

# PostgreSQL V9.2 新機能のご紹介

PostgreSQL 9.2 最新動向セミナー(2) ※講演後公開版

2012-06-26 14:40~15:30

SRA OSS, Inc. 日本支社

高塚 遥 [harukat@sraoss.co.jp](mailto:harukat@sraoss.co.jp)

# PostgreSQL のこれまでと現在

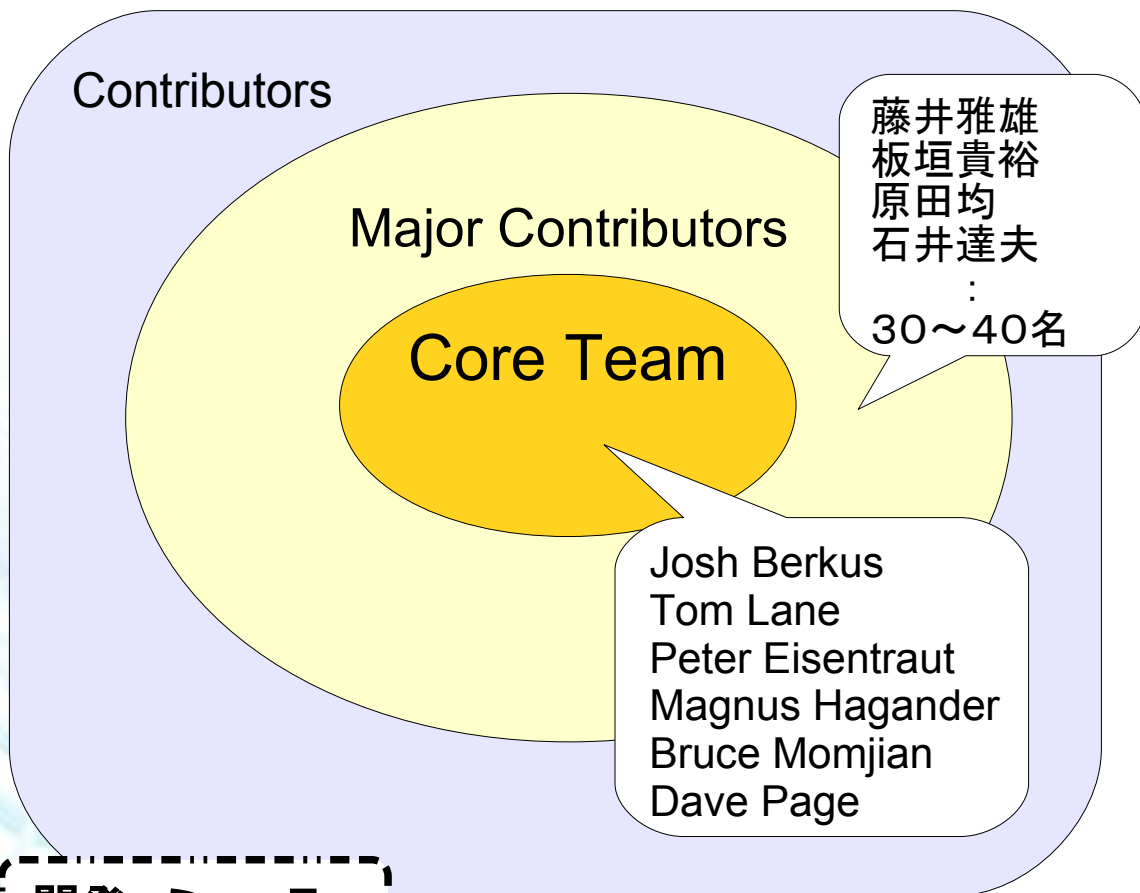
---

## 改めて・・・ PostgreSQL とは

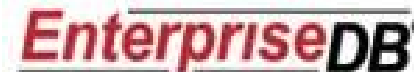
- 代表的なオープンソースRDBMS
- Ingres(1970～ UCB) を先祖に持つ
  - PostgreSQL 6.0 (1996 ～) から 15年以上の歴史
- BSDタイプのライセンスで配布
  - PostgreSQL Global Development Group と University of California が著作権を持つ
- ひとつのオーナー企業、オーナー個人を持たない
  - PostgreSQL開発に時間を割く技術者を提供している企業  
がいくつかある／その企業群も少しずつ変遷している

