AWS

SUMMIT

Amazon Aurora PostgreSQL

Mark Porter, General Manager

Amazon Aurora PostgreSQL
Amazon RDS for PostgreSQL
Amazon RDS Platform
Amazon RDS Operations

markpor@amazon.com

May 31, 2017

amazon web services

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

本日のアジェンダ

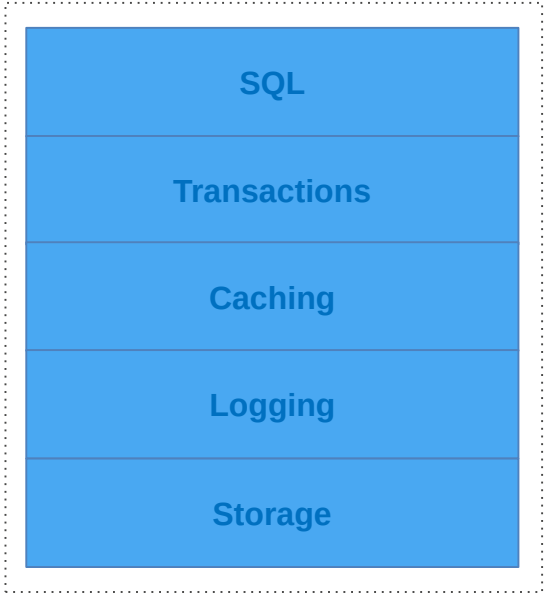
- なぜ Amazon Aurora を作ったのか？
- なぜ PostgreSQL 互換を追加したのか？
- 堅牢性と可用性のアーキテクチャ
- PostgreSQLと比較したパフォーマンス結果
- パフォーマンスアーキテクチャ
- *Performance Insights*
- ロードマップ
- Q&A

Amazon Aurora

+

PostgreSQL

伝統的なリレーショナルデータベースはスケールが難しい

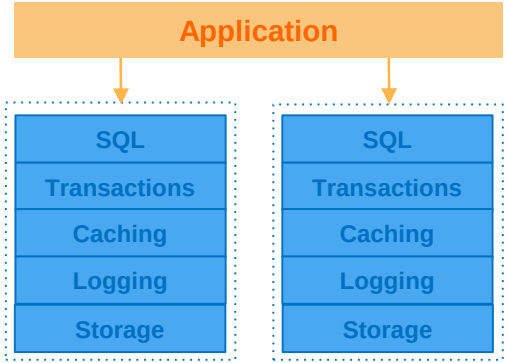


複数の機能レイヤーが
1つのモノリシックな
スタックに入っている

データベースをスケールさせる従来のアプローチ

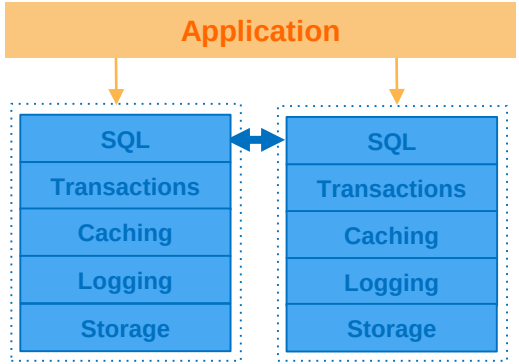
Sharding

アプリケーションレイヤで
統合



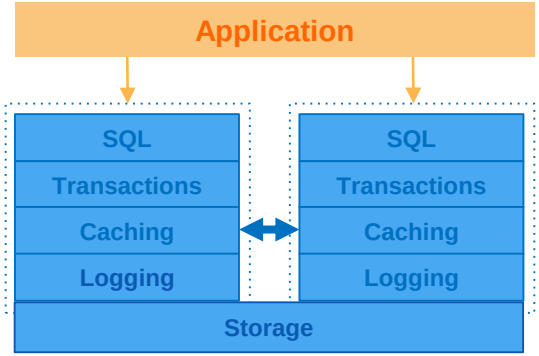
Shared Nothing

SQL レイヤで
統合



Shared Disk

キャッシング、ストレージレイヤ
で統合



どのアーキテクチャもモノリシックな考え方に縛られている

リレーショナルデータベースをもう一度考える

今、データベースを改めて実装するならどうするか？

- スタックを分割する

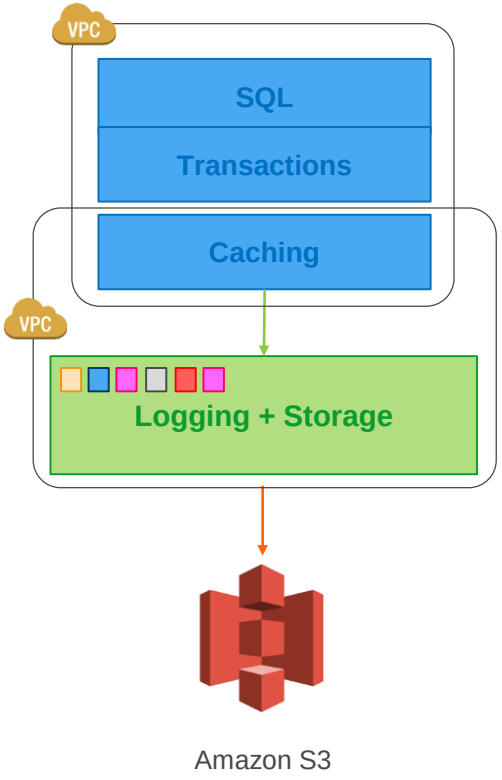
- 以下のようなことを実装するでしょう:
  - ✓ スケールアウトできる…
  - ✓ 自律的に修復できる…
  - ✓ 分散サービスを活用する…

Service Oriented Architecture をデータベースに当てはめる

1 マルチテナントでスケールアウト可能、データベースに最適化されたストレージサービスをロギング、ストレージレイヤとして採用

2 Amazon EC2, Amazon VPC, Amazon DynamoDB, Amazon SWF, Amazon Route 53 といった 他のAWS のサービスを管理・監視のために利用

3 Amazon RDS プラットフォームを利用し、マネージドサービスとして実現。運用・管理機能をオフロード



Amazon RDS

Amazon DynamoDB

Amazon SWF

Amazon Route 53

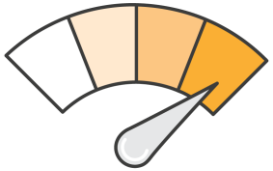
Amazon Aurora は進化し続けています

2014年 Amazon Aurora with MySQL compatibility リリース

このたび PostgreSQL compatibility を追加しました

商用データベースのパフォーマンスと可用性、オープンソースデータベースのシンプルさとコスト効率性を備えた Amazon のクラウドに最適化されたリレーショナルデータベースとして選択可能に

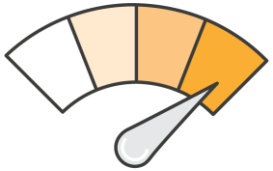
お客様の声を元に – なぜ PostgreSQL を追加したのか?



Amazon Aurora

MySQL

お客様の声を元に – なぜ PostgreSQL を追加したのか?



Amazon Aurora

PostgreSQL

PostgreSQL の概要

- オープンソースデータベース
- 20年間にわたる開発の歴史
- 単一の企業ではなく、コミュニティによる所有
- 寛大で革新に馴染みやすいオープンソースライセンス

PostgreSQL

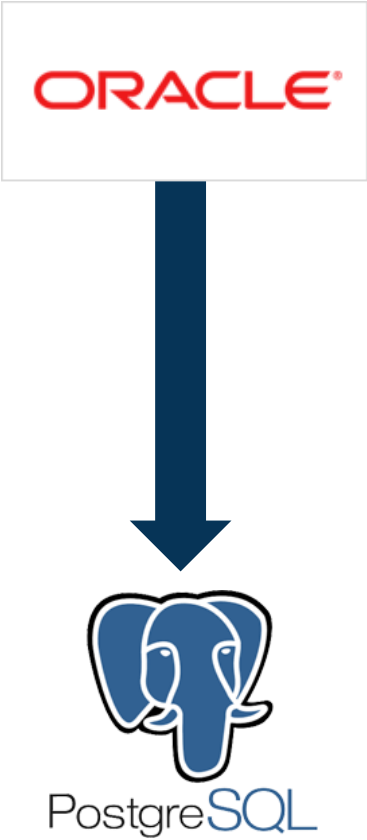
Open Source Initiative

PostgreSQL の概要

- そのままで高性能
- オブジェクト指向やANSI標準SQL:2008との互換性
- どのオープンソースデータベースよりも豊富な機能を持つ時空間(geospatial)機能
- 12 言語でのストアドプロシージャサポート (Java, Perl, Python, Ruby, Tcl, C/C++, Oracleとの互換性が高い PL/pgSQLなど)

PostgreSQL の概要

- Oracle との互換性もオープンソースデータベースの中で最も高い
- AWS Schema Conversion Tool による自動変換率は、Oracle DB から PostgreSQL が最も高い



PostgreSQL compatibility が意味するものとは(1 of 2)

