

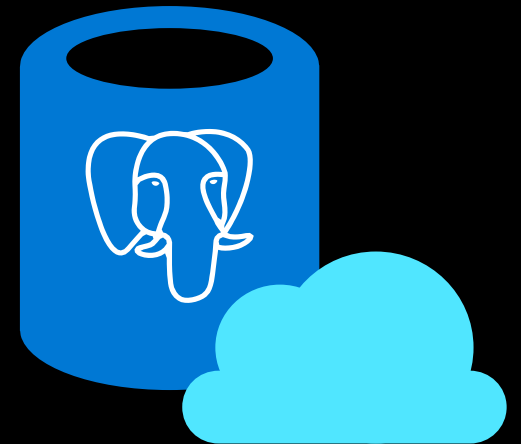


Azure Database for PostgreSQL Hyperscale (Citus)

Aud 2019

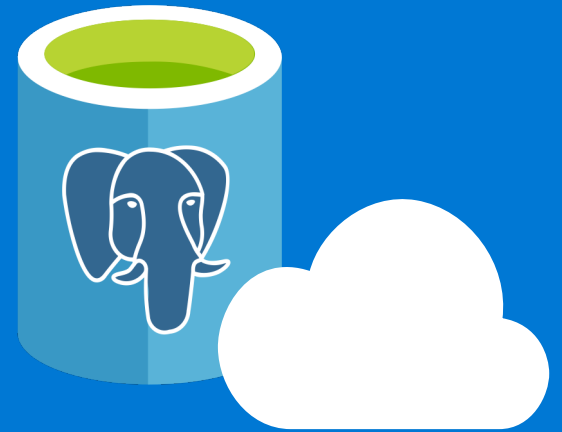
Microsoft Corporation
GBB OSS Data TSP

Rio Fujita



Azure Database for PostgreSQL

May 2019



Azure Database for PostgreSQLのデプロイオプション



Single Server

完全マネージドのHA組み込まれた
単一ノードのPostgreSQLサービス

利用例

- トランザクションでデータ操作を伴う
分析ワークロード
- JSONを使うアプリ、地理空間情報をサポート、
全文検索
- モダンなフレームワークを用いた新開発のアプリ

Hyperscale (Citus) NEW

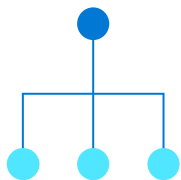
心配する必要が無い、スケールアウト設計の
クラウドのPostgreSQL

利用例

- マルチテナント、SaaSのPostgreSQLのスケーリン
グ
- リアルタイム分析
- 高スループットのトランザクションを扱うアプリ

Azure Database for PostgreSQLの利点

確信をもってワークロードを構築・移行



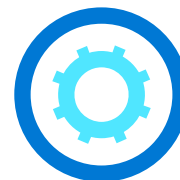
完全マネージで安全

Azure がリソースを大量に消費するタスクを管理し、さまざまなバージョンのPostgres をサポートし、業界最高の補償範囲を提供しながら、アプリに集中



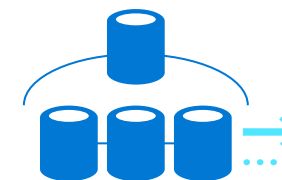
知的な性能最適化

カスタマイズ可能な推奨項目による性能の改善とコスト低減



柔軟でオープン

お気に入りの Postgres 拡張機能で生産性を維持し、Postgres コミュニティへのマイクロソフトの貢献を活用



Hyperscale (Citius)による 超高性能と スケールアウト

単一ノードPostgresの限界から脱却し、100ノード間でスケールアウト

Single Server

Hyperscale (Citius) ^{NEW}

Azureはオープンソースによる革新にコミット

PostgreSQLへの貢献

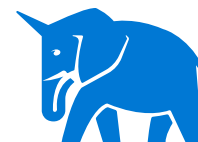
コミュニティに



Azure Data Studio
Extension



Visual Studio Code
Extension



Citus Community
Extension

And more, including:
pg_auto_failover
cstore_fdw
hll
pg_cron
topN

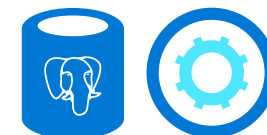
コミュニティと



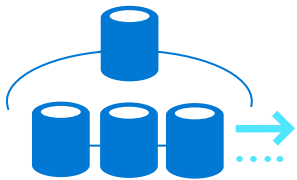
Timescale DB



Hasura



High performance scale-out with Hyperscale (Citrus)



水平にスケールアウト

単一ノードPostgresの限界から脱却し、100ノード間でスケールアウト



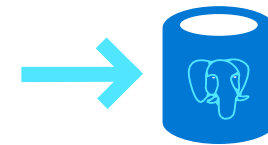
燃えるようなパフォーマンス

数十億行のデータに対し、1秒以下のレスポンスで、リアルタイムでデータベースを取り込み、クエリを実行



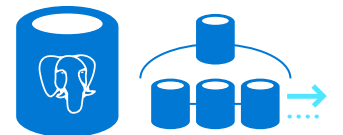
インフラの簡素化

時間の節約。
トランザクションと分析を1つのデータベースで。
さらに手動でのシャーディングのコストを避ける。



PostgreSQLの革新と共に

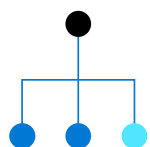
オープンソース拡張機能として開発され、すべてのPostgreSQLの専門知識とその最新のイノベーションを活用



Hyperscale (Citius)

Key use cases

Key uses cases for Hyperscale (Citrus)