# PostgreSQL開発体制

支援企業



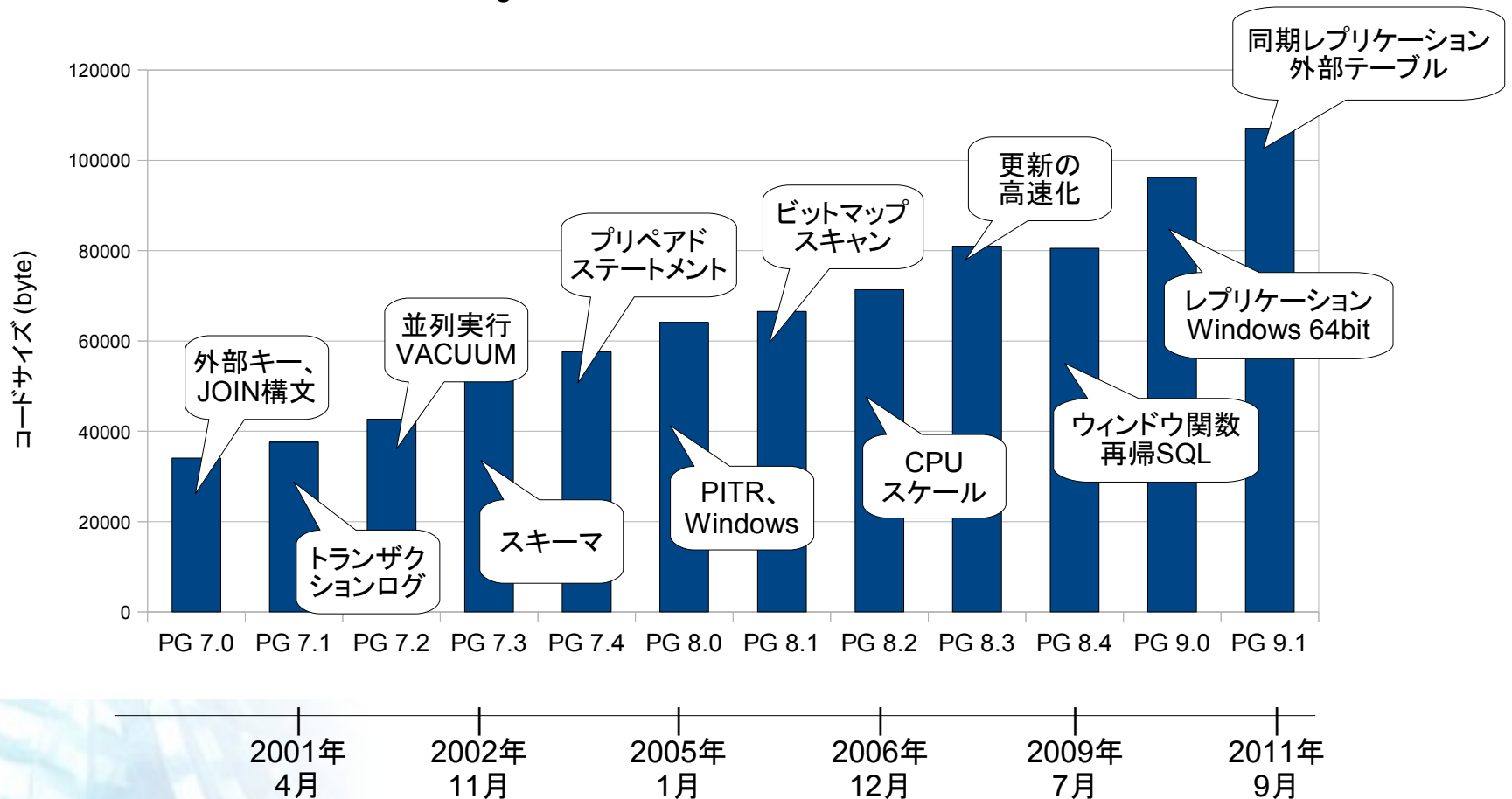
開発コミュニティ



etc...