PostgreSQL 9.6 +クラウドに最適化された Amazon Aurora ストレージ

パフォーマンス: PostgreSQL と比較して、最大 2倍以上の性能

可用性: 30 秒未満でのフェールオーバー

堅牢性: 3つの AZ にわたり 6つの複製を保持

リードレプリカ: 15台までのレプリカに対し、ラグは一桁ミリ秒

現在、そして将来にわたってのPostgreSQL との完全な互換性

互換性レイヤーではなく、ネイティブな PostgreSQL の実装を利用

Amazon Aurora Storage

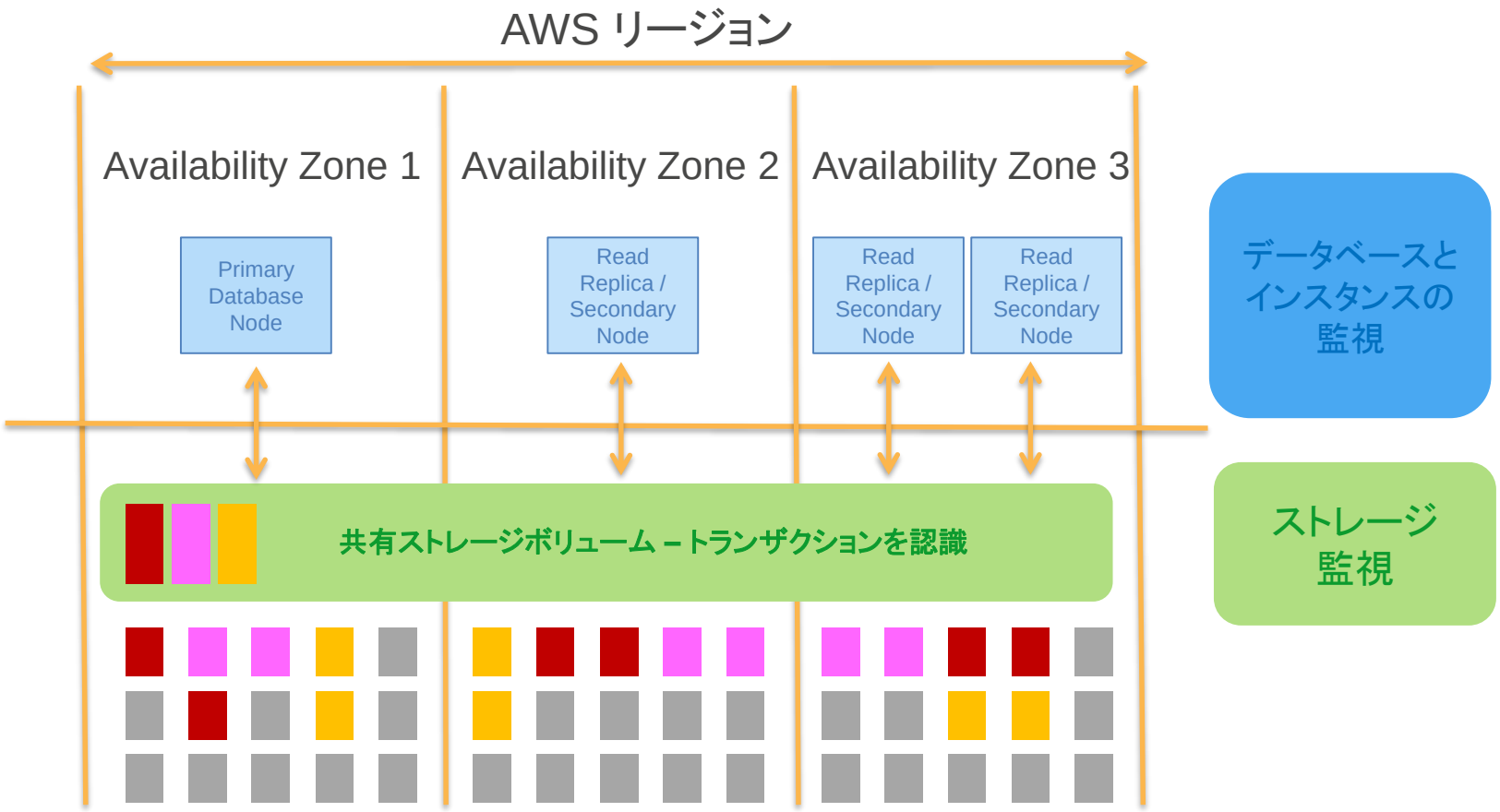
PostgreSQL compatibility が意味するものとは(2 of 2)

- クラウドネイティブなセキュリティと暗号化
  - AWS Key Management Service (KMS)
  - AWS Identity and Access Management (IAM)
- Amazon RDS による容易な運用管理
- 容易なデータロード/アンロード
  - AWS Database Migration Service
  - AWS Schema Conversion Tool
- 現在、そして将来にわたってのPostgreSQL との完全な互換性
  - 互換性のためのレイヤーがあるのではなく、ネイティブな PostgreSQL の実装をそのまま利用

Amazon Aurora with PostgreSQL Compatibility

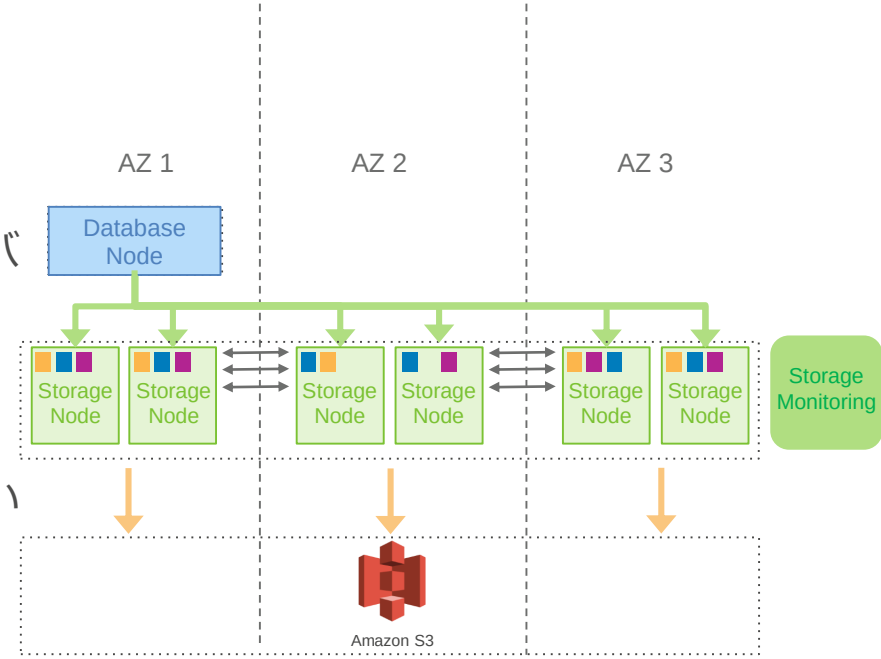
堅牢性と可用性

スケールアウト可能で分散した、ログ構造ストレージ



Amazon Aurora Storage Engine 概要

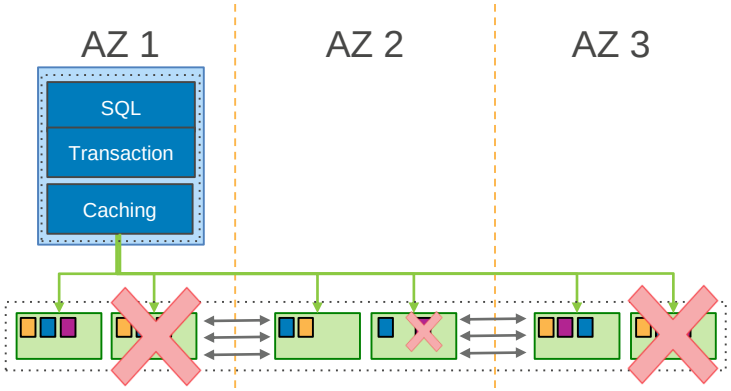
- 3 つのアベイラビリティゾーンにわたる 6つの複製
- 継続的にS3(99.999999999%の耐久性)へ増分バックアップ
- ノードとディスクに対する継続的な監視と修復
- 10GB のセグメント単位でデータを格納し、リカバリやホットスポット解消実施
- read/writeクォーラムシステム：
  レイテンシーの揺らぎに強い
- クォーラムのメンバー変更は書き込みに影響しない
- 最大 64TB まで自動的に拡張する
  ストレージボリューム

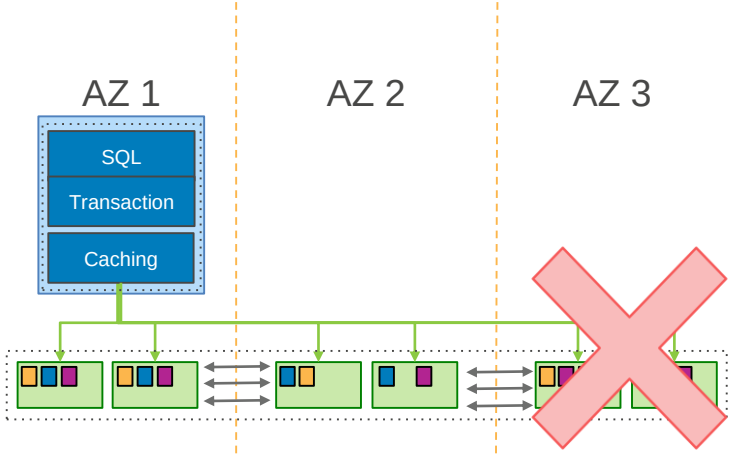


Amazon Aurora ストレージエンジンの耐障害性

何が障害を引き起こすか？
- セグメント障害(ディスク)
- ノード障害 (マシン)
- AZ障害(ネットワーク、データセンタ)

