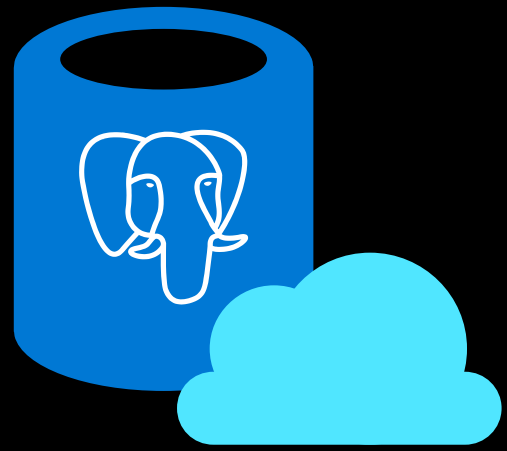
Microsoft

Azure Cosmos DB for PostgreSQL

Feb 2024

Microsoft Corporation
GBB OSS Data Senior SP

Rio Fujita



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

Flexible Server

ゾーン冗長HA、最大限の制御、簡素化された開発者エクスペリエンスを備えたフルマネージドPostgreSQL データベースサービス

使用例

- トランザクション分析および運用分析ワークロード
- JSON、地理空間サポート、または全文検索を必要とするアプリ
- 最新のフレームワーク、低いTCOおよび最新のPostgreSQL バージョンで構築されたクラウドネイティブアプリケーション
- ゾーンコロケーションを利用して低レイテンシを実現する高性能アプリ

Cosmos DB for PostgreSQL

スケールアウトするように構築されたアーキテクチャを備えたクラウド内の心配のない PostgreSQL

使用例

- PostgreSQL マルチテナント SaaS アプリケーションのスケーリング
- リアルタイムの運用分析
- 高スループットのトランザクション アプリの構築

Single Server (Legacy)

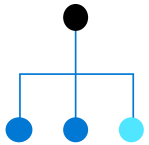
ゾーンHAを備えた完全管理された単一ノードPostgreSQLデータベースサービス

使用例

- 最新のフレームワークで構築されたクラウドネイティブアプリ
- きめ細かな制御やカスタム構成設定を必要としないアプリ
- トランザクション分析および運用分析ワークロード

Azure Arc enabled PostgreSQL Hyperscale (Preview) は、Azure Arc 対応のデータ サービスの一部としてプレビュー段階になりました。Azure クラウドの革新と、クラウドへの直接接続の有無にかかわらず、任意のインフラストラクチャで動作するハイパースケール データ ワークロードの統合ハイブリッド管理エクスペリエンスのメリットを享受できます。