# PostgreSQLの歩み

PostgreSQL のコードサイズとリリース



# PostgreSQL 9.2 の拡張

---

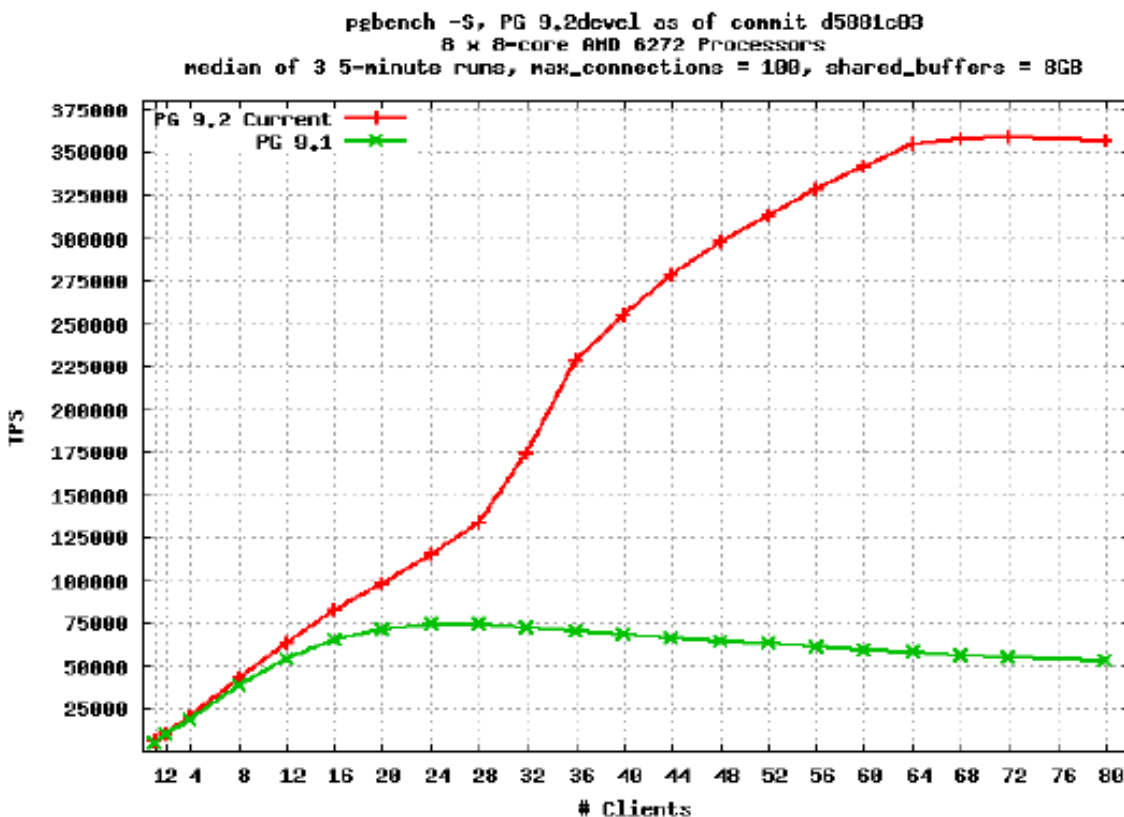
# CPUスケール対応

- PostgreSQL 9.2 でコア数32~64までスケール  
⇒ PostgreSQL 8.2 で改良 コア数 8~12 まで

64コアマシンで参照pgbench

※PGCon 2012  
EnterpriseDB社 Robert Haas の  
発表資料より引用

- Fast Pathロック
- 直列化部分を短時間に
- WALディスク同期を改善



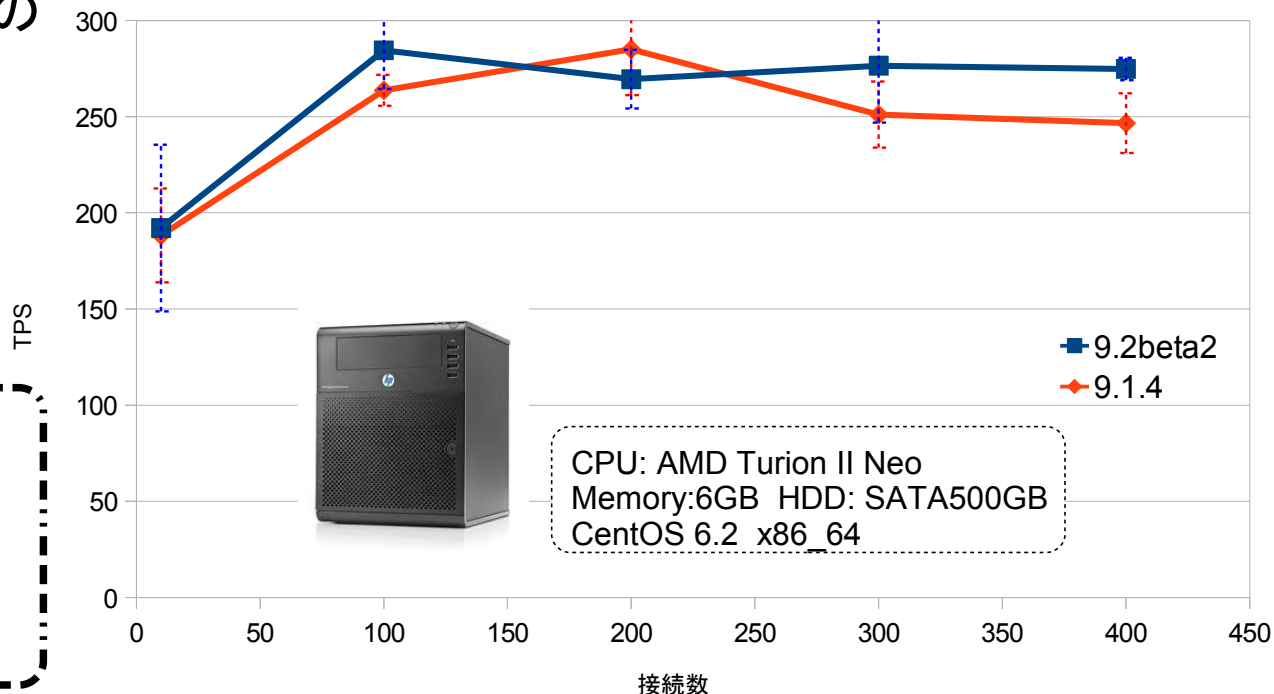
# 書き込みトランザクション改善

- 小規模性能サーバ(2コアマシン)
  - 標準 更新pgbenchでは差は僅か

多接続で 5~10%程度の改善がみられるのみ。

ストレージ律速になるとバージョン差が出ない。

pgbench 更新テスト (-s 10, -j 2)