マルチテナントとSaaSのアプリ

単一ノードの限度を超える

テナントを分散してホットスポットを最小化

オンラインで再度バランスすることが可能

大量のテナントをハードから独立



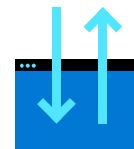
リアルタイムの運用分析

数テラバイト/日のデータを投入

1秒未満のクエリレスポンス

ノードを並列化し100倍の性能を実現

複雑なETL処理を単純化



高スループットのトランザクション/OLTPアプリ

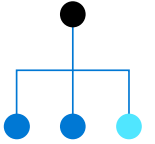
多数の同時ユーザ数でも高性能を維持

SPOFを回避

複数のノードにトランザクション処理を分散

大量のトランザクションを管理

マルチテナント & SaaSアプリをスケール



数10万テナントまで簡単にスケールするアプリを構築

必要とするシナリオ

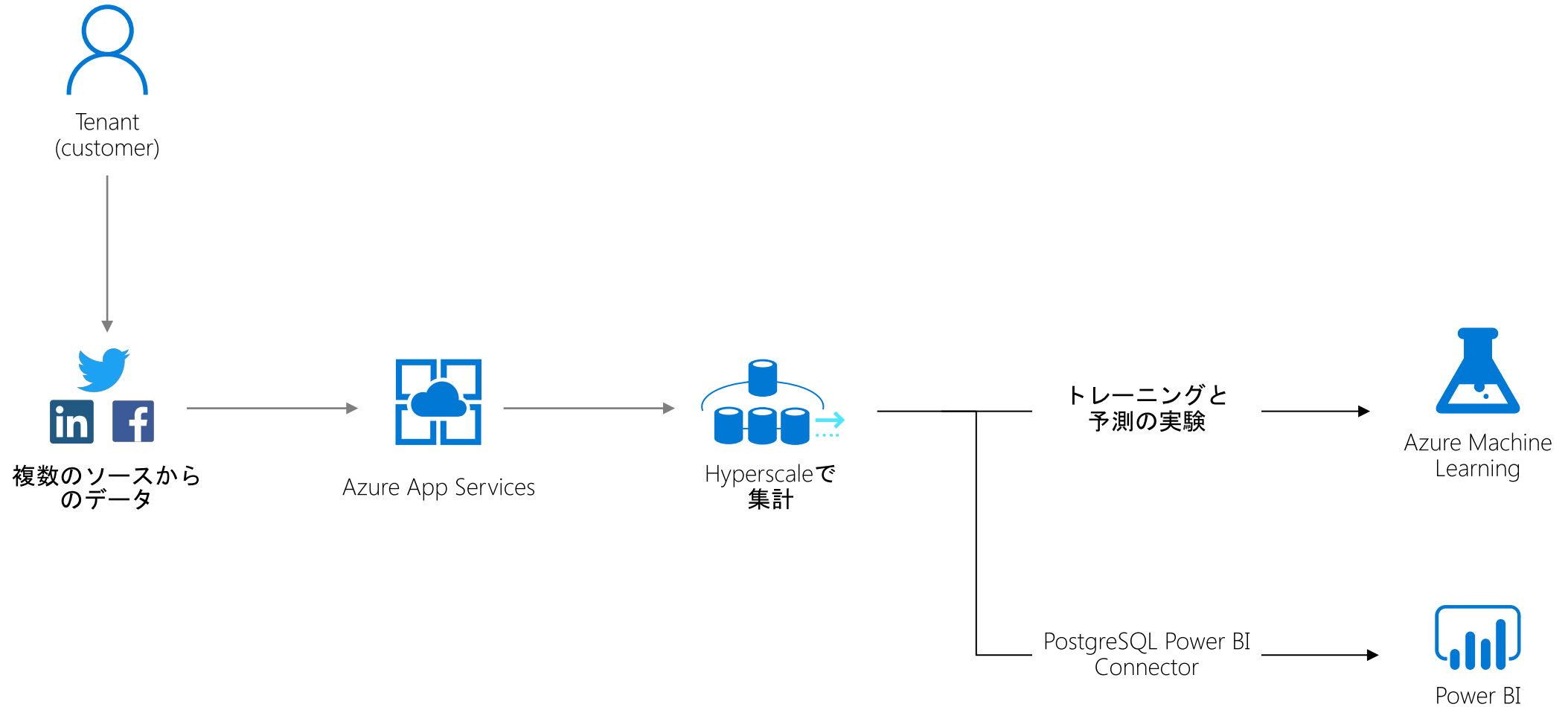
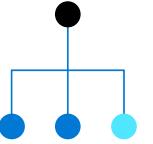
- ビジネスの成長をサポート出来る性能をもつデータベースが必要
- オープンソースのソリューションが好ましい（ロックインされたくない）
- PostgreSQLの信頼性、豊富なデータ型、拡張機能、専門知識を活用したい
- 中核となるビジネスアプリの開発者をデータベースの最適化に投入するのは無駄
- 本番環境に影響を与えずに、開発者が試せる、アナリストが分析出来ることを可能にしたい

Hyperscale (Citrus) solution

- 好みであるPostgreSQLの利用とエコシステムの活用により構築が容易: 単一のデータベース
- 慣れ親しんだPostgreSQLのツール（同じチーム、追加コスト無し）により運用維持が容易
- より良いリソースの利用率（2倍以上の性能）

