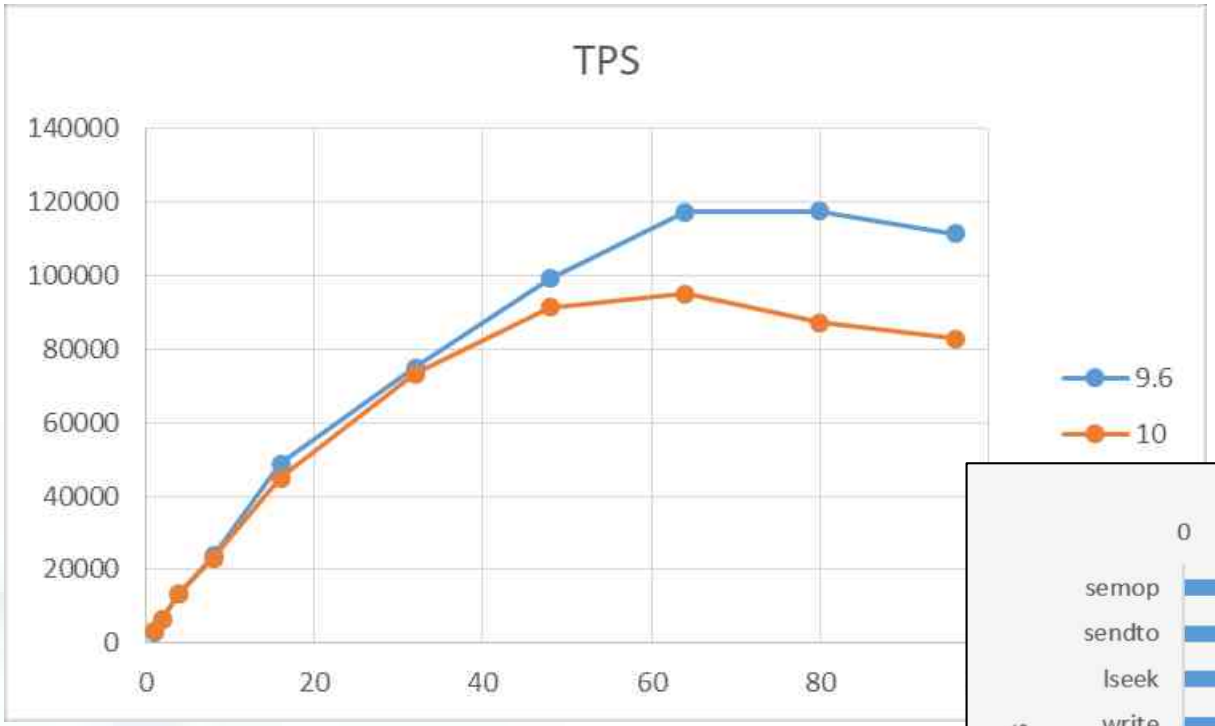
大規模OLTPとの戦い (5)

- PostgreSQLの透過的OLTPむけクラスタソリューションは無い
  - 今のところ

- （PostgreSQLでは無いが参考に）VoltDBアーキテクチャ
  - “The End of an Architectural Era” の Michael Stonebraker による
    - PostgreSQLの先祖 INGRES 開発者でもある
    - 「DBMS はバッファ制御や排他処理に CPU時間の多くを使っていて無駄」
  - オンメモリ
  - クォーラムレプリケーションで永続性保証
  - トランザクション実行を「本当に」直列化して排他処理のコードを排除
    - テーブルはパーティショニングかレプリケーションが必須
    - 個々のトランザクション実行には、複数CPUコアを使う
    - トランザクションはストアドプロシージャの形で与える

大規模OLTPとの戦い (6)