CPU: AMD Turion II Neo  
Memory:6GB HDD: SATA500GB  
CentOS 6.2 x86\_64

- Fast Pathロック
- 直列化部分を短時間に
- WALディスク同期を改善



# COMMIT遅延の改善

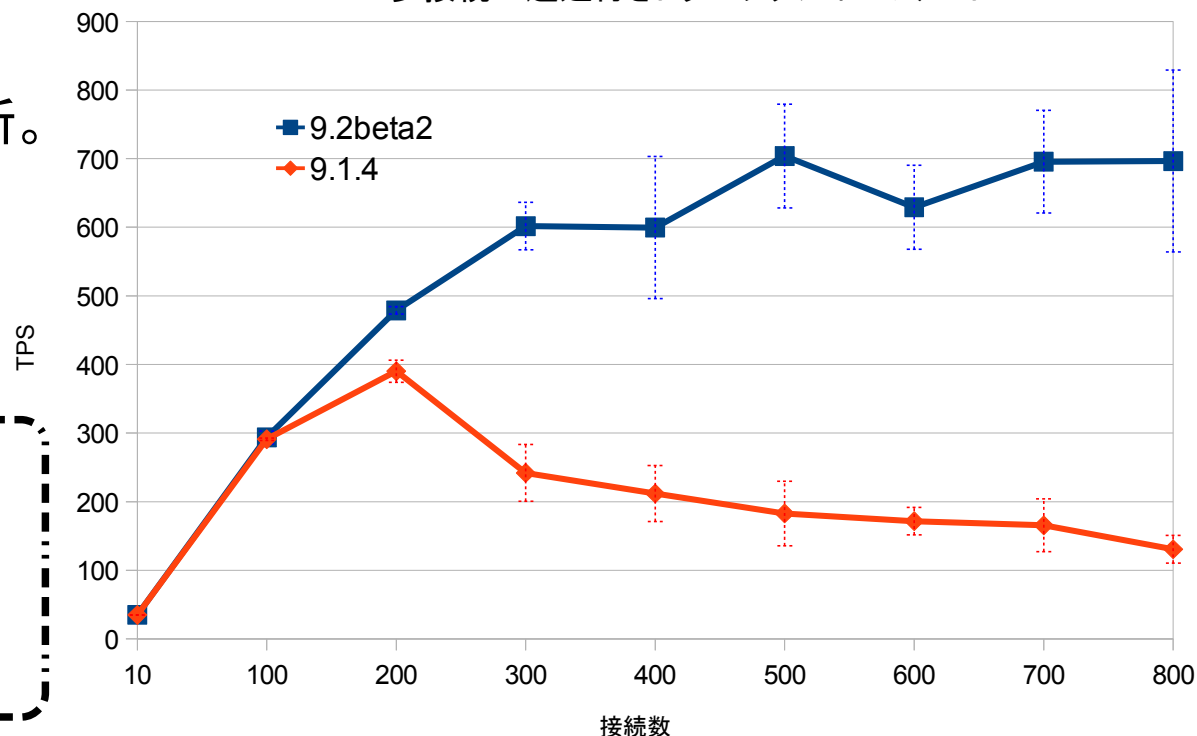
- 2コアマシンでもテスト内容によっては大幅性能向上

独自シナリオpgbench  
間に sleep を入れた更新。

従来 COMMIT遅延が  
目立ったものが解消  
されている。

- ・Fast Pathロック
- ・直列化部分を短時間に
- ・WALディスク同期を改善

多接続／遅延付きランザクションのテスト



# Index Only Scan (1)

- インデックスだけを読んで処理できる
  - 行バージョンがあるため難しかったが Visibility Map を参照して「可能なときには行う」ことで対応

```
=# EXPLAIN select min(i) from t;
```

QUERY PLAN

---

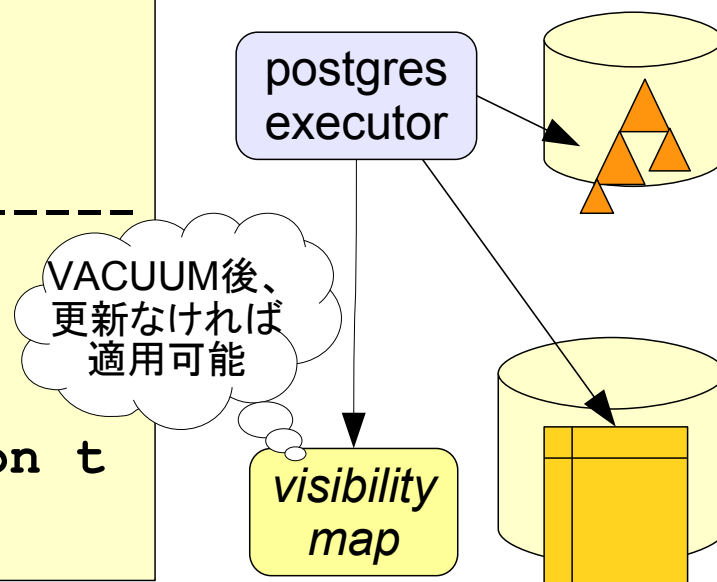
Result

```
InitPlan 1 (returns $0)
```

```
-> Limit
```

```
-> Index Only Scan using tix on t
```

```
Index Cond: (i IS NOT NULL)
```