マルチテナント & SaaSアプリをスケール

数10万テナントまで簡単にスケールするアプリを構築



リアルタイムの運用分析とレポート作成



数十億のイベントに対する1秒未満のクエリ

必要とするシナリオ

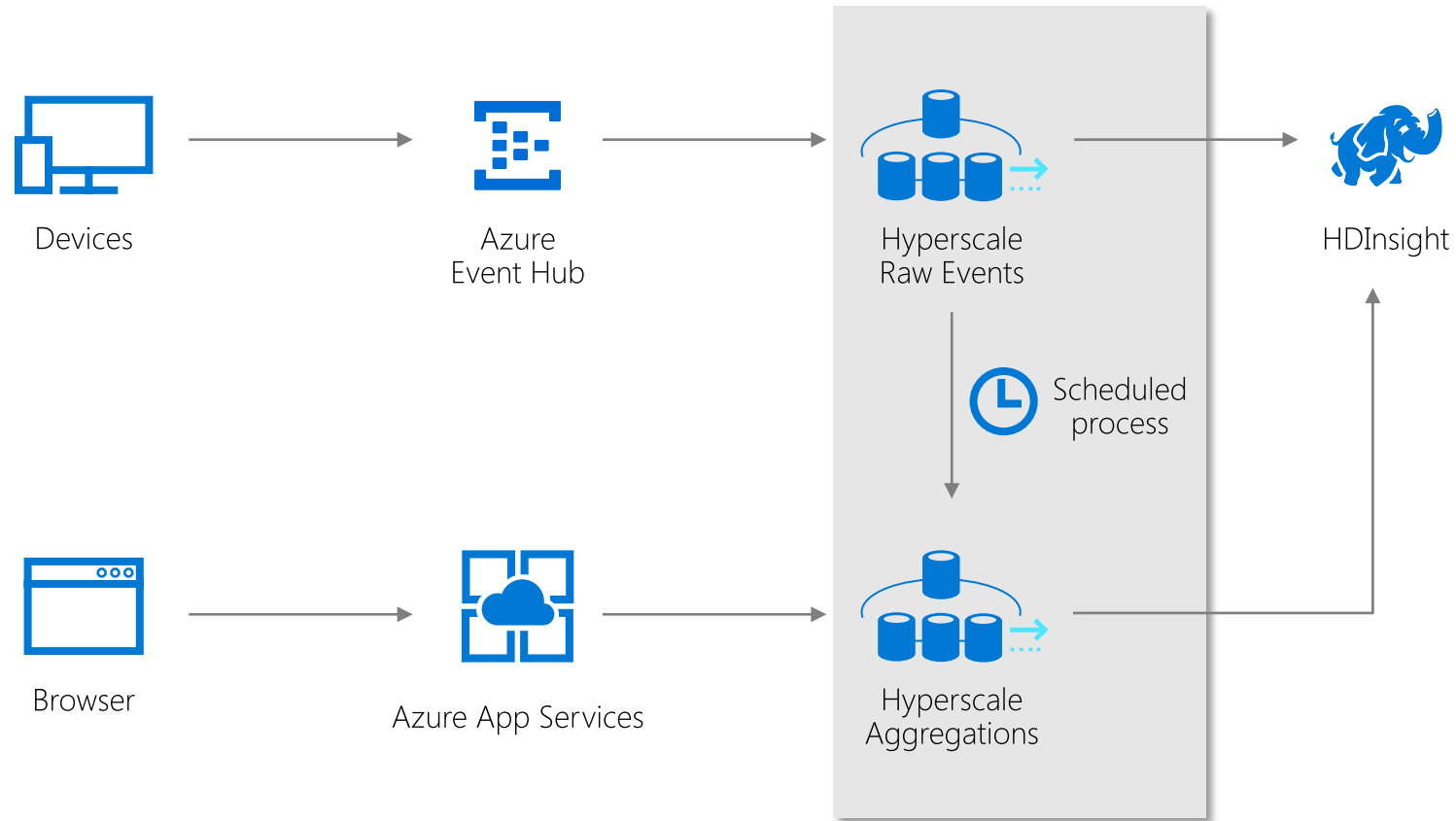
- PostgreSQLの信頼性、豊富なデータ型、拡張機能、専門知識を活用したい
- 顧客が新しいイベントと新しいデータをリアルタイムで分析できるように
- データベース内でロールアップや事前集計を実行
- 応答を待ちたくない、同時接続のユーザーの負荷がある顧客向けダッシュボード
- SQL とミリ秒の応答を並列化する分析ソリューション

Hyperscale (Citus) solution

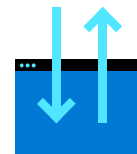
- 低コスト：コンピュート、メモリ、ストレージの利用率の向上
- 大量のユーザでの1秒未満の応答（高度な並列）
- ETLを分離せずに書き込みを分析
- 容易なノードの追加
- 広範なオープンソースシステムによる進化
- より高速な並列クエリとインデクシング

リアルタイムの運用分析とレポート作成

数十億のイベントに対する1秒未満のクエリ



高スループットのトランザクション/OLTPアプリ



数十億のイベントに対する1秒未満のクエリ

必要とするシナリオ

- PostgreSQLの信頼性、豊富なデータ型、拡張機能、専門知識を活用したい
- PostgreSQLの信頼性、データ型、拡張機能、ツールセット、専門知識の先進性を利用したい
- アプリの性能向上のために再設計する時間を浪費したくない
- スケールする性能が欲しい

Hyperscale (Citus) solution

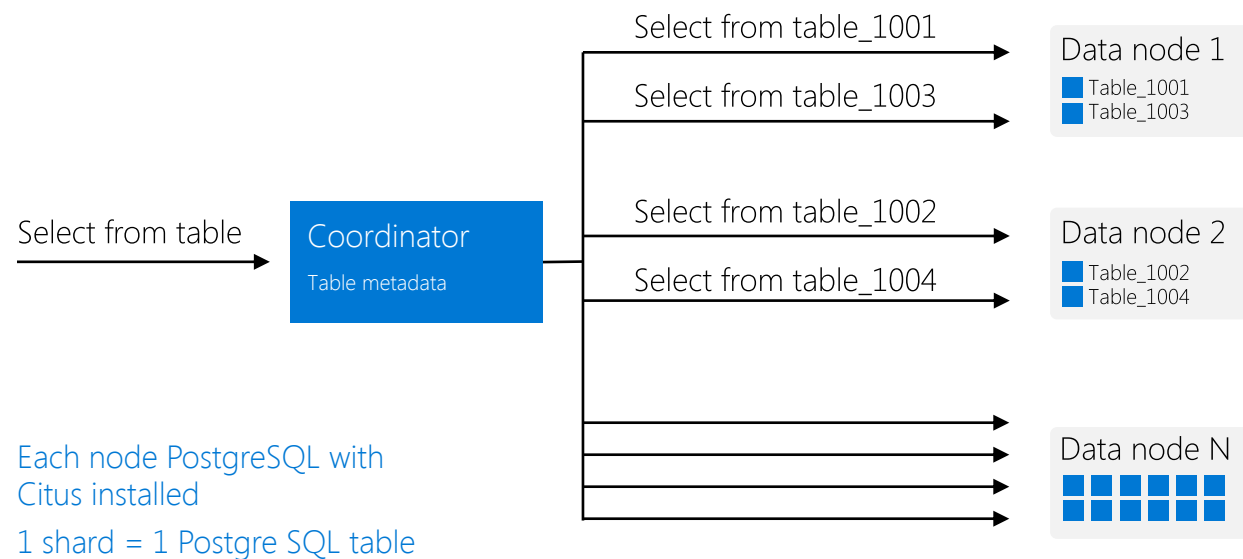
- 低コスト：メモリ、ストレージの拡張による利用率の向上。複数のデータベースに分割する必要が無い
- 大量のユーザでの1秒未満の応答（高度な並列）
- 容易なノードの追加
- 広範なオープンソースシステムによる進化
- より高速なデータのロードとインデクシング