最適化
- 4/6クォーラム：書き込み
- 3/6クォーラム: 読み込み
- リカバリのための peer to peer レプリケーション





Amazon Auroraのレプリカ

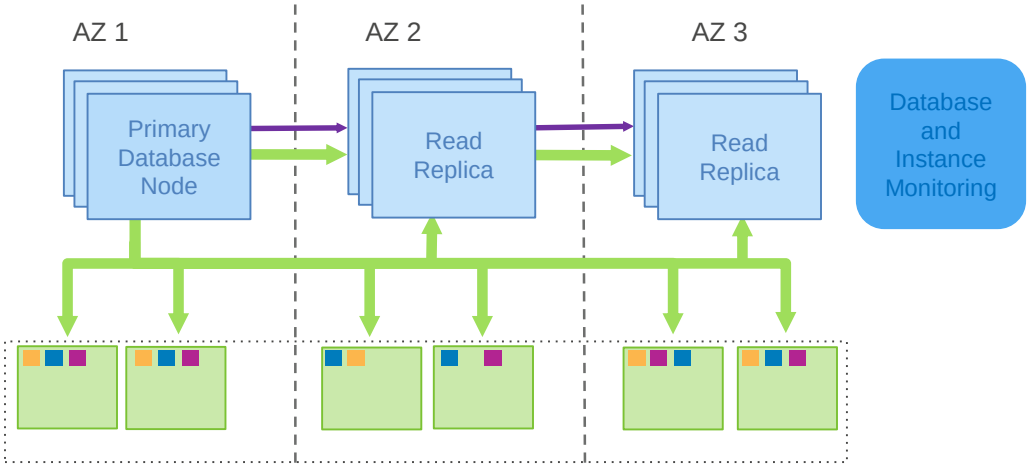
可用性

データベースノードに不具合が発生すると、自動的に検知して置換

不具合が発生したプロセスを自動的に検知し、再起動

フェイルオーバーの必要がある場合、レプリカは自動的にプライマリへ昇格

フェイルオーバー順を指定可能

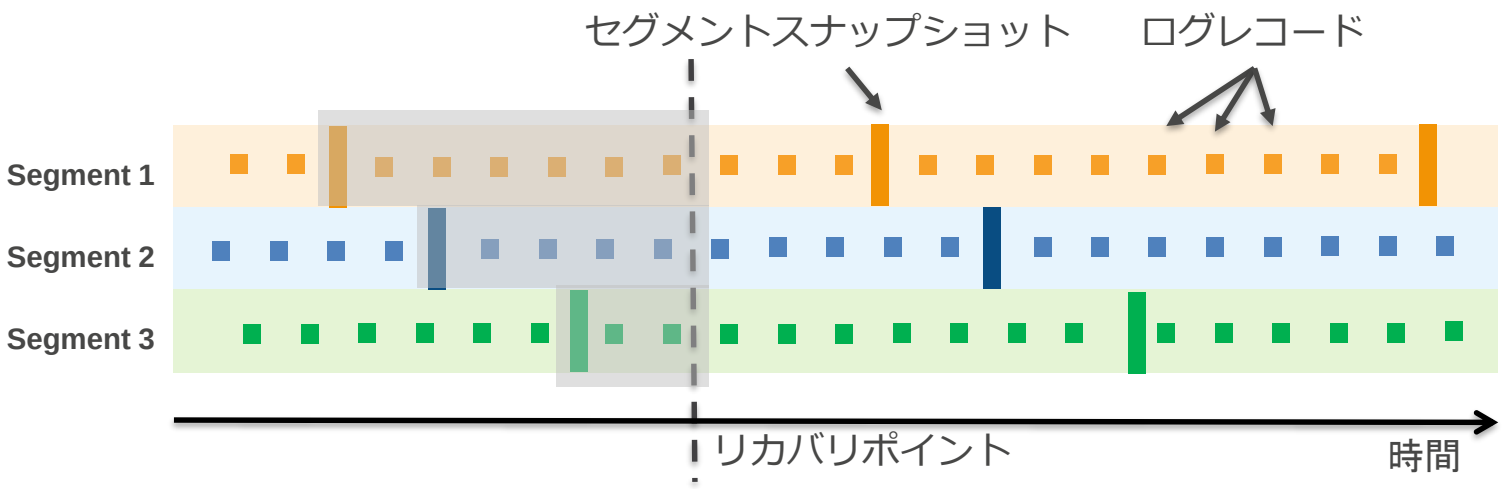


パフォーマンス

リードレプリカを利用して、読み込みのスケールアウトが可能

リードレプリカにまたがって読み込みの分散が可能

継続的なバックアップ

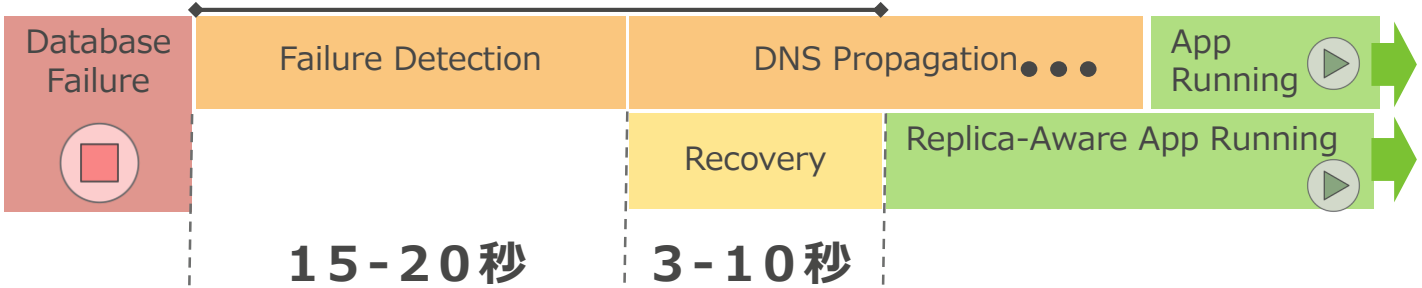


- 各セグメントごとに Amazon S3 へ定期的にスナップショットを並列に取得
- バックアップは継続的に行われ、パフォーマンスや可用性への影響はない
- リストア時、適切なセグメントのスナップショットとログストリームを取得し、ストレージへ適用
- ログストリームをセグメントスナップショットに並列非同期で適用

高速でより予測可能なフェイルオーバー時間

Amazon RDS for PostgreSQL のフェイルオーバー時間：60秒程度

Amazon Aurora のフェイルオーバー時間：30秒未満

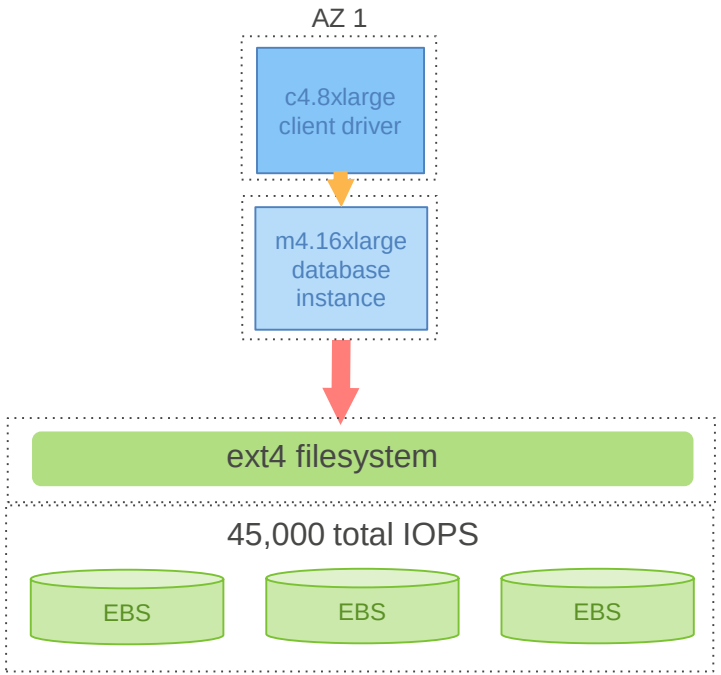


Amazon Aurora with PostgreSQL Compatibility

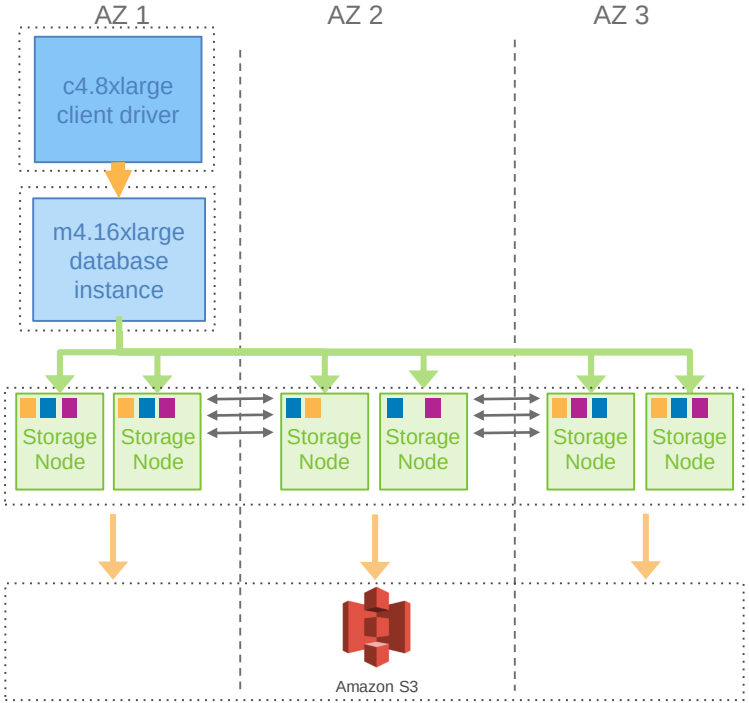
PostgreSQLとのパフォーマンス比較

ベンチマークのシステム構成

PostgreSQL

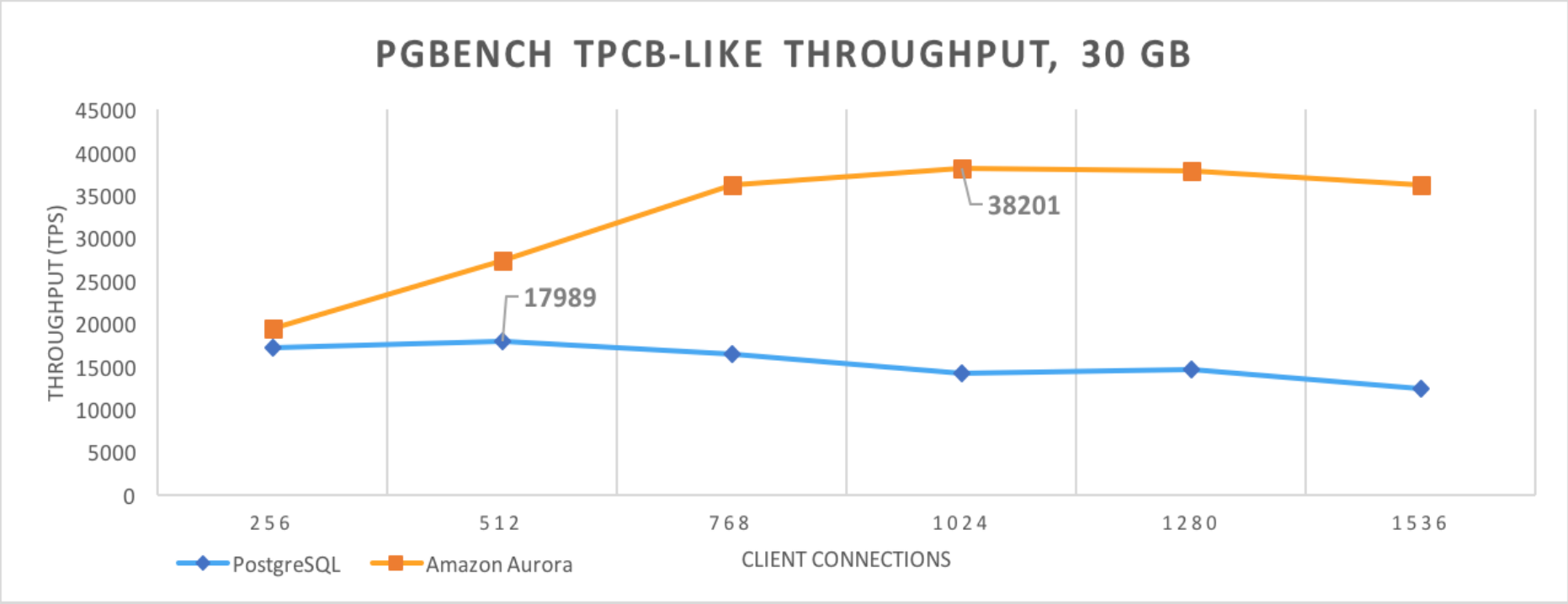


Amazon Aurora



m4.16xlarge (64 VCPU, 256GiB), c4.8xlarge (36 VCPU, 60GiB)