Key uses cases for Azure Cosmos DB for PostgreSQL



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

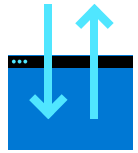
リアルタイムの運用分析

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

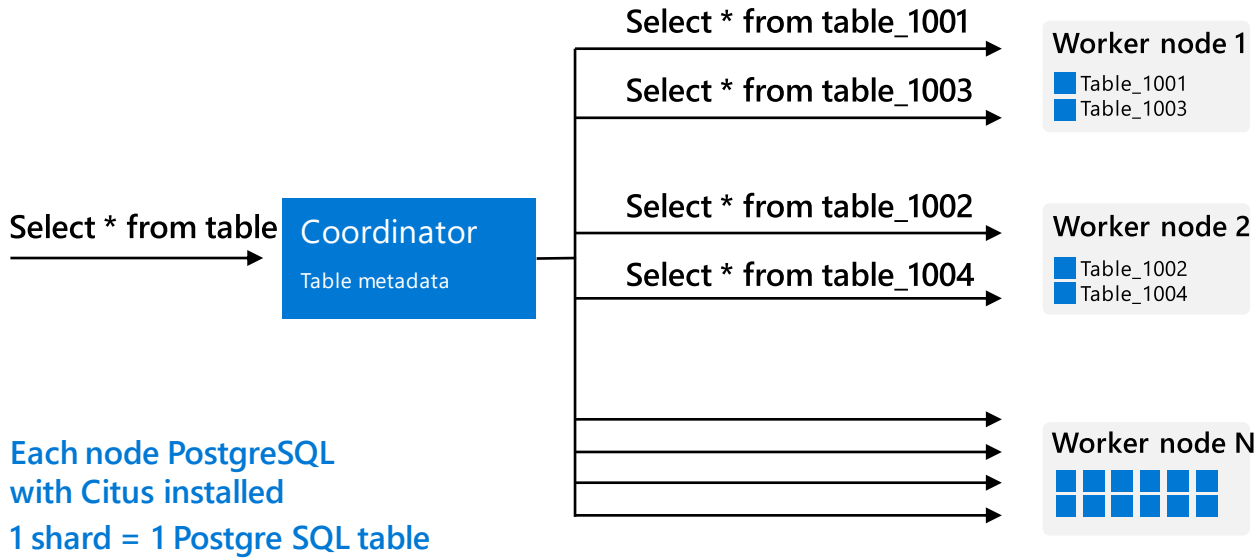
大量のトランザクションを管理

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

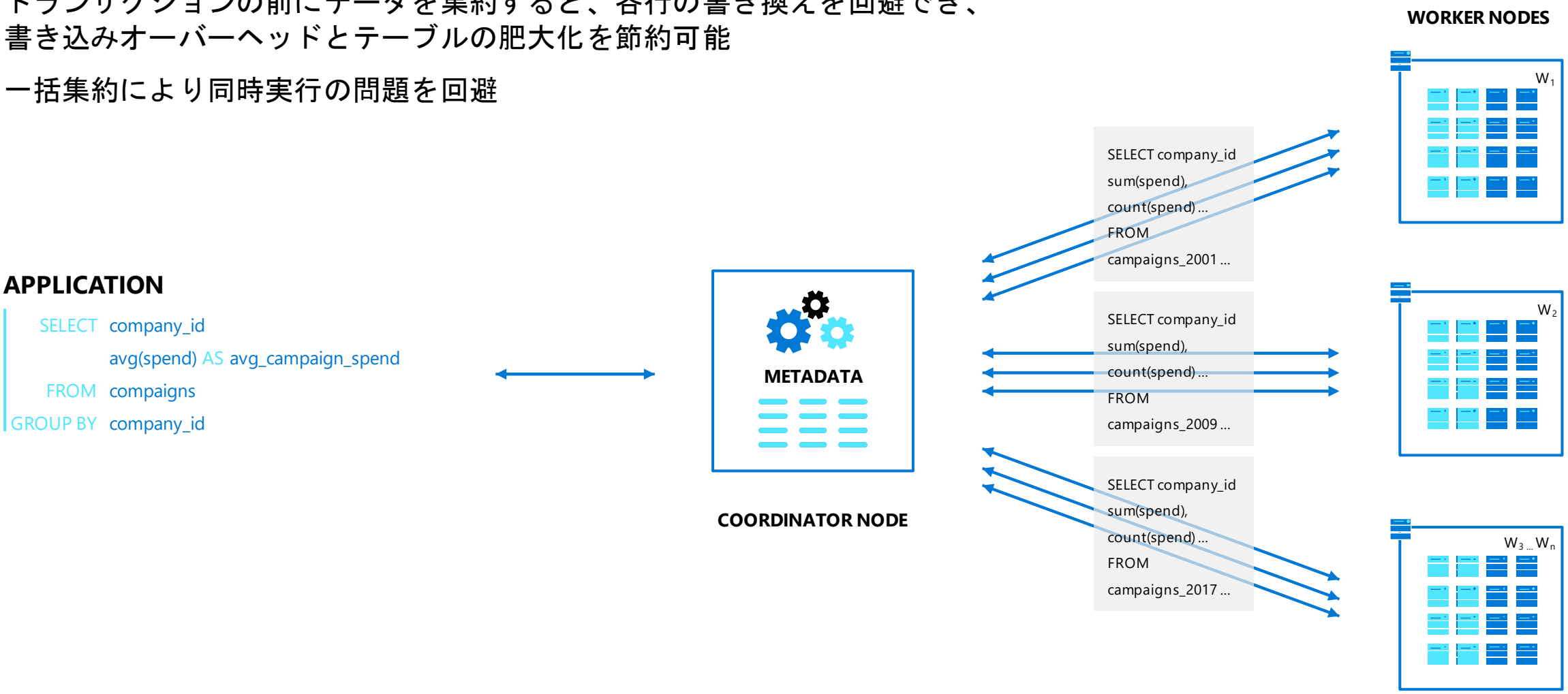
100ノードにスケール アウト



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



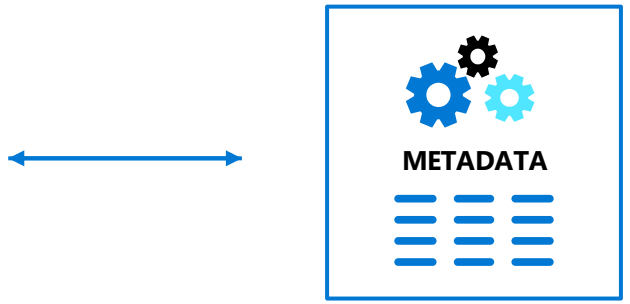
Co-located join

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に一緒に配置

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

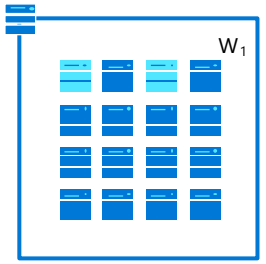
```
SELECT  count(*)
  FROM  ads JOIN campaigns ON
        ads.company_id = campaigns.company_id
 WHERE  ads.designer_name = 'Isaac'
   AND  campaigns.company_id = 'Elly Co'
```