100ノードで構成する 単一のPostgreSQL

PostgreSQLデータベースを複数のノードに分け、
アプリケーションにより多くのメモリ、
コンピューティング、
ディスクストレージを提供

各ノード内でも並列処理を実現しながら、
ワーカーノードを簡単に追加して
水平スケールを実現

100ノードにスケールアウト



データの スケールアウトを 効率的に管理

シャード・リバランサーは、
データスケールアウトをバランスするために、
新旧ワーカーノードにシャードを再配布

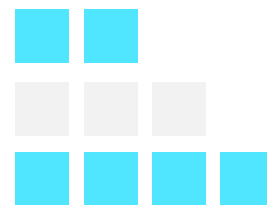
シャード・リバランスは、シャードをより均等
に配置できる場合にリバランスを推奨

より詳細な制御のために、テナント分離を使用
して、より高いニーズを持つ特定のテナント専
用に割り当てることが簡単に

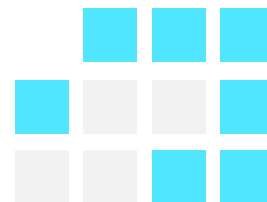
Citus Cloud Shard Rebalancer

■ Unaffected Shard ■ Shard to be moved (Source) □ Shard being moved (target) ■ Successfully moved shard

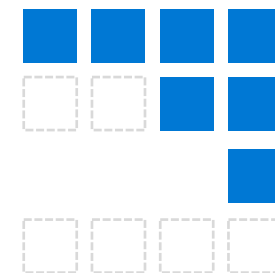
Node 1



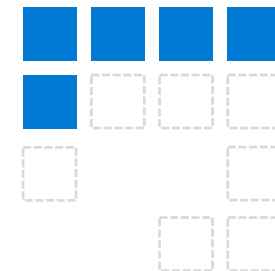
Node 2



Node 3



Node 4

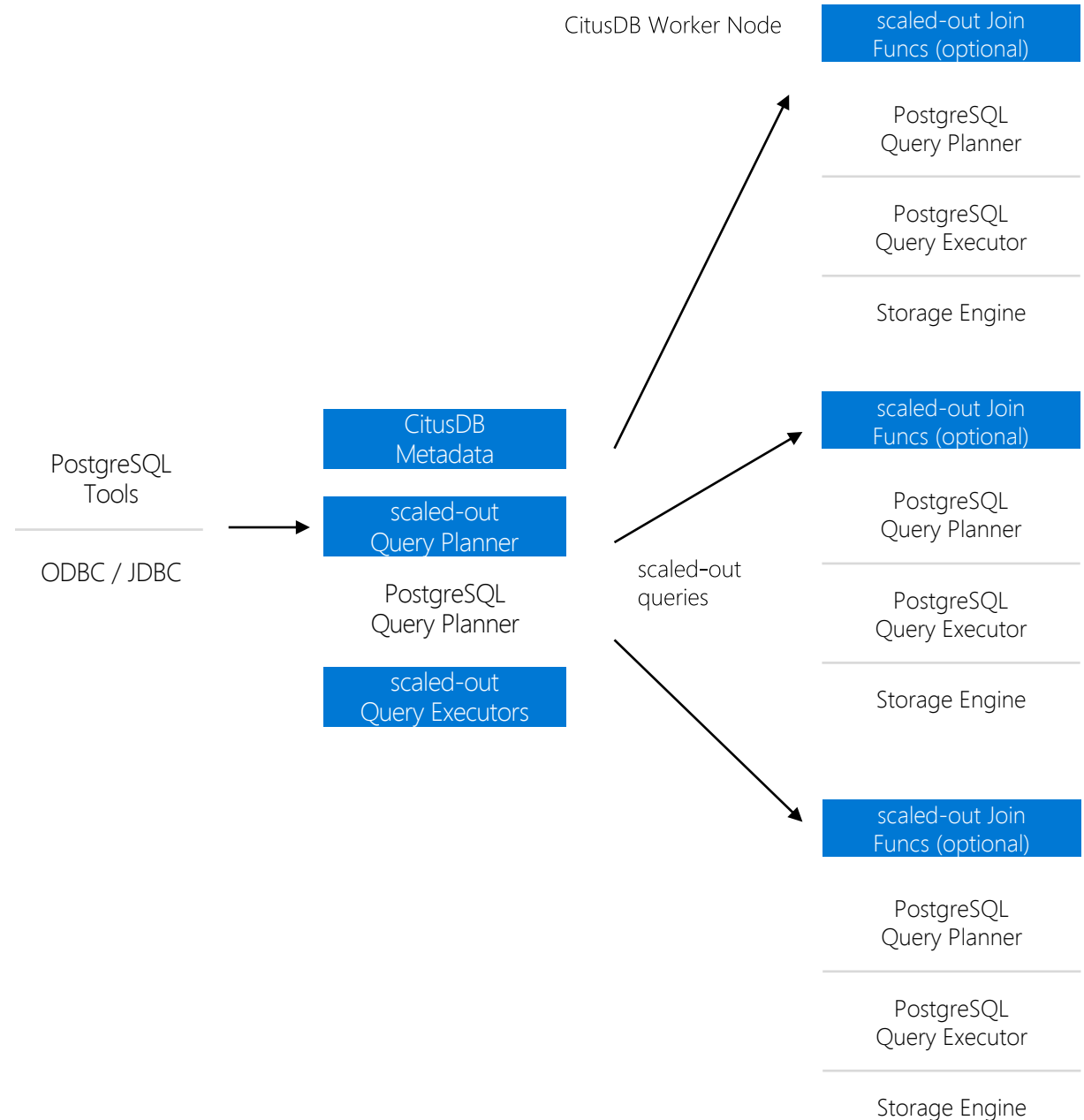


クエリをスケールアウトして高速に処理

クエリを自動的にスケールアウトして、データベース クラスタ全体で使用可能なすべてのメモリコアと CPU コアを活用

Hyperscale (Citius) は、クエリの種類に基づいて 3 つの組み込みクエリ実行プログラムから自動的に選択

- リアルタイム実行プログラム（既定）：
フィルタ、集計、および co-located join を使用したクエリに対する高速な応答
- ルーター実行プログラム：
必要なすべてのデータが単一のノードに保存される場合
- タスクトラッカー実行プログラム：
長時間実行される
複雑なデータウェアハウジングクエリ用