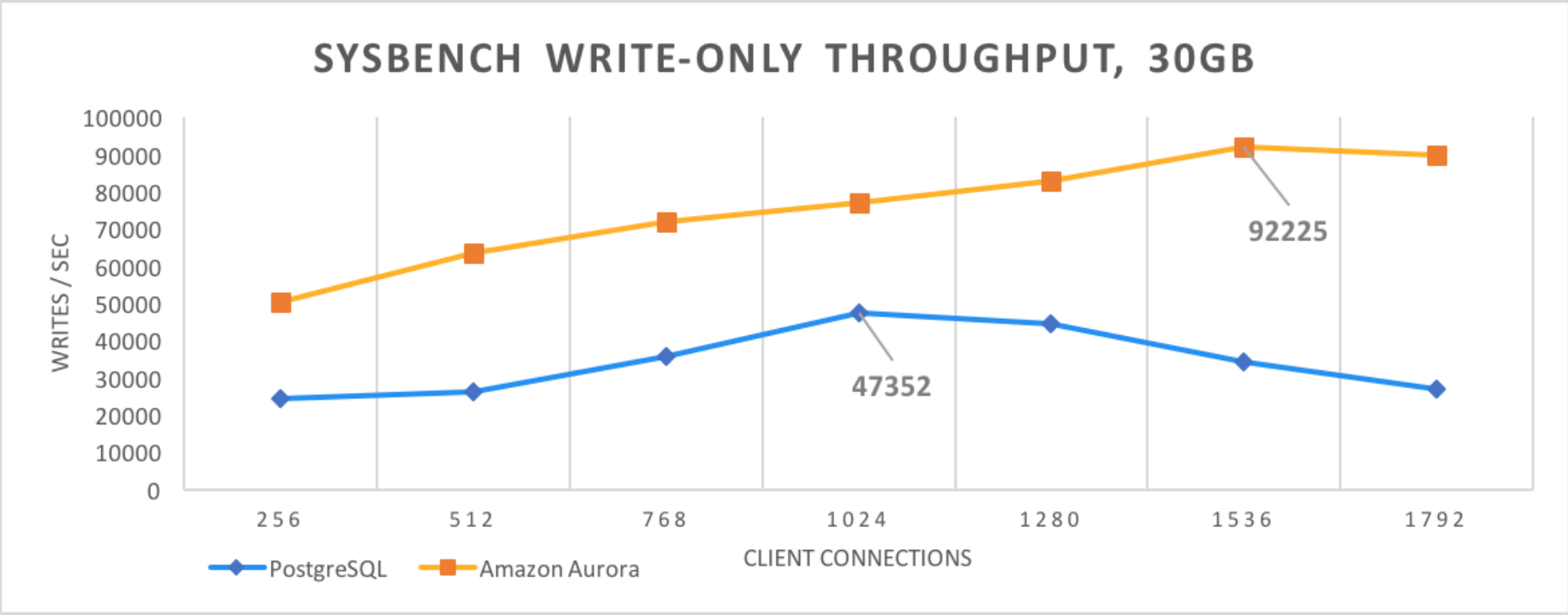
PgBenchでAmazon Auroraは２倍以上高速な結果に



pgbench “tpcb-like” workload, scale 2000 (30GiB). 60分間実行

SysBenchではAmazon Auroraが2~3倍高速な結果に

Amazon AuroraはPostgreSQLのピーク性能と比較して2倍、コネクション数が多い環境では３倍の性能



SysBench oltp(write-only) workload 30GBデータベース(250テーブル, 各テーブルに初期400,000行)

Amazon Aurora: 秒間120,000書き込み以上のパフォーマンス

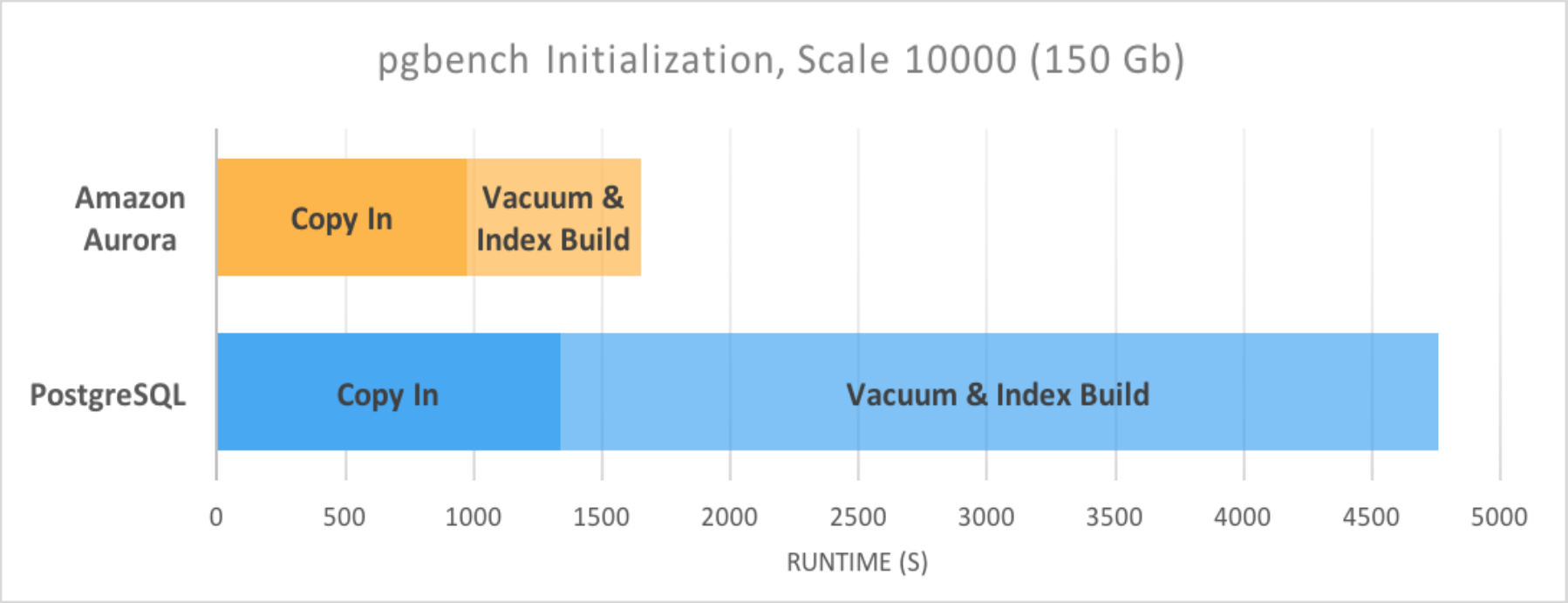
Sysbenchで安定して120K writes/secの性能

```
OLTP test statistics:
    queries performed:
        read:                                0
        write:                               432772903
        other:(begin + commit)               216366749
        total:                               649139652
    transactions:                            108163671 (30044.73 per sec.)
    read/write requests:                     432772903 (120211.75 per sec.)
    other operations:                        216366749 (60100.40 per sec.)
    ignored errors:                          39407  (10.95 per sec.)
    reconnects:                              0      (0.00 per sec.)
```

sysbench write-only 10GB workload 250テーブル　各テーブルに初期25,00行. 10分 warmup, 3,076クライアント
key constraint errorsは考慮しない設定を使用

データのロードでも３倍高速

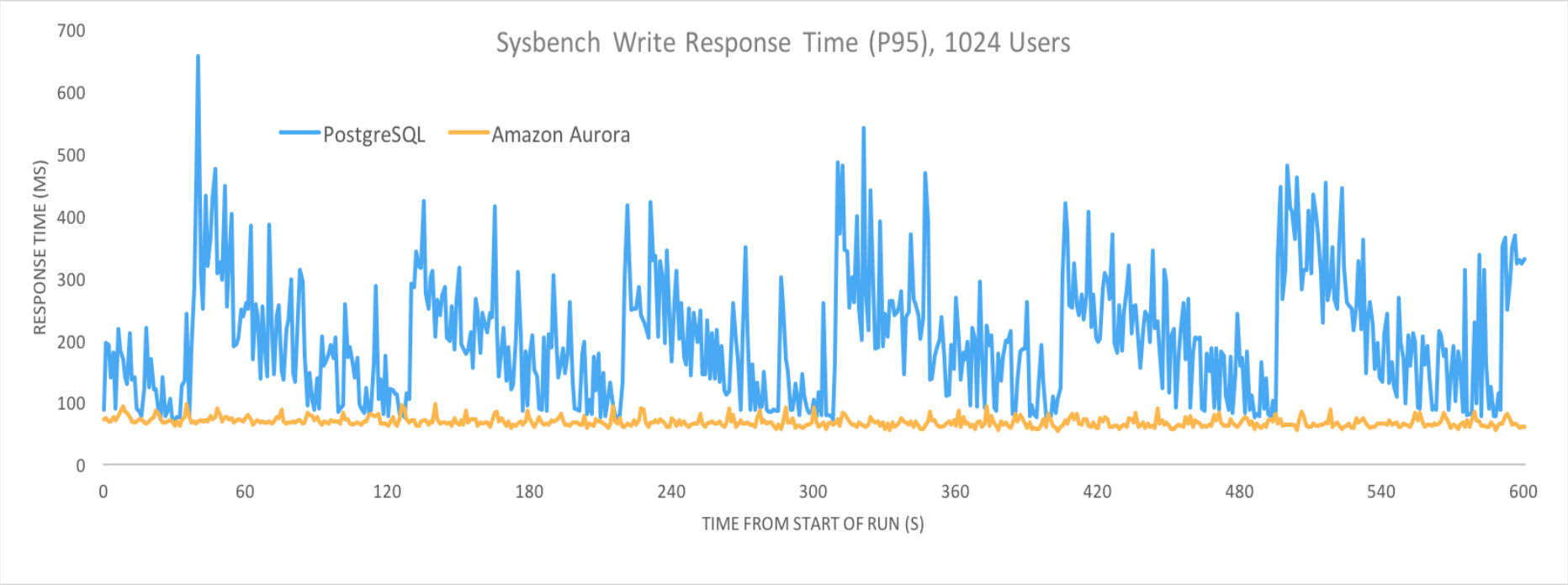
PgBenchを利用した際のデータベースの初期化作業が
PostgreSQLと比較して３倍高速



Command: `pgbench -i -s 2000 -F 90`

Amazon Auroraはレスポンスタイムが２倍以上高速

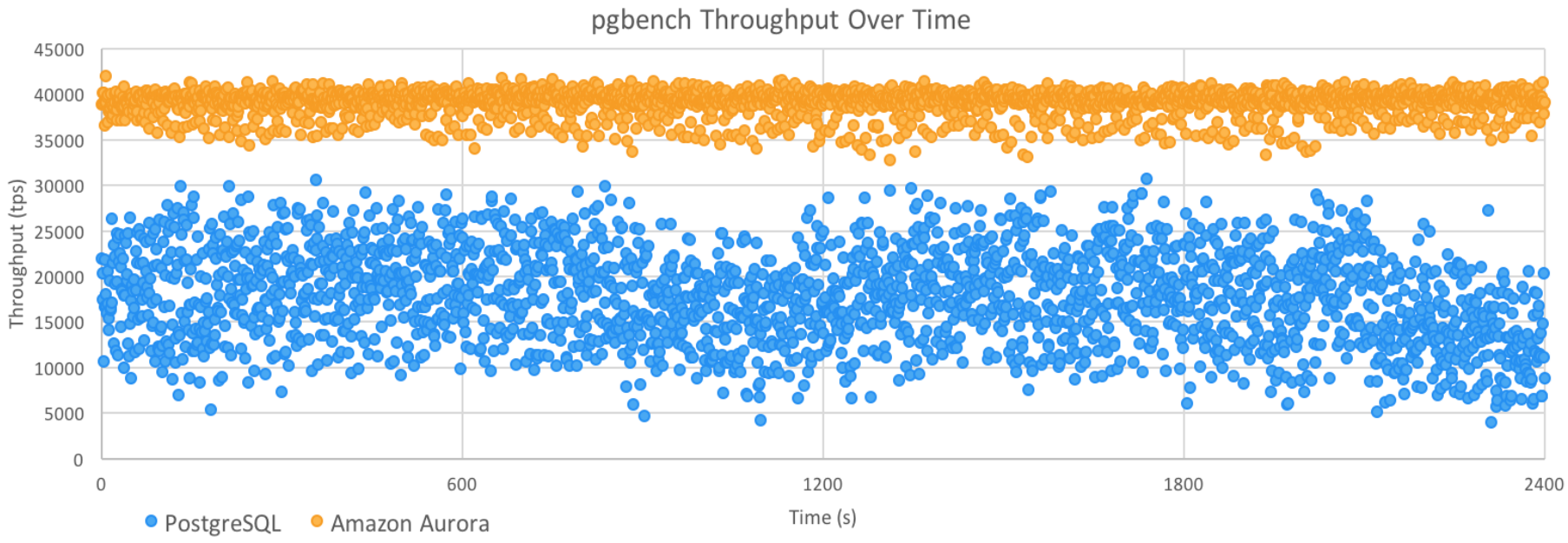
書込みが多いワークロードで、PostgreSQLと比較してレスポンスタイムが2倍以上高速 (10倍以上安定したレスポンスタイム)



SysBench oltp(write-only) 23GiB workload　250テーブル 各テーブルに初期300,000行. 10分 warmup.