## Index Only Scan (2)

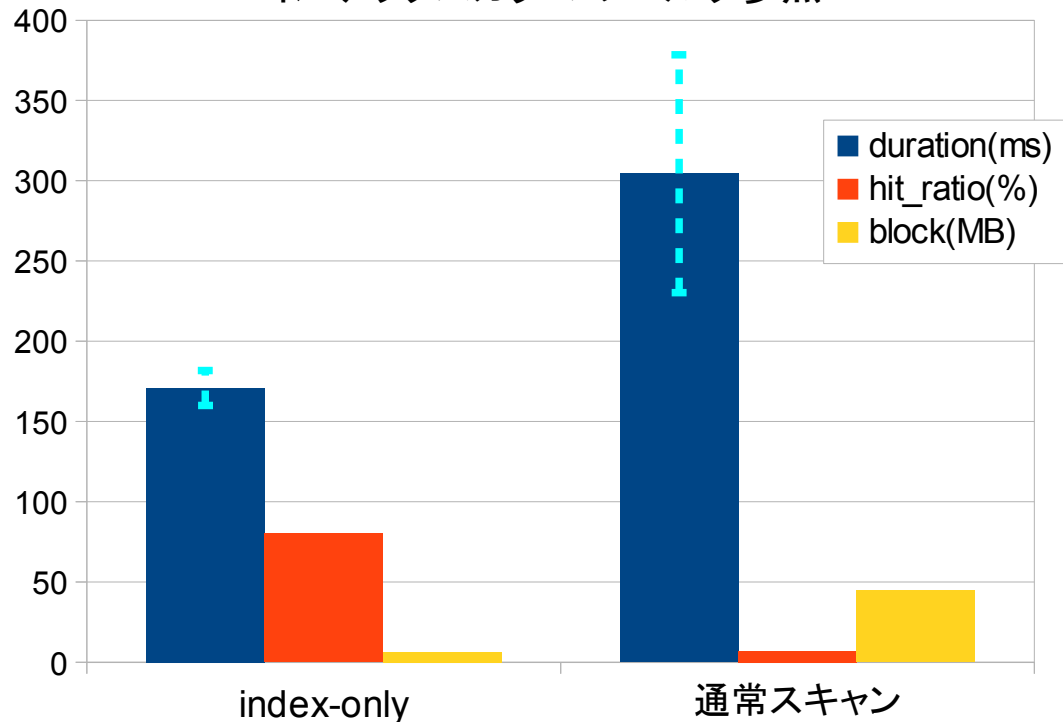
- 適用されれば大きな効果

- 読み込みブロック減
- バッファヒット率向上
- min, max, count などに有効

《全50万行から30万行を参照》

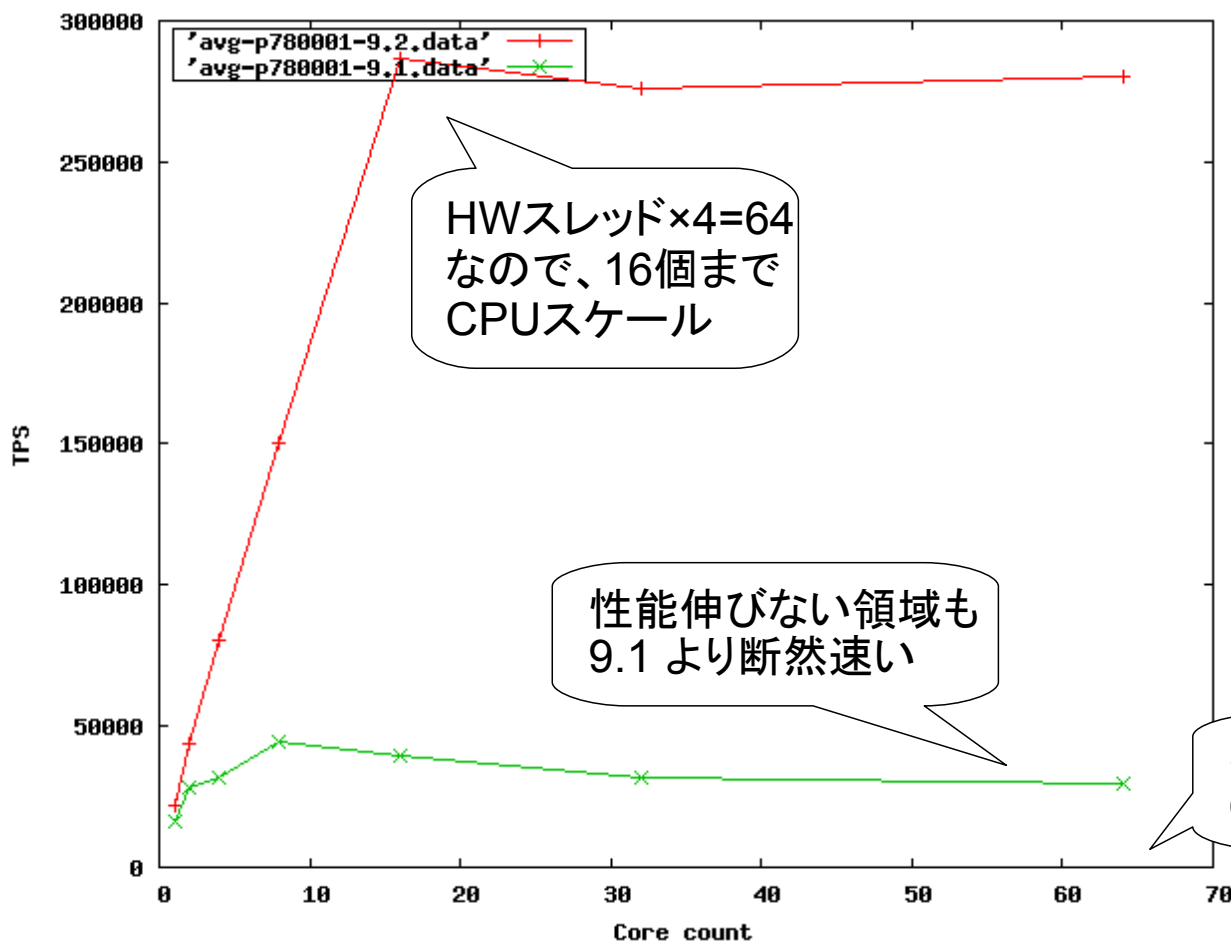
```
=> SELECT aid FROM
pgbench_accounts
WHERE aid < 300000;
```