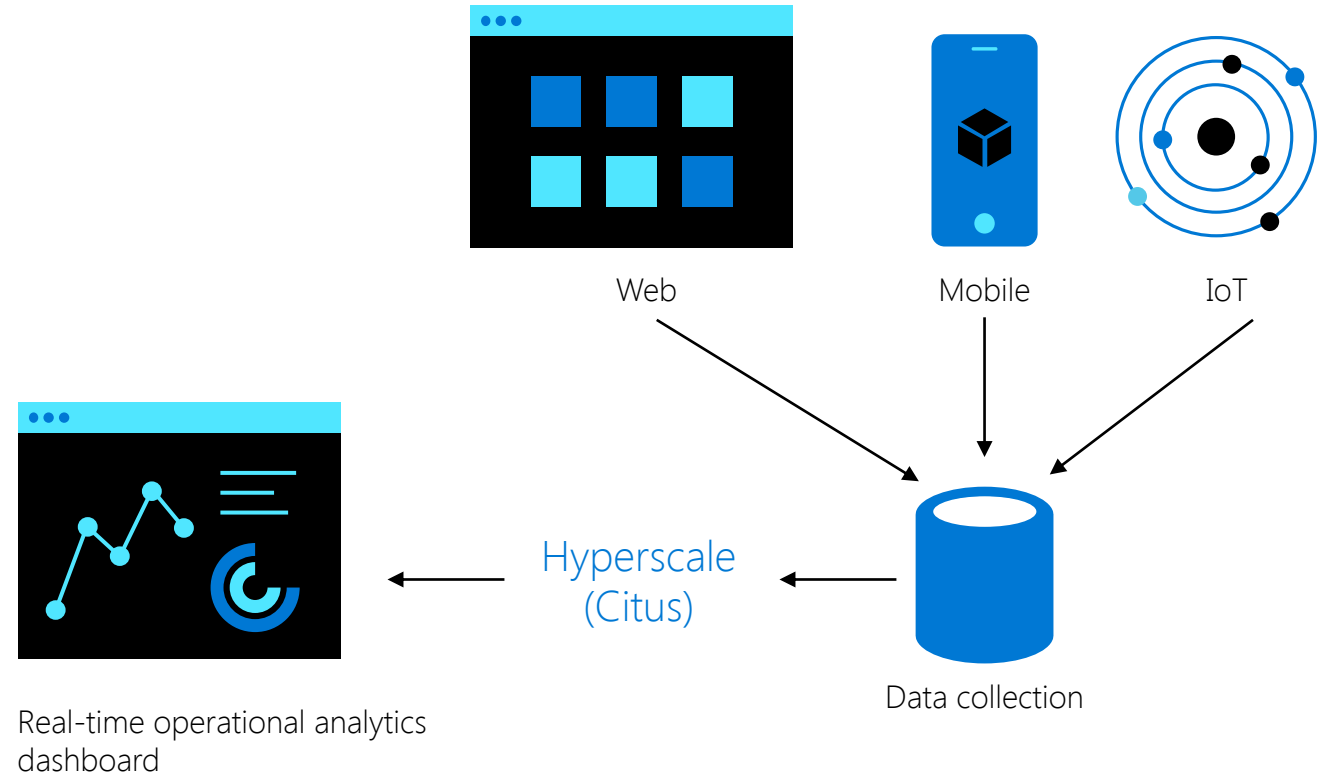


大規模な並列処理で スケールアウト分析を 実行

単一のクエリを要素に分割し、
CPU コアの広大な並列処理を実現

データセット全体にわたって分析をリアル
タイムで実行し、新しいビジネスインサイ
トを明らかにする

テナント間クエリサポートにより、
すべての顧客データにわたって
強力なクロスシャードクエリを実行



PostgreSQLの専門知識を活用

Hyperscale (Citus) はPostgreSQLの拡張機能として開発され、フォークではない

Hyperscale (Citus)は、PostgreSQLの最新の技術革新、バージョン、ツールと常に互換

PostgreSQLで利用できる、すべてのデータ型、オペレータ、関数、ツールを使用して、PostgreSQLの専門知識を活用

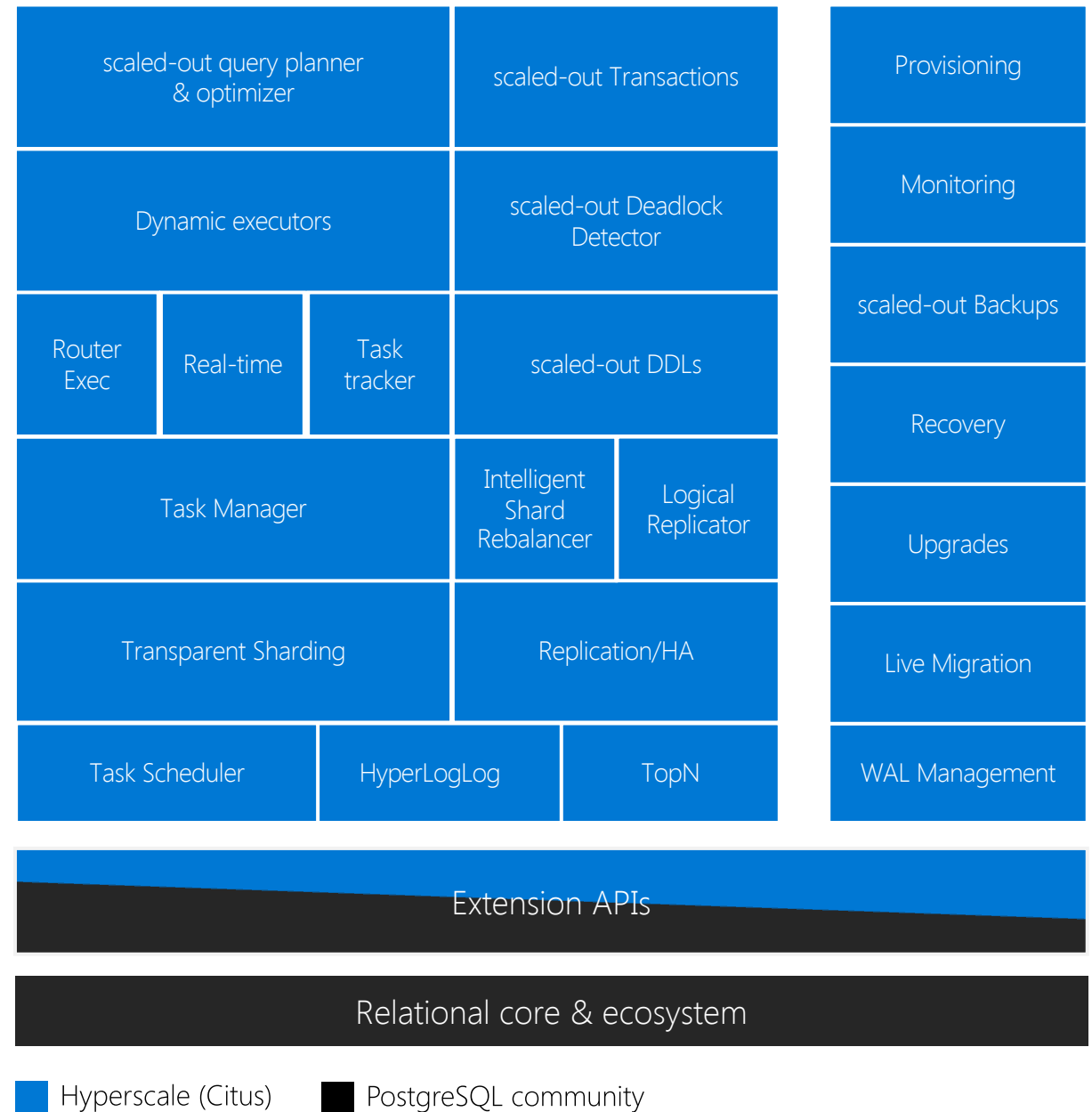
PostgreSQLコミュニティの信頼性と継続的な革新の恩恵を受けられる



PostgreSQLへの エンジニアリングの 投資の恩恵を受ける

Hyperscale (Citus) はPostgreSQLを
スケールアウトする業界のリーダー

Hyperscale (Citus) はPostgreSQLコミュニ
ティに深い知識とコミットメントをもたらし、スケーラビリティのための無料のソリューションを開発します



Hyperscale (Citrus)

Horizontally Scaled-Out

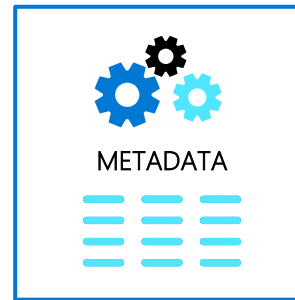
集計のスケールアウト

トランザクションの前にデータを集約すると、各行の書き換えを回避でき、書き込みオーバーヘッドとテーブルの肥大化を節約可能