Amazon Auroraは安定したスループットを発揮

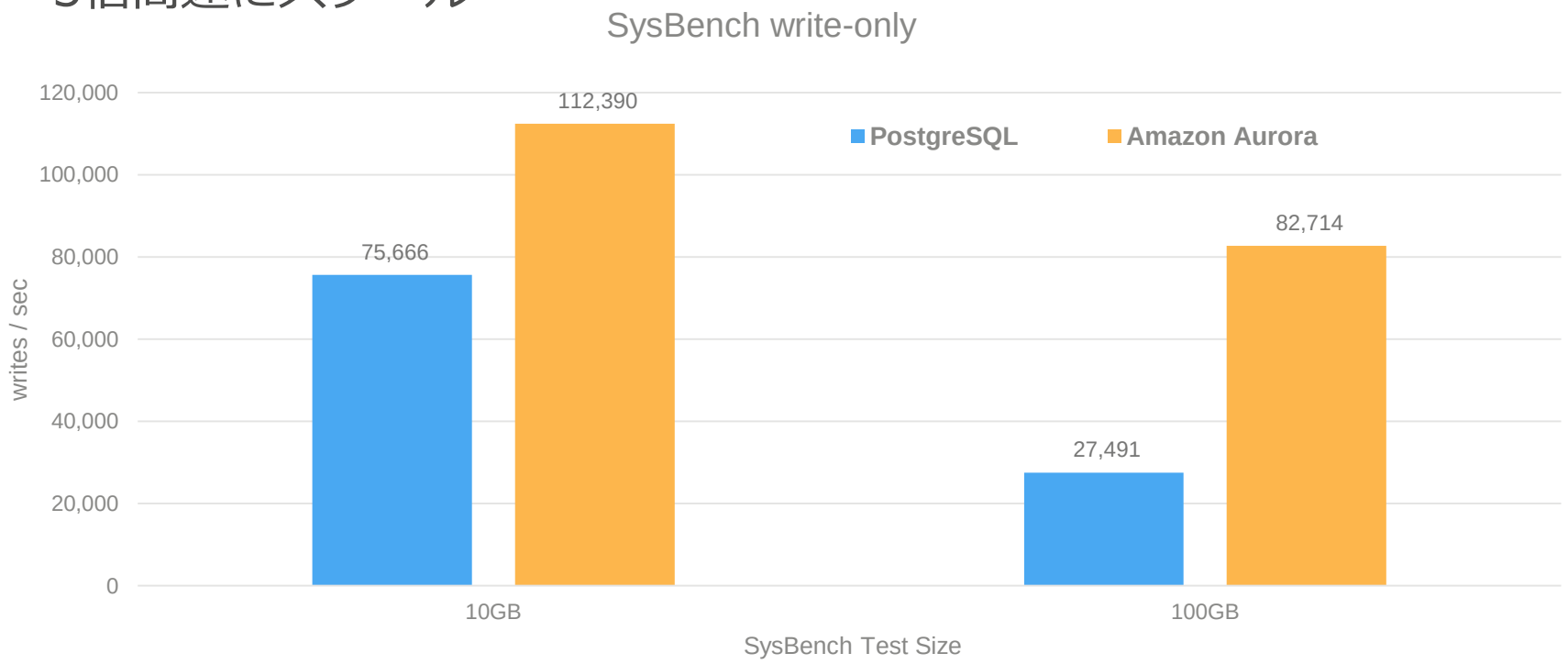
負荷をかけている間、パフォーマンスは３倍以上
PostgreSQLよりも安定したスループット



PgBench “tpcb-like” workload (scale ,2000). Amazon Aurora (1,280クライアント). PostgreSQL (512クライアント). (最良のスループットを出した並列度)

データベースサイズが大きいケースで3倍高速な結果に

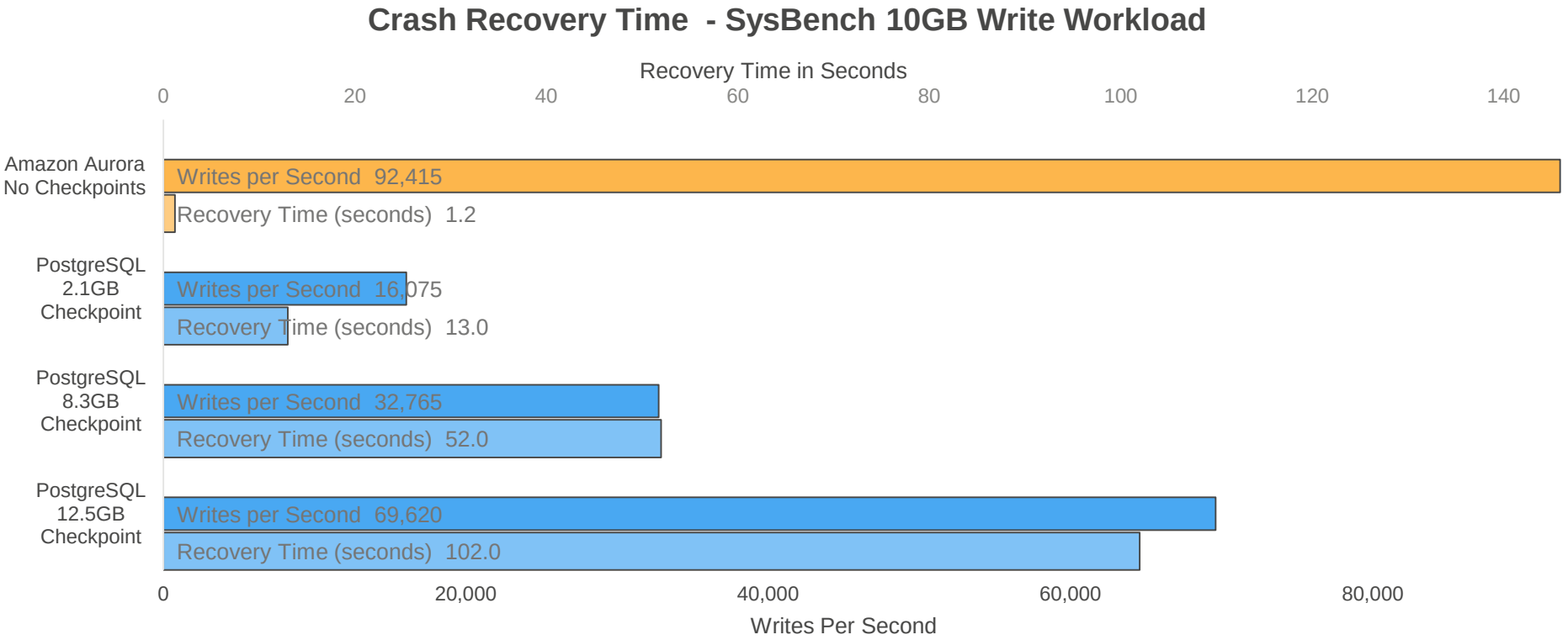
データベースサイズが 10GiB から 100 GiB になった際には、
1.5～3倍高速にスケール



SysBench oltp(write-only) – 10GiB 250テーブル 150,000行　/ 100GiB 250テーブル 1,500,000行

最大85倍高速なリカバリ

ログストラクチャード・ストレージシステムにより**ほぼ即座に復旧**



SysBench oltp(write-only) 10GiB workload 250テーブル 150,000行

Amazon Aurora with PostgreSQL Compatibility 性能測定のまとめ

計測項目	結果
PgBench	>= 2x 高速
SysBench	2x-3x 高速
データロード	3x 高速
レスポンスタイム	>2x 高速
スループットの性能ゆらぎ	>3x より安定
大規模でのスループット	3x 高速
リカバリー速度	最大 85x 高速

Amazon Aurora with PostgreSQL Compatibility

パフォーマンス アーキテクチャ

性能向上のために行っていること

DO LESS WORK

I/Oを減らす

ネットワークパケットを最小限にする

結果をキャッシュしておく

データベースエンジンをオフロードする

BE MORE EFFICIENT

非同期で処理する

レイテンシーの通り道を減らす

ロックフリーなデータ構造を使う

バッチ操作を同時に行う

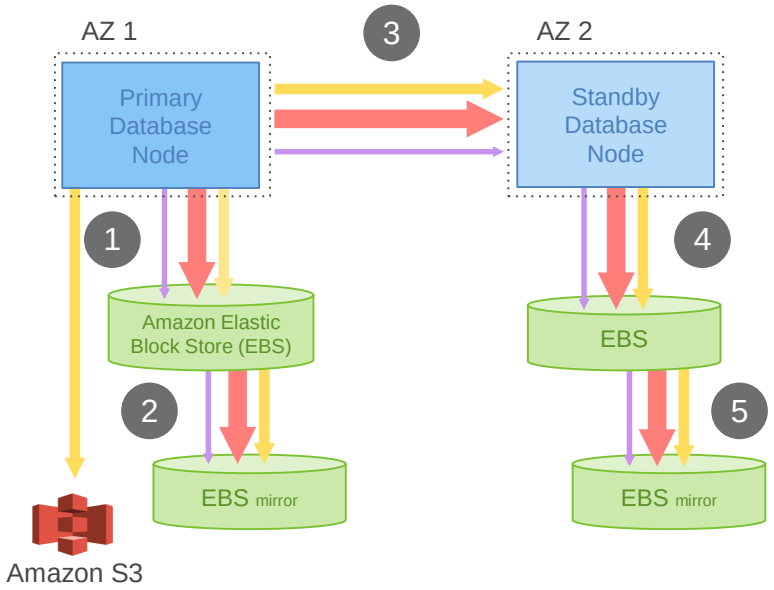
データベースは I/O が全て

ネットワーク接続したストレージは PACKETS/SECOND が全て

高スループットの処理には CPUとメモリの最適化 が必要

Amazon RDS for PostgreSQLのwrite I/O

RDS FOR POSTGRESQL WITH MULTI-AZ



IO FLOW

Amazon EBSに書き込み発行後、EBSがミラーへ複製
両方終了後ackを返す
書き込みをスタンバイインスタンスへ伝播
スタンバイインスタンス側のEBSに書き込み発行