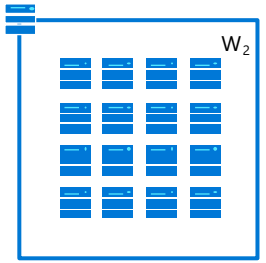


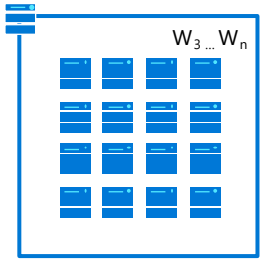
COORDINATOR NODE

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

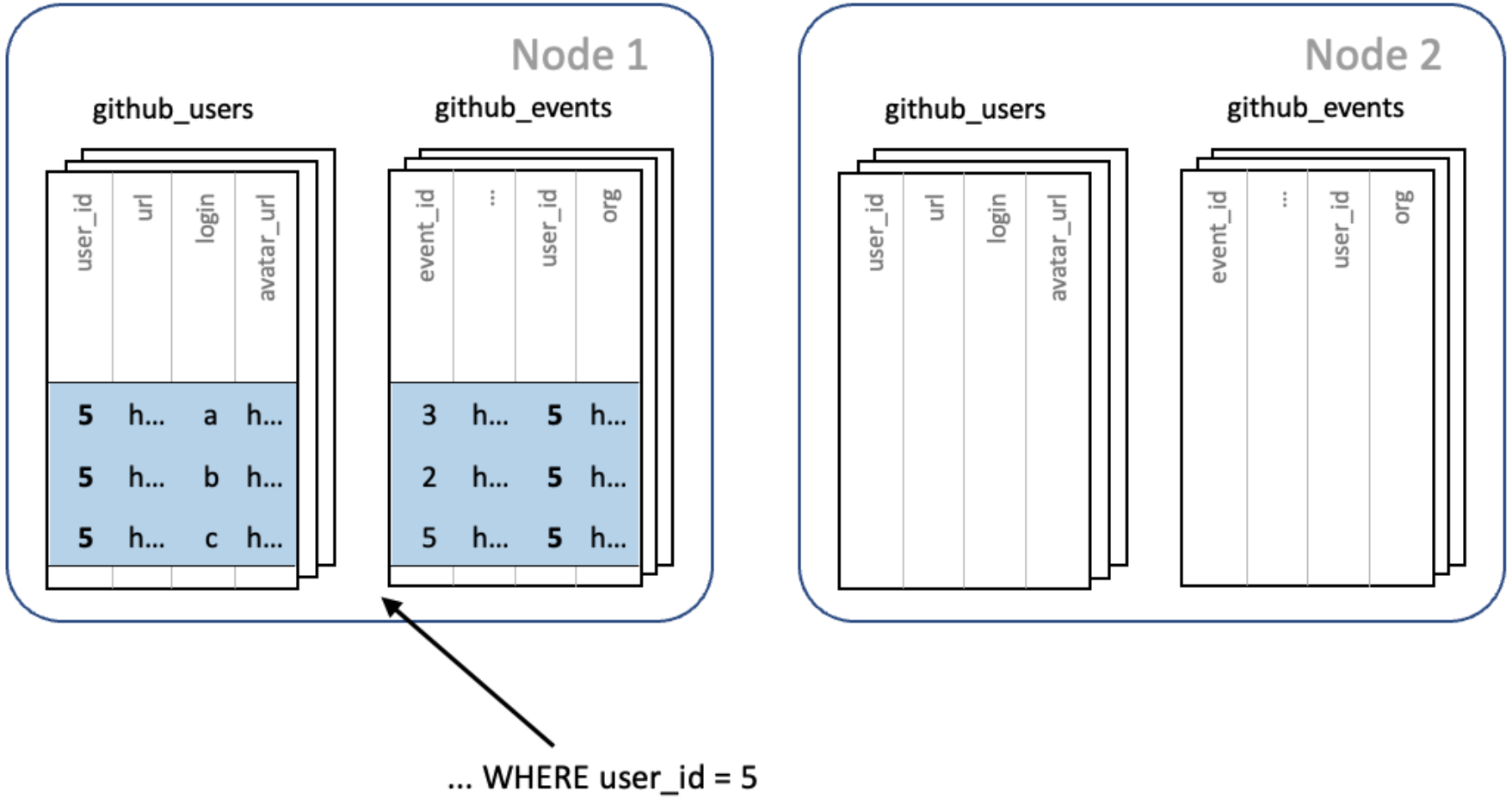
WORKER NODES



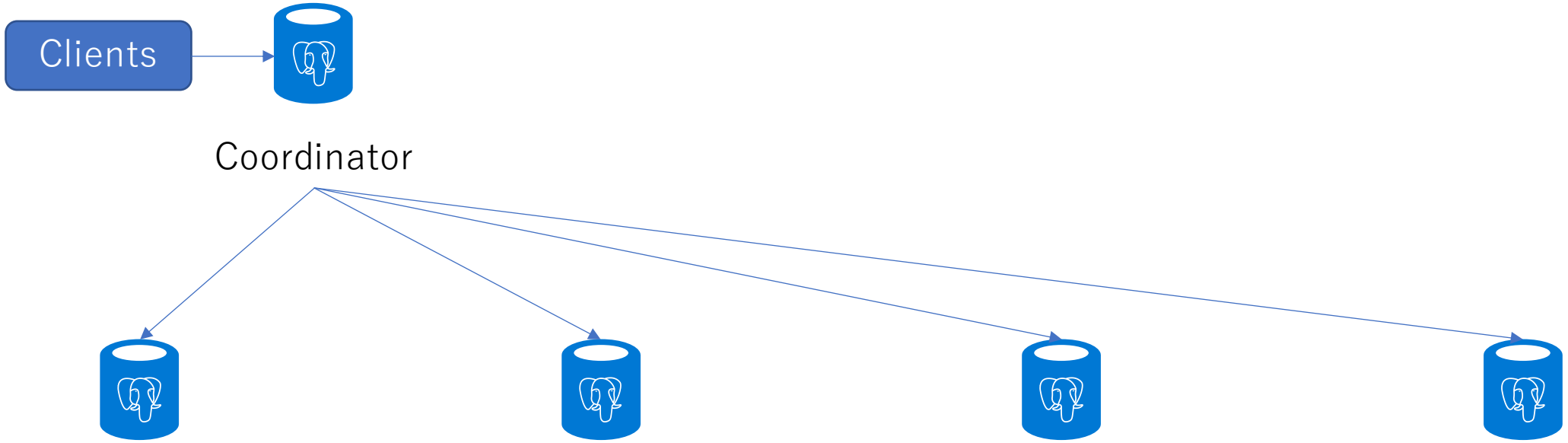




Co-located join (contd.)