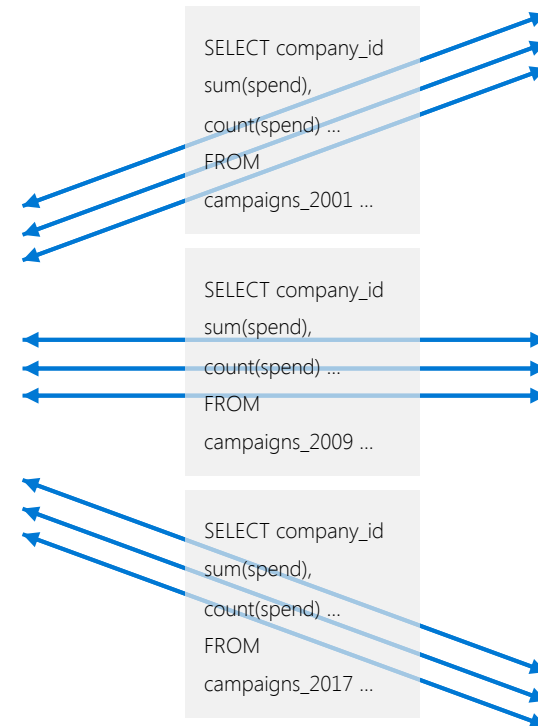
一括集約により同時実行の問題を回避

APPLICATION

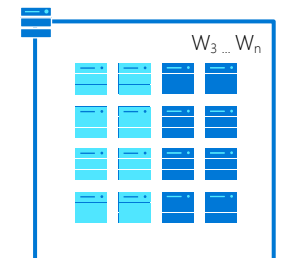
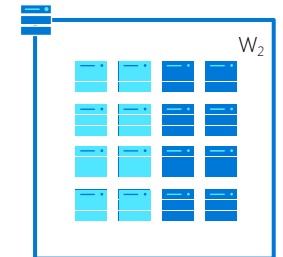
```
SELECT company_id  
       avg(spend) AS avg_campaign_spend  
FROM   campaigns  
GROUP BY company_id
```



COORDINATOR NODE



WORKER NODES



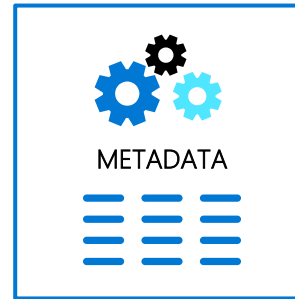
Co-located join

関連するテーブルの関連行を含むシャードを同じノードと一緒に配置

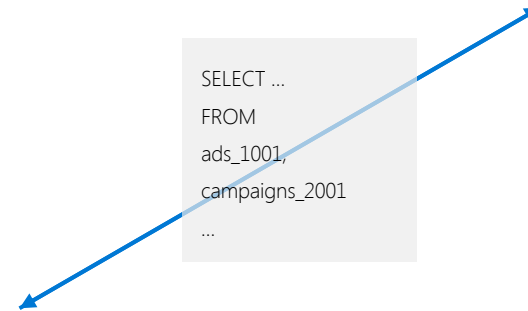
関連する行間でクエリを結合すると、
ネットワーク上で送信されるデータの量を減らすことが可能

APPLICATION

```
SELECT count(*)  
FROM ads JOIN campaigns ON  
      ads.company_id = campaigns.company_id  
WHERE ads.designer_name = 'Isaac'  
      AND campaigns.company_id = 'Elly Co'
```



COORDINATOR NODE

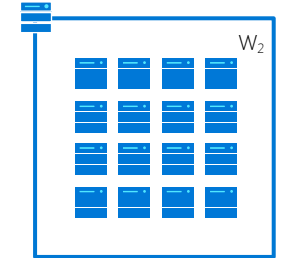


```
SELECT ...  
FROM  
ads_1001,  
campaigns_2001  
...
```