インデックスカラムのバルク参照



# IBM Power780 による測定結果(1)

参照pgbench -s 100, -S -c 64 -j 32 -T 300



HWスレッド×4=64  
なので、16個まで  
CPUスケール

性能伸びない領域も  
9.1 より断然速い

接続数でなく、  
CPUコア数を増やす



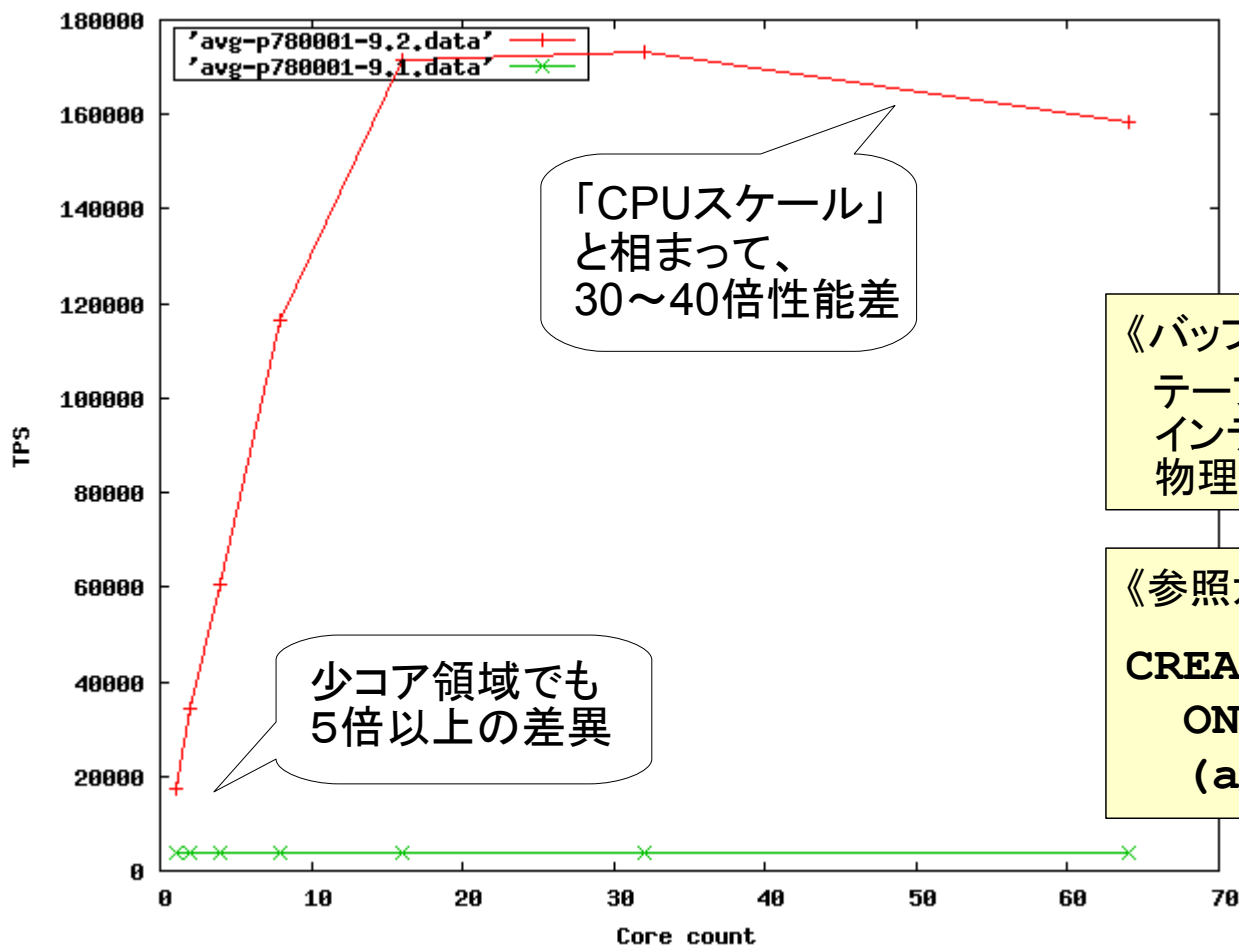
CPU: POWER7 64core × HW4thread  
Memory:64GB Storage: Fibre/RAID