データ格納イメージ（シャーディング）



Plant ID	Time	Data 1
1		
1		
5		
5		
1		
5		

Plant ID	Time	Data 1
2		
6		
2		
2		
6		
6		

Plant ID	Time	Data 1
7		
3		
3		
7		
3		
3		

Plant ID	Time	Data 1
4		
4		
4		
8		
8		
4		

データ格納イメージ（パーティション）

Worker 1

Plant ID	Time	Data 1
1	2022-06	
1	2022-06	
5	2022-06	
5	2022-06	
1	2022-06	
5	2022-06	

Partition 2022-06

Plant ID	Time	Data 1
1	2022-05	
1	2022-05	
5	2022-05	
5	2022-05	
1	2022-05	
5	2022-05	

Partition 2022-05

Plant ID	Time	Data 1
1	2022-04	
1	2022-04	
5	2022-04	
5	2022-04	
1	2022-04	
5	2022-04	

Partition 2022-04

Plant ID	Time	Data 1
1	2022-03	
1	2022-03	
5	2022-03	
5	2022-03	
1	2022-03	
5	2022-03	

Partition 2022-03

データ格納イメージ（カラムナーストレージ）

Worker 1

Partition 2022-06

Plant ID	Time	Data 1
1	2022-06	
1	2022-06	
5	2022-06	
5	2022-06	
1	2022-06	
5	2022-06	

Partition 2022-05

Plant ID	Time	Data 1
1	2022-05	
1	2022-05	
5	2022-05	
5	2022-05	
1	2022-05	
5	2022-05	

…

Partition 2015-12

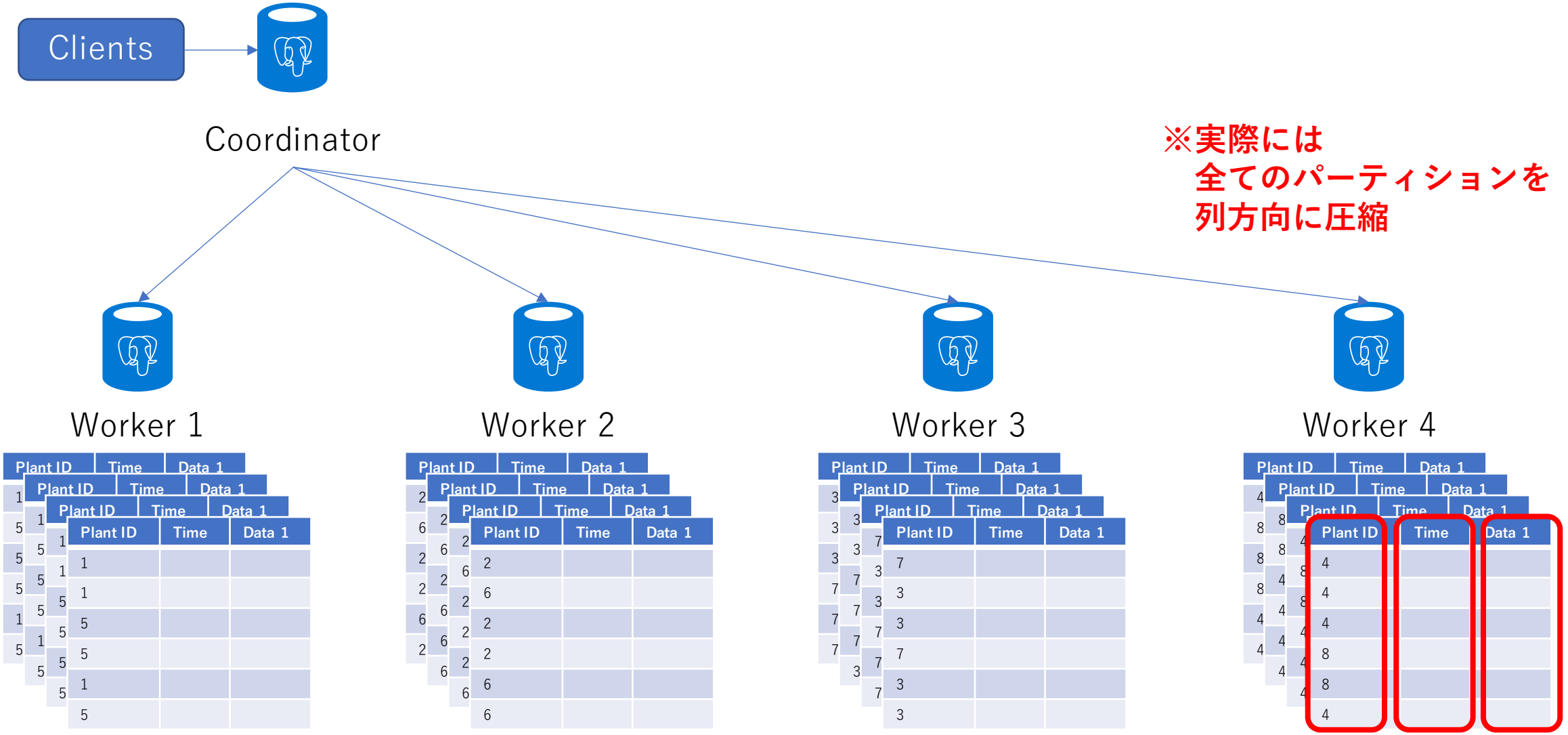
Plant ID	Time	Data 1
1	2015-12	
1	2015-12	
5	2015-12	
5	2015-12	
1	2015-12	
5	2015-12	