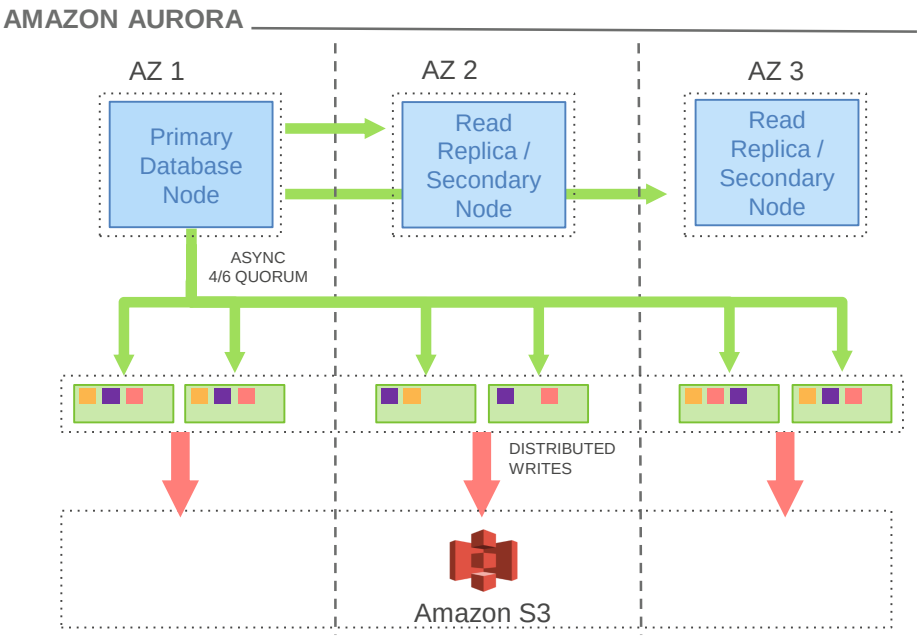
OBSERVATIONS

ステップ1, 3, 5はシーケンシャルかつ同期
それによりレイテンシーもパフォーマンスのゆらぎも増加
各ユーザー操作には様々な書き込みタイプがある

TYPE OF WRITE

WAL　DATA　COMMIT LOG & FILES

Auroraのwrite I/Oトラフィック (データベースノード)



IO FLOW

REDOログレコードをまとめる – 完全にLSN順に並ぶ
適切なセグメントに分割する – 部分ごとに並ぶ
ストレージノードへまとめて書き込む

OBSERVATIONS

WALレコードのみ書き込む; 全てのステップは非同期
データブロックは書き込みはない(チェックポイント, キャッシュ置換時)
6倍のログ書き込みだが, **1/9の**ネットワークトラフィック
ネットワークやストレージのレイテンシー異常時の耐性がある

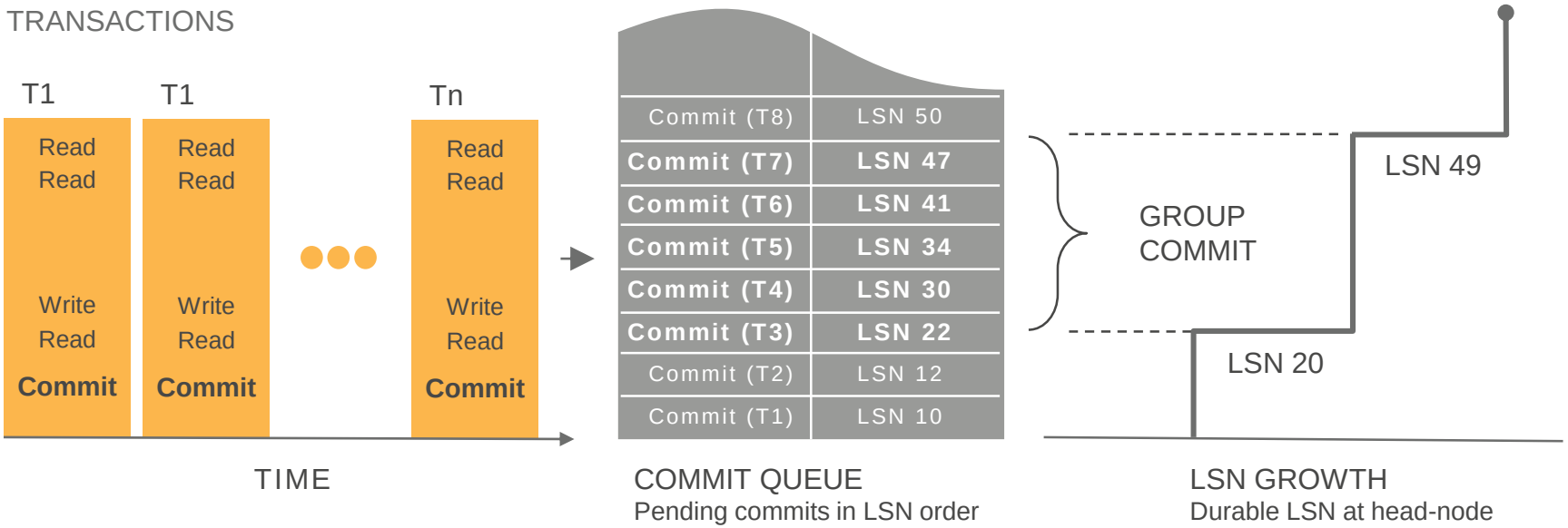
PERFORMANCE

write-only もしくは、read/write が混在するワークロードにおいて、PostgreSQL のコミュニティエディションに比べて、2倍以上の性能を発揮

TYPE OF WRITE

AMAZON AURORA + WAL LOG | WAL | DATA | COMMIT LOG & FILES

非同期グループコミット



TRADITIONAL APPROACH

ログレコードをディスクへ書き込むためのバッファを管理

バッファが一杯になるか一定の待ち時間を超過すると書き込みを実行

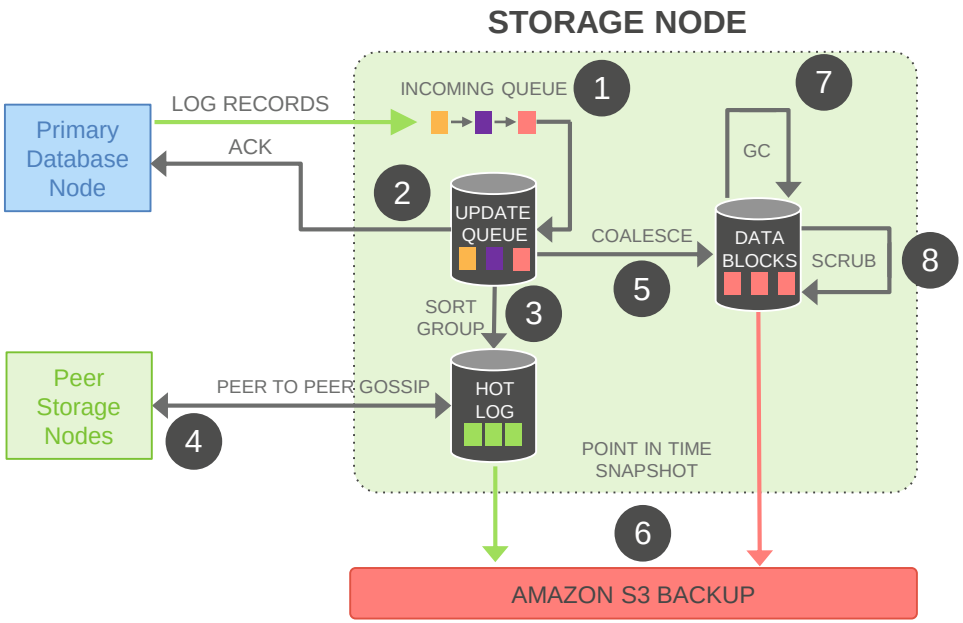
書き込み頻度が少ない場合は最初の書き込みが遅くなる

AMAZON AURORA

最初の書き込みと同時にI/Oリクエストを実行。書き込みが実行されるまでバッファを埋める

6つの内4つのストレージノードからACKが返ってきた時点で堅牢性のある書き込みが完了

AuroraのI/Oトラフィック (ストレージノード)



IO FLOW

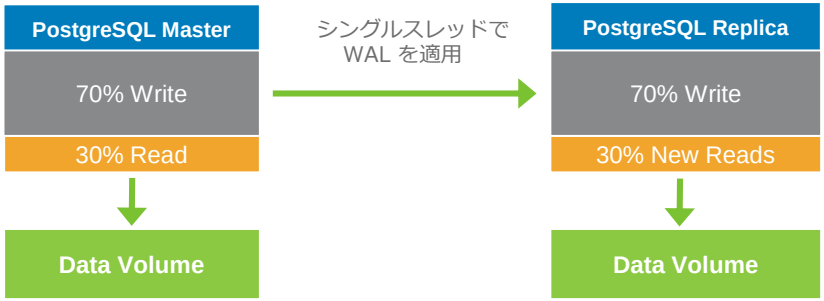
① レコードを受信しインメモリのキューに追加
② レコードを永続化してACK
③ レコードを整理してログのギャップを把握
④ ピアとゴシップ通信して穴埋め
⑤ ログレコードを新しいバージョンのデータブロックに合体
⑥ 定期的にログと新しいバージョンのブロックをS3に転送
⑦ 定期的に古いバージョンのガベージコレクションを実施
⑧ 定期的にブロックのCRCを検証

OBSERVATIONS

全てのステップは非同期
ステップ1と 2だけがフォアグラウンドのレイテンシーに影響
インプットキューはPostgreSQLに比べて**極めて小さい**
レイテンシーにセンシティブな操作に向く
ディスク領域をバッファーに使ってスパイクに対処

AuroraレプリカのI/Oトラフィック

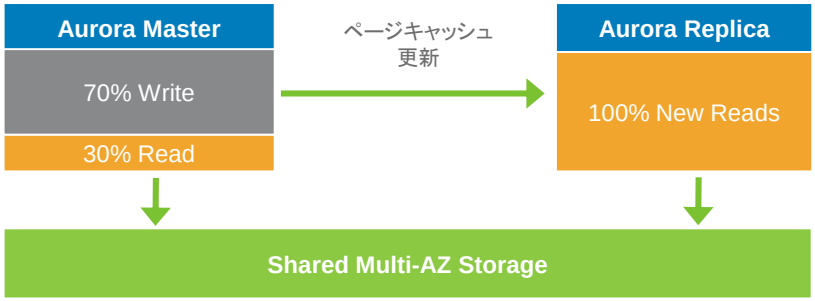
POSTGRESQL READ SCALING



Physical: WAL をレプリカに送信

- 書き込み負荷は両インスタンスとも同様
- ストレージはそれぞれ独立
  (Shared Nothing)

AMAZON AURORA READ SCALING



Physical: WAL をマスターからレプリカに送信

- ストレージは共有。レプリカ側でストレージへの書き込みは起こらない
- キャッシュされたページにWALを適用
- 全てのコミットが参照可能になったら、read viewを進める