64コアスケールを確認。

9.2 で POWERチップむけ  
固有の改修あり

# IBM Power780 による測定結果 (2)

Index Only Scan を pgbench -s 10000, -S -c 64 -j 32 -T 300 でテスト



Index Only Scan のある 9.2 が圧倒的な性能優位を示す。

「CPUスケール」と相まって、30~40倍性能差

少コア領域でも5倍以上の差異

《バッファに載らないデータ規模》

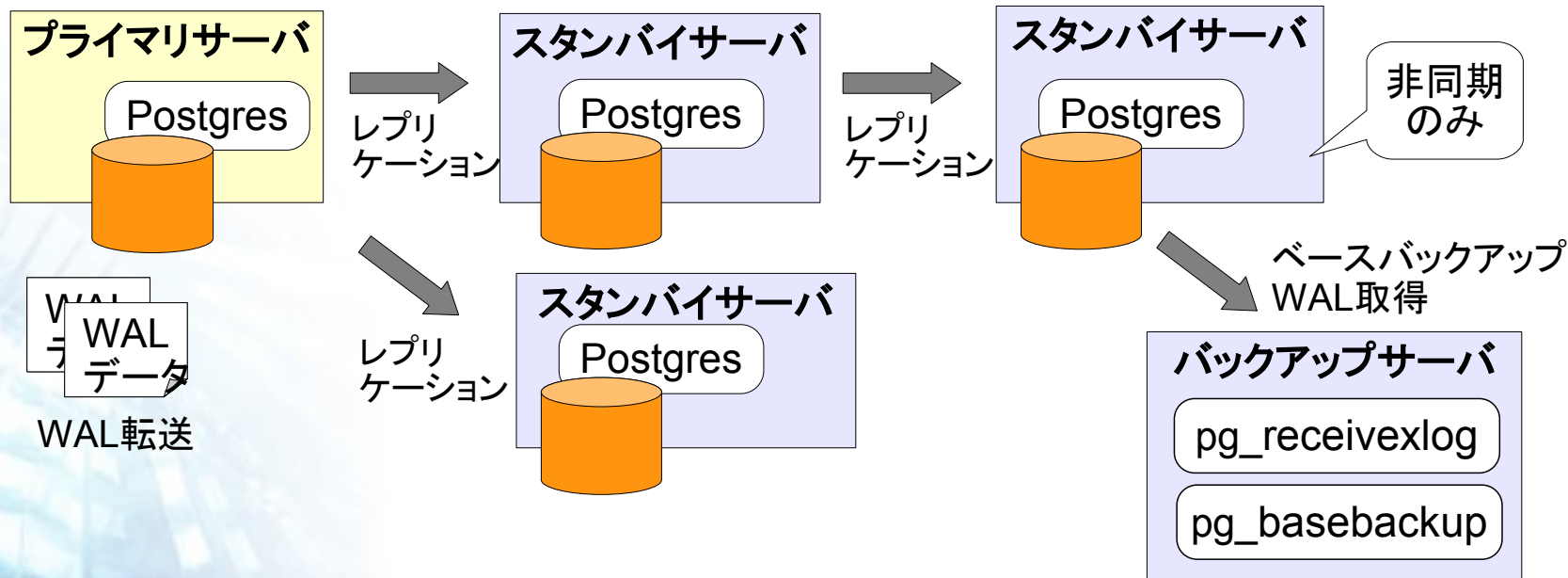
テーブル	125 GB
インデックス	42 GB
物理メモリ	64 GB

《参照カラムにインデックス作成》

```
CREATE INDEX
ON pgbench_accounts
(aid, abalance);
```

# レプリケーション拡張(1)

- カスケード構成に対応
  - マスターサーバの負荷軽減策に
  - オンライン物理バックアップをスタンバイ側から取れる



# レプリケーション拡張(2)

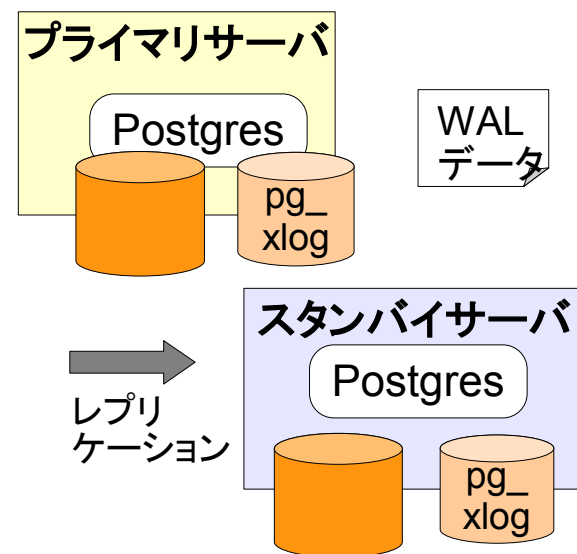
- remote\_write モードが追加

- より軽量、低信頼な同期レプリケーション

同期レプリの  
オプション

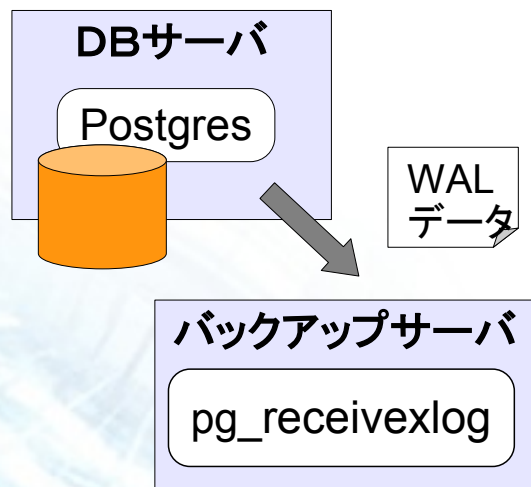
```
synchronous_commit = 'remote_write' # 'off', 'local', 'remote_write' or 'on'
```

高速 ↑ ↓ 安全		COMMIT時のWAL書き込み動作
	off	ディスク同期を待たない
	local	自機のディスク同期を待つ
	remote_write	自機のディスク同期を待つ レプリ先への書き込み完了を待つ
	on	自機のディスク同期を待つ レプリ先のディスク同期を待つ



## レプリケーション拡張(3)

- pg\_receivexlog ユーティリティが追加
  - 接続してWALを受領して保存するユーティリティ



	archive_command	pg_receivexlog
特性	プッシュ型	プル型
単位	ファイル単位	ストリーム
経路	cpコマンド等	ネットワーク経由
サーバ側の動作	アーカイブに失敗している限り pg_xlog から消えない	pg_xlog 内のWALは構わず消える。要求した古いWALが無いとエラーになる。

組み合わせ指定例:

```
archive_command = 'sleep 5 && test -f /mnt/server/archivedir/%f'
```



# Range data type (1)

- 範囲を表現するデータ型
  - 重なり検出する演算子 && など
    - gist インデックスが利用可能
  - 8.4 で導入された排他制約と組み合わせ
    - 重なりがあったら制約違反
    - これまでは BOX型くらいしか使える対象がなかった

《排他制約の定義例》

```
ALTER TABLE reservation
ADD EXCLUDE
USING gist (during WITH &&);
```

データ型	要素データ型
int4range	int
int8range	bigint
numrange	numeric
tsrange	timestamp
tstzrange	timestamptz
daterange	date