Partition 2015-11

Plant ID	Time	Data 1
1	2015-11	
1	2015-11	
5	2015-11	
5	2015-11	
1	2015-11	
5	2015-11	

列方向に圧縮

データ格納イメージ（全体）



Node capacity

Worker nodes

Worker node count	2 – 20*1
vCores per worker node	4, 8, 16, 32, 64, 96*2
Storage per worker node (TB)	0.5, 1, 2, 4, 8, 16
IOPS	2300, 5000, 7500

Expand your server group and scale your database by adding worker nodes.

Select up to 96 vCores with 8 GB RAM per vCore and up to 16 TB of storage with up to 7500 IOPS per node

Coordinator node

vCores per cordinator node	4, 8, 16, 32, 64, 96*2
Storage per worker node (TB)	0.5, 1, 2, 4, 8, 16
IOPS	2300, 5000, 7500

Configure your coordinator node performance by selecting CPU vCore and storage capacity.

Select up to 96 vCores with 4 GB RAM per vCore and up to 16 TB of storage with up to 7500 IOPS.

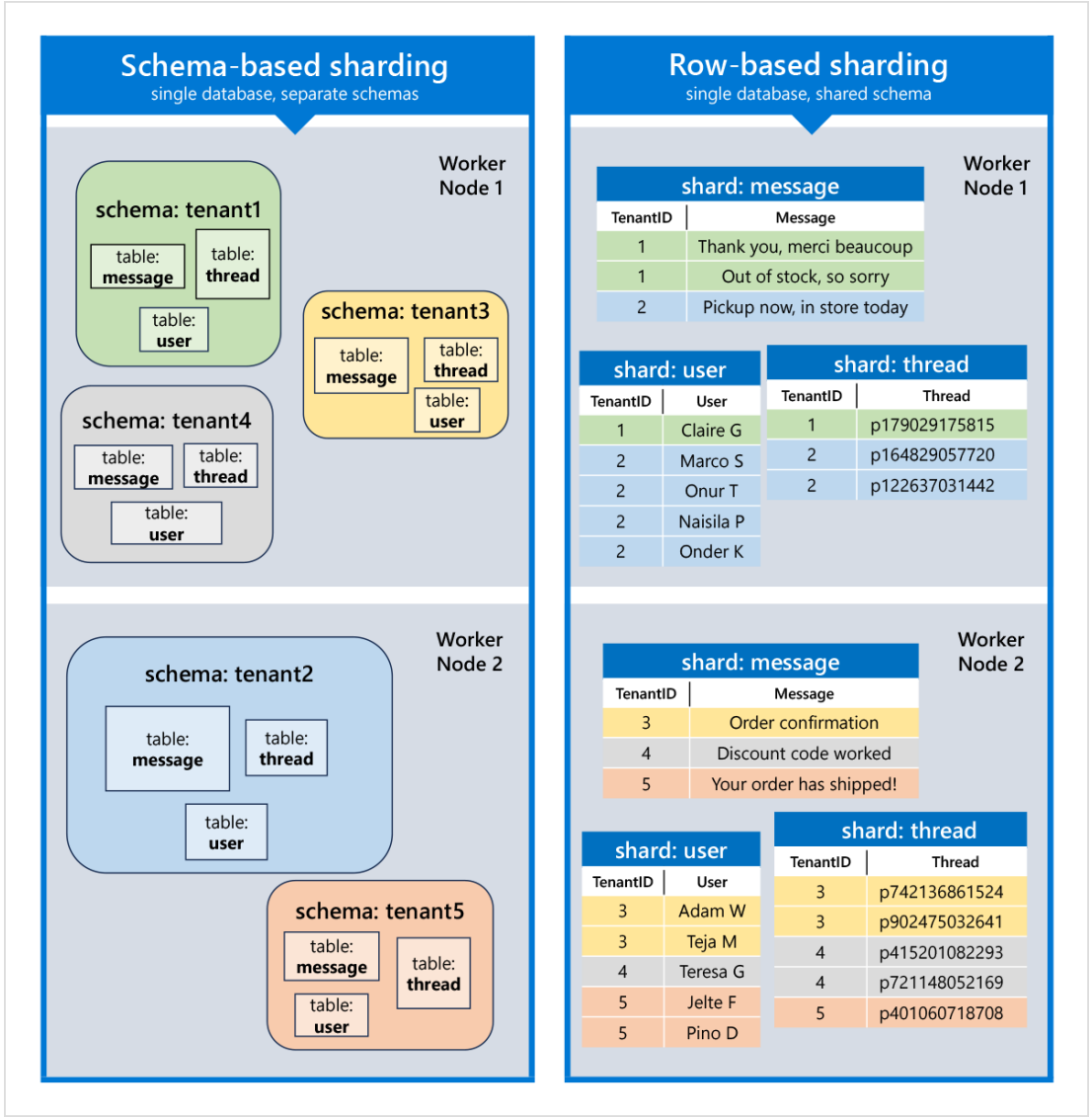
*1 サポートリクエストに応じて使用可能なワーカーノードの数を増やすことが可能
*2 リージョンによって104 vCPUまで可能

Schema-based Sharding (Citus 12)

テーブル定義を変更せず、スキーマ（テナント）単位でシャーディングをする機能。

以下のようなシナリオで有効。

- マルチテナントのSaaSアプリケーション
- 単一のデータベースを利用するマイクロサービス
- テーブル群で垂直分割する



Schema-based Sharding (Citus 12) – 続き

Row-basedとの比較

	Shema-based sharding	Row-based sharding
マルチテナントモデル	テナント毎にスキーマを分割	テナントID列でテーブルを共有
Citusのバージョン	12.0以降	全バージョン
オリジナルPostgreSQLに対する追加手順	なし、設定変更のみ	create_distributed_table()を用いて各テーブルを分散し、テナントIDでテーブル同士をco-locate
テナント数	1-1万	100-100万以上
データモデリングの要件	分散スキーマ間での外部キーは利用不可	テナントID列を含み、各テーブルの分散列として利用する。プライマリキー、外部キーとしても利用可
シングルノードのクエリのSQL要件	クエリー毎に単一の分散スキーマを利用	JOINとWHERE句にテナントID列を含む必要あり
テナントを跨ぐ並列クエリ	No	Yes
テナント固有のテーブル定義	Yes	No
アクセスコントロール	スキーマパーミッション	行レベルセキュリティ
テナント間でのデータ共有	参照テーブルを利用（スキーマを跨ぐ）	参照テーブルを利用
シャード分離	定義によって各テナントがシャードグループを保持	保持するシャードグループを特定のテナントに設定することが可能