サバイバル可能なキャシュでアプリケーションの再開を高速に

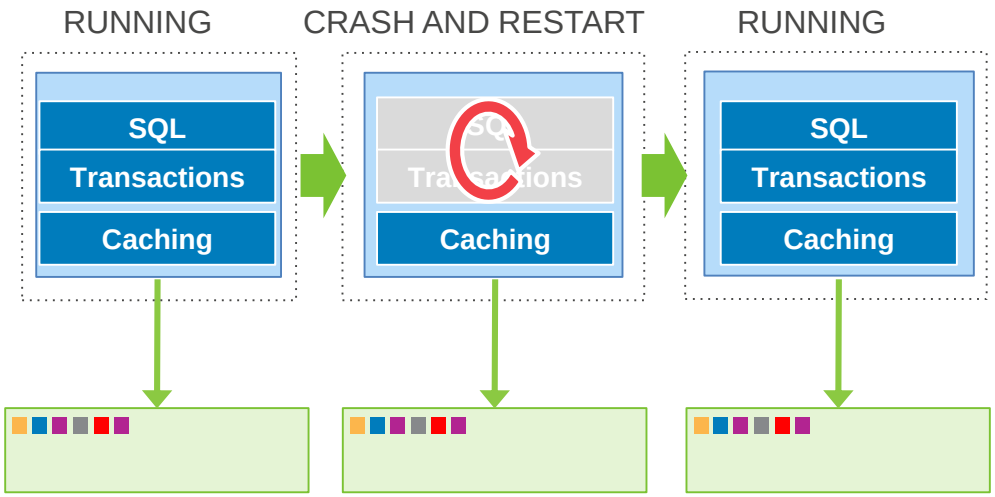
キャッシュレイヤは通常、データベースプロセス内にある
→データベースがダウンした場合、消去される

Auroraはキャッシュをデータベースプロセス外に移動

データベースプロセスのリスタートが発生してもキャッシュが残った状態を維持可能

データベースが定常のパフォーマンスを発揮できる状態にいち早く戻すことが出来る

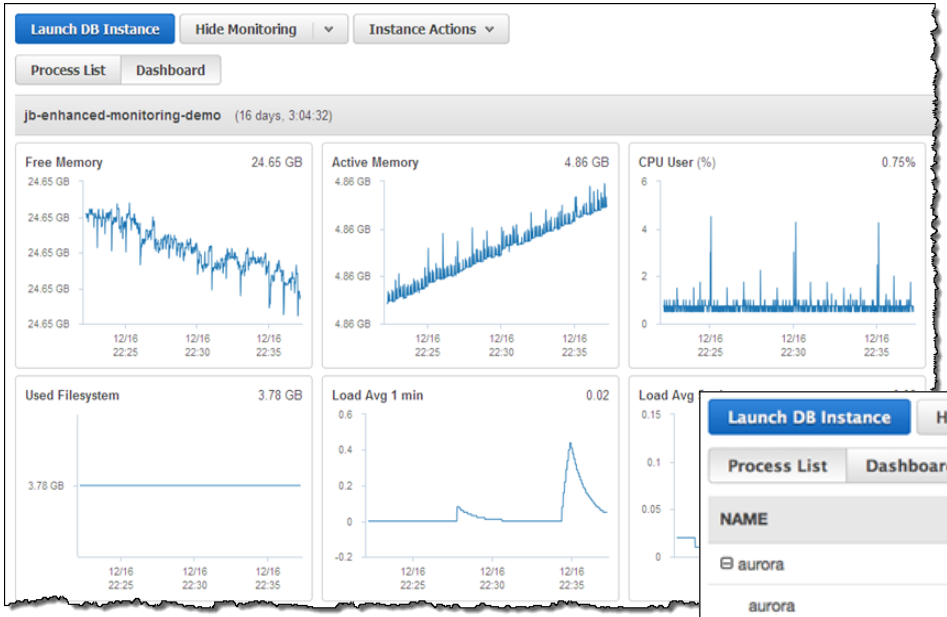
キャッシュはデータベースプロセス外で管理されているため、データベースプロセスのリスタートが発生してもキャッシュを維持可能



Amazon Aurora with PostgreSQL Compatibility

Performance Insightsのご紹介

第1弾: 拡張モニタリング

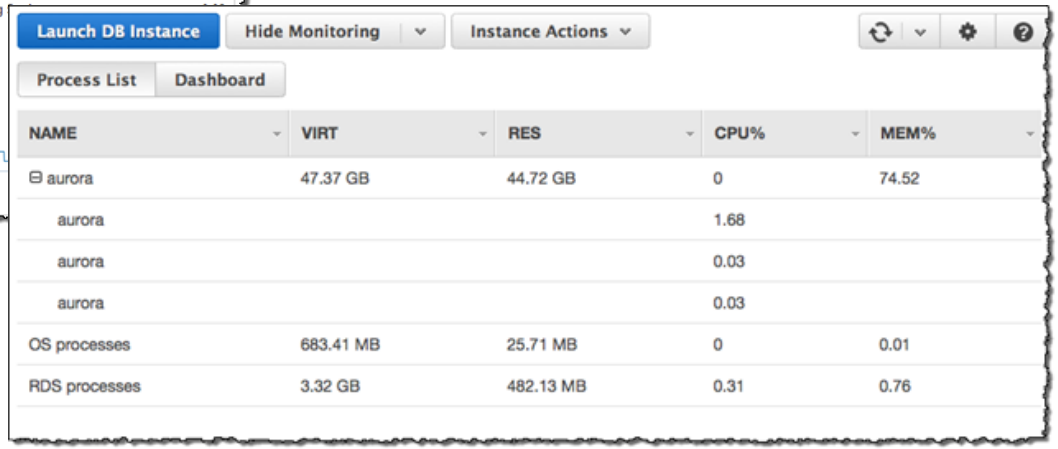


2016年リリース

O/Sメトリクス

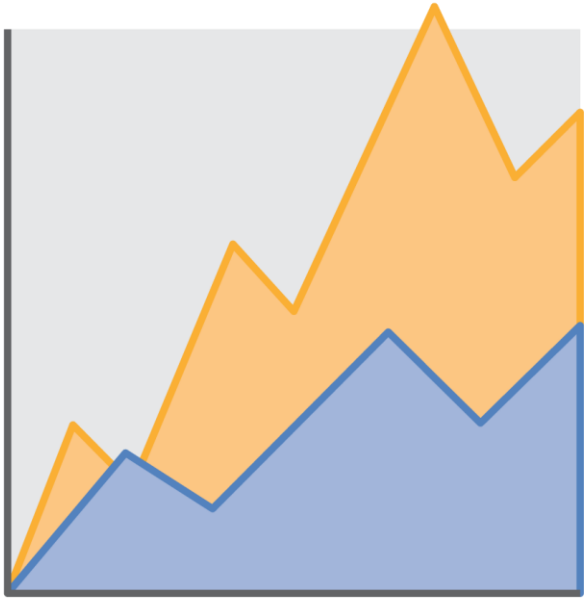
プロセス＆スレッド一覧

最小1秒単位の粒度



NAME	VIRT	RES	CPU%	MEM%
⊖ aurora	47.37 GB	44.72 GB	0	74.52
aurora			1.68	
aurora			0.03	
aurora			0.03	
OS processes	683.41 MB	25.71 MB	0	0.01
RDS processes	3.32 GB	482.13 MB	0.31	0.76

第2弾: Performance Insights



データベースエンジンの
パフォーマンス・チューニング

AWS Services RDS DMS EMR Edit

new

RDS Dashboard

- Instances
- Clusters
- Reserved Purchases
- Snapshots
- Security Groups
- Parameter Groups
- External Licenses
- Option Groups
- Subnet Groups
- Events
- Event Subscriptions

Notifications 18

Launch DB Instance | Show Monitoring | Instance Actions

Filter: All Instances | Search DB Instances... | Viewing 100 of 100 DB Instances

Engine	DB Instance	Status	CPU	DB Load	Maintenance	Class	VP
MySQL	ashkma-mysql51	available	0.25%		None	db.m3.xlarge	vp
MySQL	test-piranha	available	0.21%	Enable	None	db.m3.xlarge	vp
Aurora	wenliant-test-replica	available	4.08%	N/A	None	db.r3.large	de
Aurora	test-restore-piranha	available	2.00%	N/A	None	db.r3.xlarge	dn
Aurora	test-enhance1	available	4.5%		None	db.r3.large	dn
Aurora	test-enhance1-us-east-1b	available	3.83%	Enable	None	db.r3.large	dn
Aurora	test-aurora	available	4.25%		None	db.r3.large	dn
Aurora	test-aurora-us-east-1c	available	3.67%		None	db.r3.large	dn
Aurora	restoretest1	available	4.00%	N/A	None	db.r3.large	de
Aurora	restore-non-non	available	4.00%	Enable	None	db.r3.large	dn
Aurora	restore-non-en	available	4.00%	N/A	None	db.r3.large	dn
Aurora	restore-en-en	available	4.00%		None	db.r3.large	dn
Aurora	nktestaurora-0413	available	4.42%	N/A	Available	db.r3.large	dn
Aurora	nktestaurora-0413-us-east-1c	available	3.83%		Available	db.r3.large	dn
Aurora	jeffrugg-dms-aurora-src-01-us-east-1c	available	4.00%	Enable	None	db.r3.large	dn
Aurora	hotfix-movetovpc-au	available	4.10%	N/A	None	db.r3.large	dn

Feedback English

© 2008 - 2016, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Privacy Policy Terms of Use

new

AWS | Services | RDS | DMS | EMR | Edit | eichbaum @ awsdbux | N. Virginia

RDS Dashboard
- Instances
- Clusters
- Reserved Purchases
- Snapshots
- Security Groups
- Parameter Groups
- External Licenses
- Option Groups
- Subnet Groups
- Events
- Event Subscriptions
- Notifications 18

Launch DB Instance | Show Monitoring | Instance Actions

Filter: All Instances | Search DB Instances... | Viewing 100 of 100 DB Instances

Engine	DB Instance	Status	CPU	DB Load	Maintenance	Class
MySQL	ashkma					
MySQL	test-pira					
Aurora	wenliant					
Aurora	test-rest					
Aurora	test-enh					
Aurora	test-enh					
Aurora	test-aur					
Aurora	test-aur					
Aurora	restorete					
Aurora	nktestau					
Aurora	nktestau					
Aurora	jeffrugg-					
Aurora	hotfix-m					

Feedback | English

AWS | Services | Edit | eichbaum @ awsdesign | N. Virginia | Support

RDS Dashboard
- Instances
- Performance Insights
- Clusters
- Reserved Purchases
- Snapshots
- Security Groups
- Parameter Groups
- Option Groups
- Subnet Groups
- Events
- Event Subscriptions
- Notifications