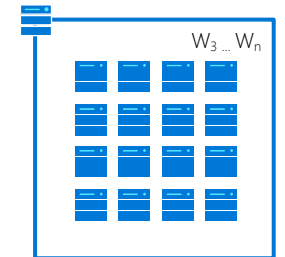
WORKER NODES



W₁



W₂



W₃ ... W_n

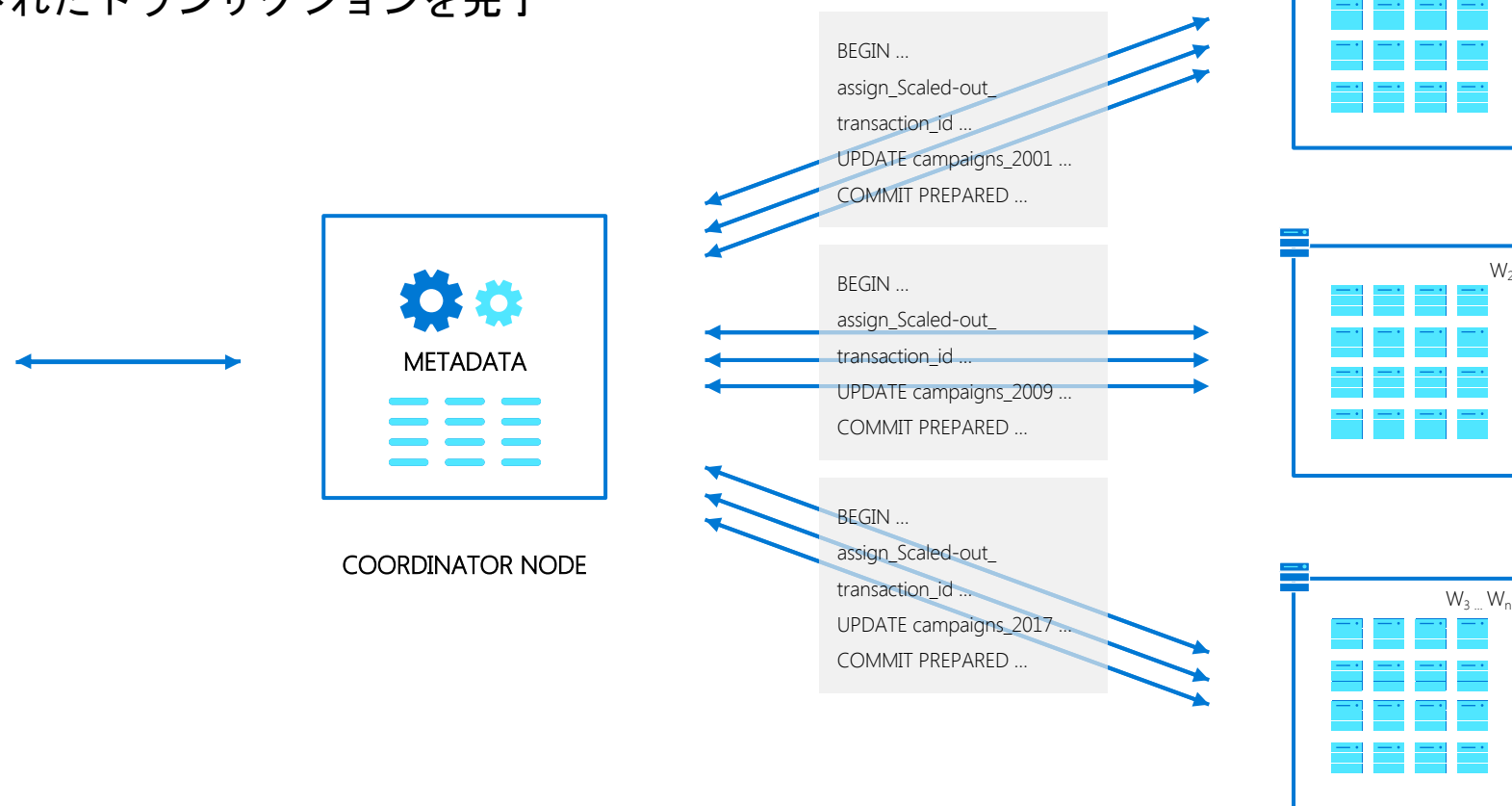
トランザクションのスケールアウト

Hyperscale (Citus) は、組み込みの2PCプロトコルを活用して、コーディネータノードを介してトランザクションを準備

ワーカーがトランザクションにコミット、ロックを解放し、受信確認を送信すると、コーディネータはスケールアウトされたトランザクションを完了

- APPLICATION

```
BEGIN;  
UPDATE campaigns  
  SET feedback 'relevance'  
WHERE company_type 'platinum'  
UPDATE ads  
  SET feedback 'relevance'  
WHERE company_type 'platinum'  
COMMIT;
```

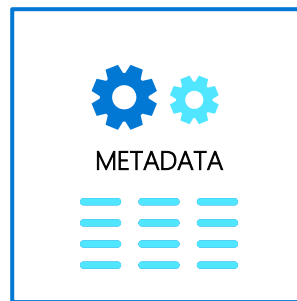


スキーマの変更

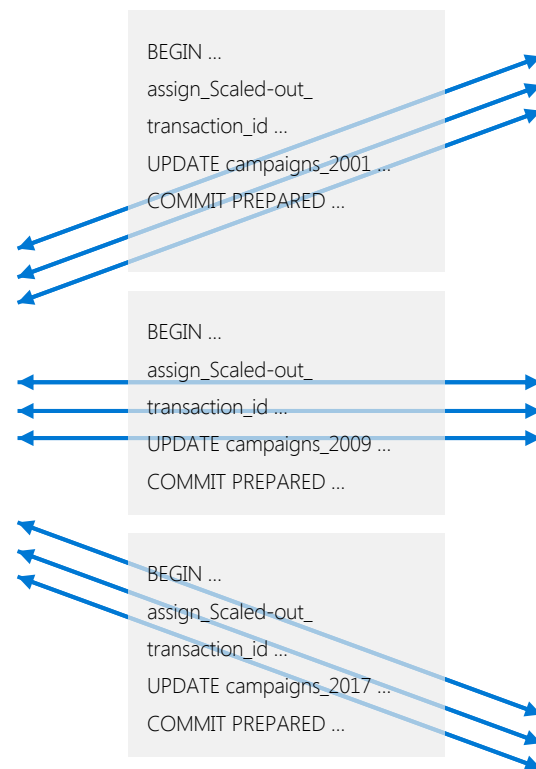
スキーマは、テーブルの種類とスケールアウト設定の変更時に更新が可能
移行用のソーステーブルの準備とスケールアウトキーの追加