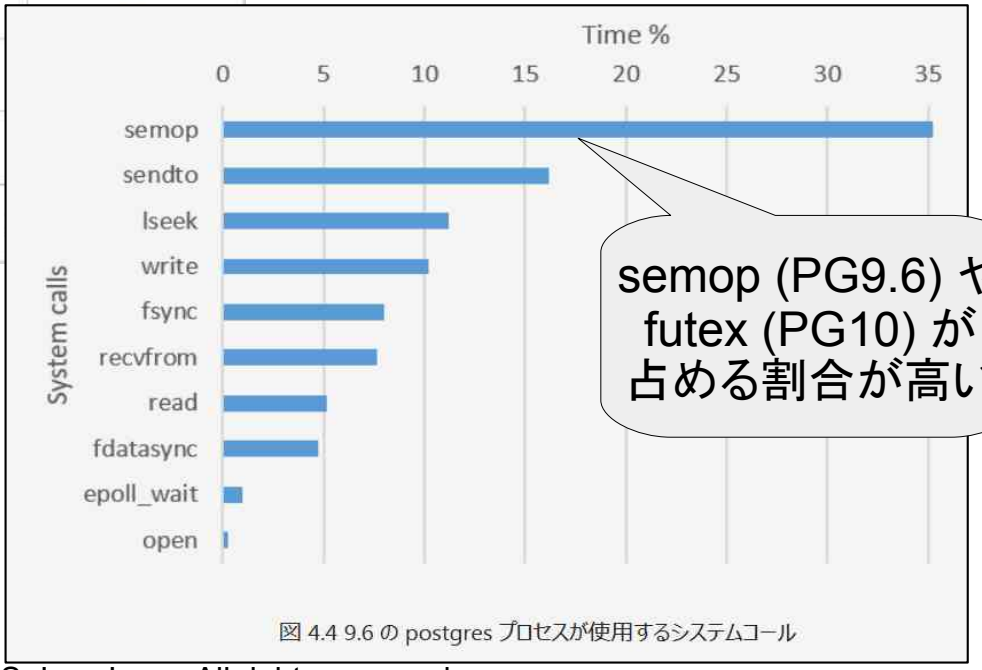
- 1台のDBサーバを多CPUコアにスケールアップ
  - PGECons pgbenchベンチマーク結果



CPU: 2p/32c
memory: 384GB

シナリオ:
index検索ランダム単一行の UPDATE
1行はメタデータを除く100byte程度
全データがメモリに載る

横軸はクライアント同時実行数



semop (PG9.6) や
futex (PG10) が
占める割合が高い

両グラフとも
PGECons「2017年度WG1活動報告書」
の『4.7. 検証結果(更新系)』より

このとき 10.x が
遅い件はマイナー
アップで修正された

大規模OLTPとの戦い (7)

- PostgreSQL の 多CPUコア～CPU律速 むけ改修
  - PostgreSQL 9.6～11 での改修
    - 共有メモリ上の構造体の分割 ／ 排他制御の範囲を小さく
    - 共有バッファのページ管理方式を変更 ／ スピンロックを回避
    - 各種構造体を小さく ／ CPUキャッシュに収まるように
    - コミットログのグルーピング更新
    - JITコンパイル … これはOLAPむけ

      → この種の改善は今後も続くであろう
  - PostgreSQL 12 での改修
    - ストレージエンジンが Pluggable になった
    - MySQLでは昔から
    - Aurora の成功例
    - 商用版ベンダ EnterpriseDB社による Oracleライクなエンジン開発

      → PostgreSQLを素晴らしくする競争が始まった！
- GPU ソリューション
  - PG-Strom … これも OLAP むけ

まとめ

- 伝統的な RDBMS としてトランザクション処理に十分な機能をもつ PostgreSQL

- 大規模OLTPに対するスケールアップ、スケールアウトの対応は現状では難しい
  - 新たなアーキテクチャ取り込みが期待される

オープンソースとともに

URL：http://www.sraoss.co.jp/
E-mail：sales@sraoss.co.jp
Tel：03-5979-2701