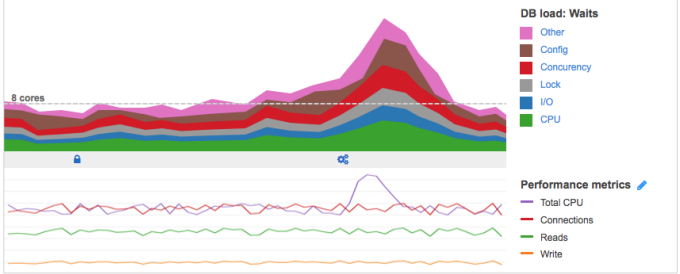
Find by instance name

Name
- demo-db-1
- demo-db-2
- demo-db-3
- demo-db-4
- demo-db-5
- demo-db-6

demo-db-1 Details | Close | Tutorial

Last 60 minutes

SQL | Waits | Sessions | Users | Hosts

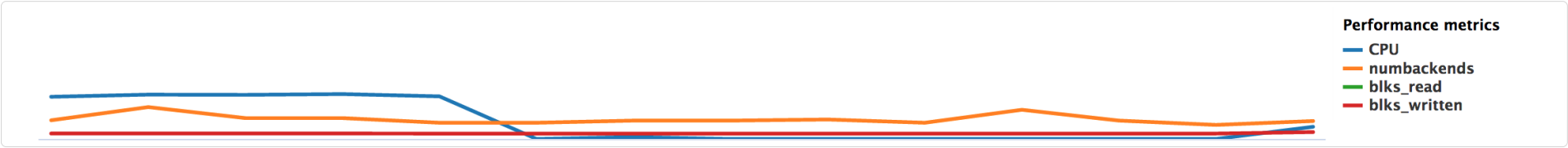


SQL | Waits | Sessions | Users | Hosts

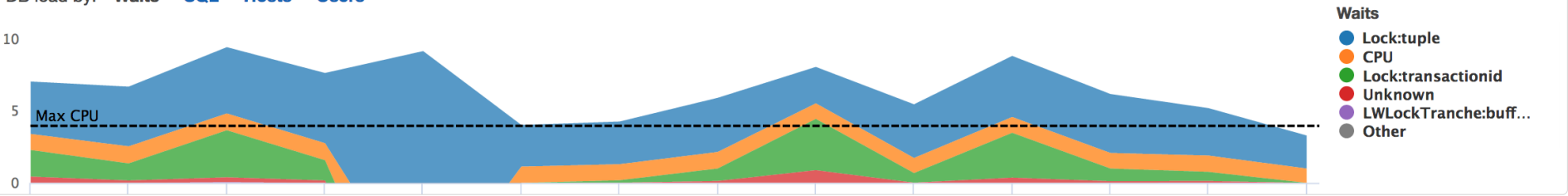
Find by SQL statement text | Showing X of X

ID	DB load	Query
6937C6261555878C		SELECT * FROM 'PRODUCTS' WHERE 'CATEGORY' = ? AND 'SPECIAL' = ? LIMIT ?
CA4BB448AEA3B128		UPDATE 'DS2' . 'INVENTORY' SET 'QUAN_IN_STOCK' = ? , 'SALES' = ? WHERE 'PROD_ID' = ?
0051D481692F0B1A		SELECT * FROM 'PRODUCTS' WHERE MATCH ('ACTOR') 'AGAINST' (?) LIMIT ?
C38F0D78054639D6		SELECT 'QUAN_IN_STOCK' , 'SALES' FROM 'DS2' . 'INVENTORY' WHERE 'PROD_ID' = ?
8F26108AB81288C9		SELECT * FROM 'PRODUCTS' WHERE MATCH ('TITLE') 'AGAINST' (?) LIMIT ?

Last 5 minutes



DB load by: **Waits** SQL Hosts Users



Waits | **SQL** | Hosts | Users

Search SQL Queries

	SQL Digest	DB Load	SQL
▸	4e15b546005d9489980349e399cc1d24		UPDATE pgbench_tellers SET tbalance = tbalance + ? WHERE tid = ?;
▸	9037de313c04df497488ab3670c2466b		UPDATE pgbench_branches SET bbalance = bbalance + ? WHERE bid = ?;
▸	f64d0eecd0bac50e4d71b98c500599af		ROLLBACK TO SAVEPOINT JDBC_SAVEPOINT_1
▸	a30112fac30fcf95bebbdc07e3e38573		select foo();
▸	dc20ac1a0efa57e29ebf7f3df136c600		SELECT * FROM LOGIN("username_in" := $1,"password_in" := $2)
▸	3e20d081813ac00ef7ecd3f778eaefa5		SELECT abalance FROM pgbench_accounts WHERE aid = ?;

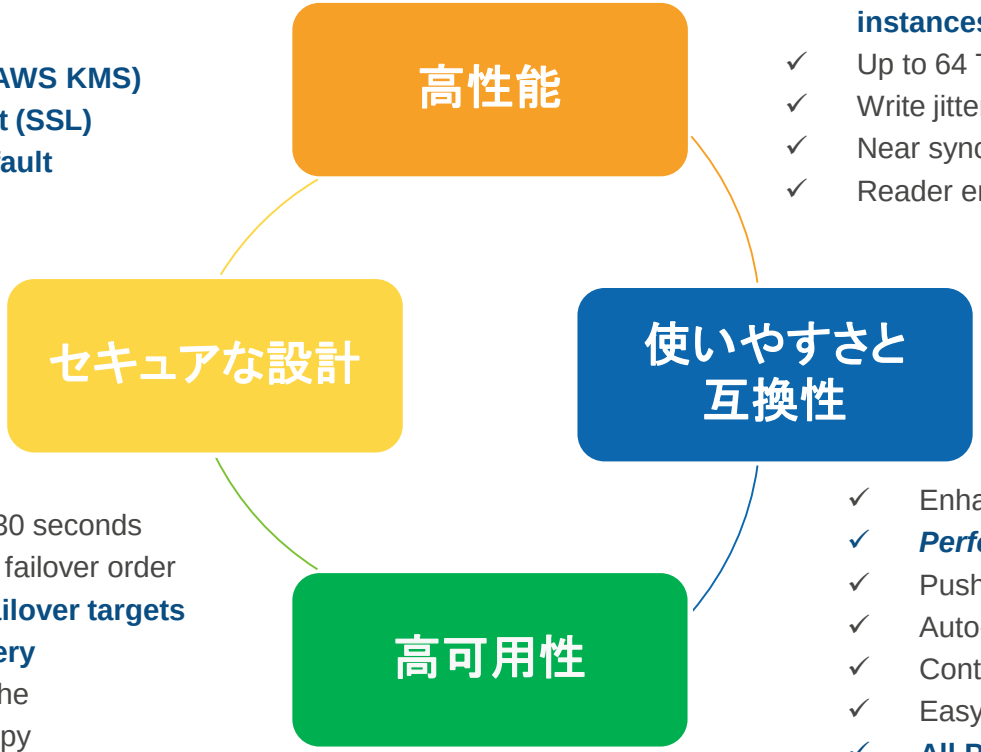
データベース負荷以外にも…

- ロック検出
- 実行計画
- APIアクセス
- RDSに含まれる
- 35日間のデータ保持
- 2017年中にRDSの全データベースエンジンで対応予定

Amazon Aurora with PostgreSQL Compatibility

ロードマップ

Amazon Aurora with PostgreSQL Compatibility
ローンチ時のロードマップ



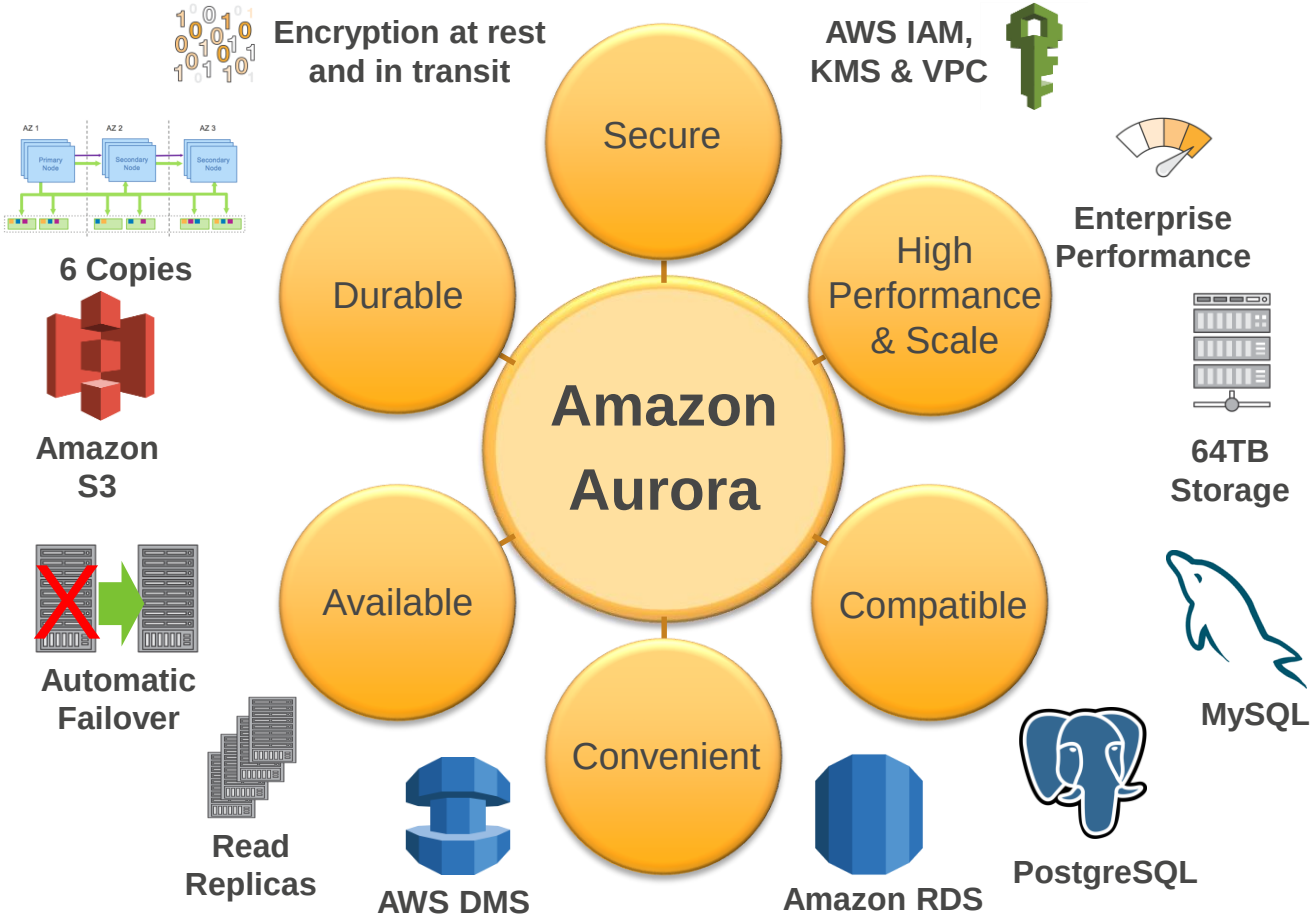
- ✓ **2-4x faster than PostgreSQL on large instances**
- ✓ Up to 64 TB of storage per instance
- ✓ Write jitter reduction
- ✓ Near synchronous replicas (<25ms)
- ✓ Reader endpoint

- ✓ Enhanced OS monitoring
- ✓ ***Performance Insights***
- ✓ Push button migration
- ✓ Auto-scaling storage
- ✓ Continuous backup and PITR
- ✓ Easy provisioning / patching
- ✓ **All PostgreSQL features**
- ✓ **All RDS for PostgreSQL extensions**
- ✓ **AWS DMS supported inbound**

- ✓ Failover in less than 30 seconds
- ✓ Customer specifiable failover order
- ✓ **Up to 15 readable failover targets**
- ✓ **Instant crash recovery**
- ✓ Survivable buffer cache
- ✓ X-region snapshot copy

- ✓ **Encryption at rest (AWS KMS)**
- ✓ **Encryption in transit (SSL)**
- ✓ **Amazon VPC by default**
- ✓ Row Level Security

Amazon Aurora データベース ファミリー



AWS

SUMMIT

Mark Porter
markpor@amazon.com

amazon web services