APPLICATION

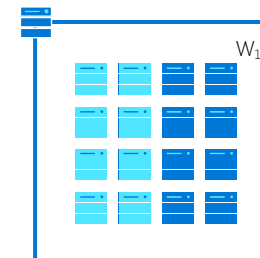
```
-- Schema Change  
ALTER TABLE campaigns  
ADD COLUMN company_type text
```



COORDINATOR NODE



WORKER NODES



W₁



W₂



W₃ ... W_n

Citus vs. Single-Node Postgres Database | A Side-by-Side Comparison

- <https://www.youtube.com/watch?v=g3H4nGsJsl0>

REGULAR POSTGRES

```
citus=> \copy github_events FROM 'in.csv' WITH CSV
COPY 1146625
Time: 109330.073 ms
citus=> CREATE INDEX p_idx ON github_events
citus-> USING GIN (payload jsonb_path_ops);
CREATE INDEX
Time: 109493.491 ms
citus=> █
```

109 sec.

CITUS

```
citus=> \copy github_events FROM 'in.csv' WITH CSV
COPY 1146625
Time: 87123.368 ms
citus=> CREATE INDEX p_idx ON github_events
citus-> USING GIN (payload jsonb_path_ops);
CREATE INDEX
Time: 7976.715 ms
citus=> █
```

8 sec.

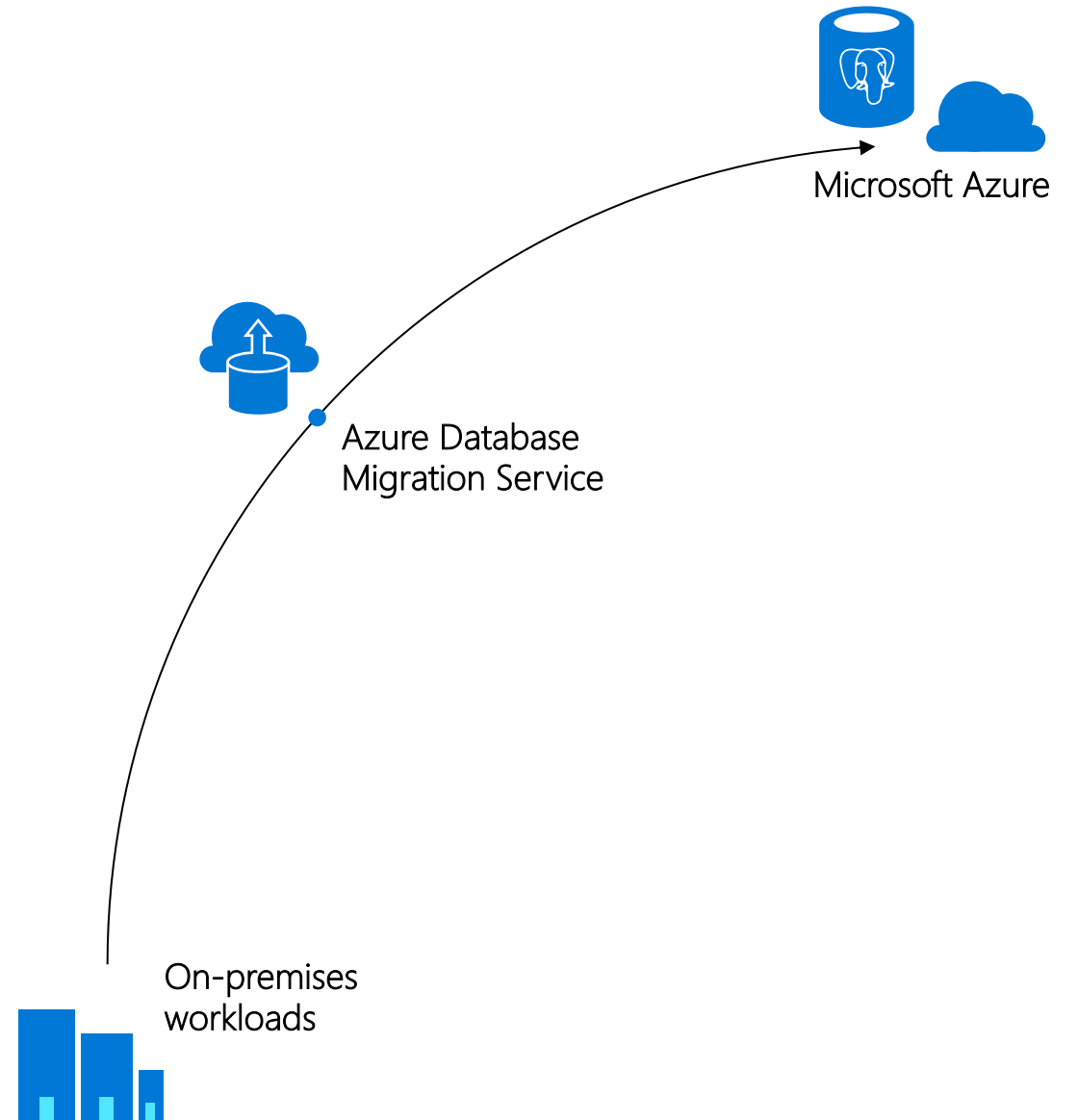
>10X faster index creation

Resources for migration

Database Migration Guide
<http://aka.ms/datamigration>

Azure Database Migration Service
Migrate with minimum downtime
<http://aka.ms/get-dms>

Sign up for Preview
<http://aka.ms/dms-preview>



General resources to learn more

Azure service page: <http://aka.ms/postgresql>

Documentation: [Azure Database for PostgreSQL](#)

Discussion forum: [MSDN](#), [StackOverflow](#)

Feedback forum: [User Voice](#)

Hands-on Lab: <http://aka.ms/postgresqlhol>

GitHub repo: <https://github.com/Azure/azure-postgresql>

Handson lab

<http://tinyurl.com/y4py882l>

日本語訳

https://github.com/rrioriost/Citus_doc_jp