演算子	意味
<@	含まれる
&&	重なりがある
*	共通範囲
+	合併範囲
>=	範囲の大小比較

## Range data type (2)

- ユーザ定義の範囲型を利用可能

大小比較ができる型

```
=> CREATE TYPE float4range AS RANGE (SUBTYPE = float4);
```

- 境界値を含む／含まないを指定できる

```
=> SELECT '(1.5, 2.1]'::float4range;
float4range
```

```
-----
(1.5,2.1]
```

```
=> SELECT 1.5::float4 <@ '(1.5, 2.1]'::float4range;
?column?
```

```
-----
f
```

空範囲 '(,)'  
開範囲 '(0, NULL)'  
なども表現可能

境界値を含まない  
ので False になる

# JSONデータ型

- JSON構文チェック

- 配列や行からJSON型への変換関数がある

逆向きの  
変換手段は  
乏しい

```
=> SELECT * FROM t;
```

id	name	tags
1	犬	{従順, 賢い}
2	猫	{自由}
3	虎	{強い, 怖い, 希少}

これは  
文字列の  
配列型

```
=> SELECT row_to_json(t) FROM t WHERE id = 3;
           row_to_json
```

```
-----
{"id":3, "name":"虎", "tags":["強い", "怖い", "希少"]}
```

JSONの  
配列に  
なっている

# セキュリティ関連

- セキュリティバリア ビュー
  - 関数を通してアクセス不可のカラム内容が漏れることがありえたのをガード

```
=> SELECT * FROM v1 WHERE f_leak(passwd);  
NOTICE:  f_leak => passwd12345
```

```
CREATE VIEW v1 WITH (security_barrier) AS..  
ALTER VIEW v1 SET (security_barrier=true)
```

- LEAKPROOF 関数定義オプション
  - 「この関数は安全」を宣言
- SE-pgsql の対応範囲拡大

CREATE、  
DROP なども

デフォルト  
false

## その他の拡張(1) - 内部実装

- 省電力

- 補助プロセスが動作する回数を減らすことで、処理が無い時間帯においてサーバ省電力機能が効果的に働く

動作 11.5回/秒



動作 0.4回/秒

- 各種プランナ、executer改善

- Parameterized Plan選択
- 高速ソート
- 配列のコスト見積もり改善
- ほか多数...

プリペアドステートメント  
であっても、必ずしも  
プラン固定しない

# Parameterized Plan のテスト

```
=> PREPARE p1 (bigint)
    AS SELECT * FROM pgbench_accounts WHERE aid < $1;
```

```
=> EXPLAIN (COSTS OFF) EXECUTE p1(100000);
    QUERY PLAN
```

9.1 では  
プラン固定

```
-----
Seq Scan on pgbench_accounts
  Filter: (aid < 100000::bigint)
```

行数が多ければ  
Seq Scan

```
=> EXPLAIN (COSTS OFF) EXECUTE p1(10);
    QUERY PLAN
```

```
-----
Index Scan using pgbench_accounts_pkey
  on pgbench_accounts
  Index Cond: (aid < 10::bigint)
```

行数少なければ  
Index Scan

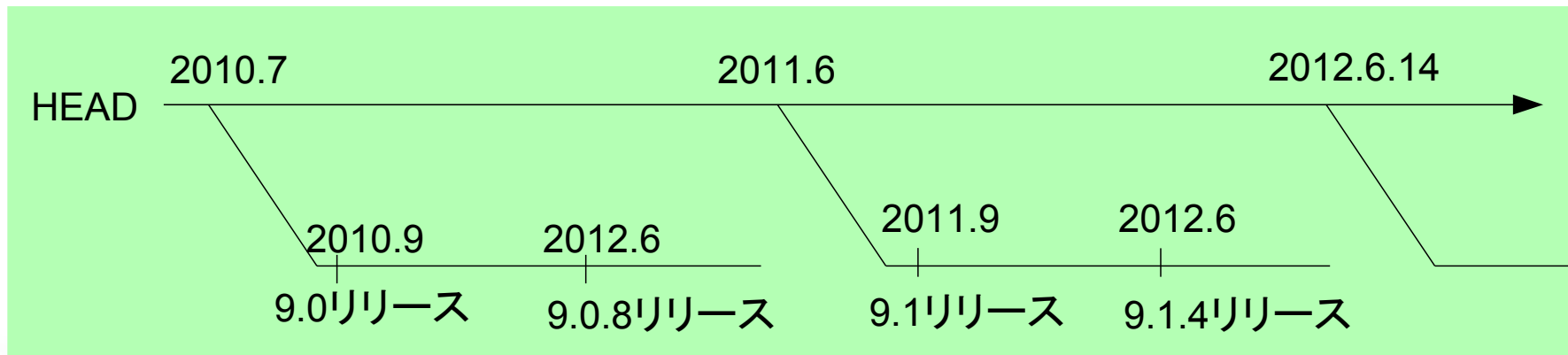
## その他の拡張(2) - 管理機能など

- contrib/pg\_stat\_statement
  - SQL集約が改善、I/O 情報収集
- 実行時統計情報の拡充
  - pg\_stat\_bgwriter チェックポイント動作の情報
  - pg\_stat\_database デッドロックカウンタ
- ALTER TABLE .. ALTER TYPE の改善
  - 必要なければ全レコード書き込みしない
- DROP INDEX CONCURRENTLY
  - 強いテーブルロックを取らない

SQL実行統計を  
収集する機能

インデックス再構築  
のオンライン実行を  
助ける機能

# PostgreSQL 9.2 のリリース予定



⇒ 今期は alpha版リリースは無かった

2012-05-14 9.2 beta1 リリース

2012-06-04 9.2 beta2 リリース

⇒ 続いて RC版リリースがある

⇒ 例年通りの流れなら 2012年初秋 正式版リリースか