Amazon RDS for Aurora Deep Dive

TA-02: Tech Deep Dive by Amazon

Yutaka Hoshino

Amazon Data Services Japan K.K.

amazon web services

Amazon Auroraは現在Preview中のため、頻繁に更新が行われています

今回お話する内容は2015/6/2現在の情報となっている点ご注意下さい

amazon web services

自己紹介

- 星野 豊 (ほしの ゆたか)
  - @con_mame
  - facebook.com/conmame
  - ソリューションアーキテクト
- 経歴
  - 全てオンプレ環境のインフラエンジニア
  - 全てAWS環境のインフラエンジニア
- 担当
  - Webサービス / game / Video・Live Streamingなどのメディア系のお客様

Amazon Aurora

amazon web services

データベース管理を簡単に

- データベースを数分で作成可能
- 自動でパッチの適用
- ボタンをクリックするだけでスケールアウト可能
- S3への継続的なバックアップ
- 障害の自動検知と自動フェールオーバ

Amazon RDS

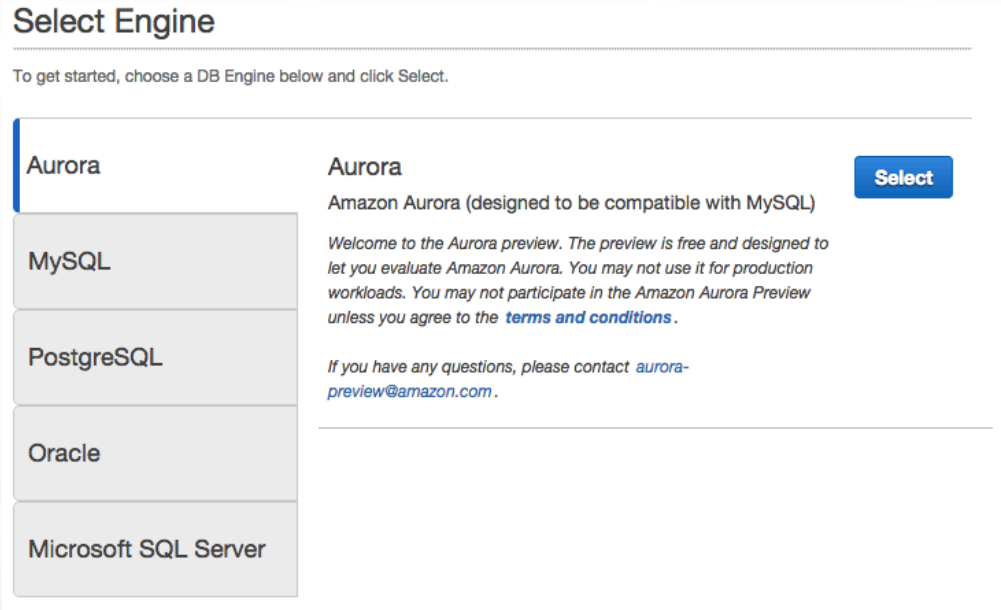

Introducing Amazon Aurora
new!
Commercial-grade database engine at open-source cost

Amazon Aurora

- re:Invent 2014で発表されたRDSの新しいエンジン
- Amazonがクラウド時代にリレーショナル・データベースを作るとどうなるかを1から考え構築
  - 新しい技術的チャレンジを盛り込んでいる
- エンタープライズグレードの可用性とOSSレベルのコストを両立

Amazon Aurora

- Amazon AuroraはRDSが提供するエンジンのうちの1つ
  - RDSでは現在、MySQL / PostgreSQL / Oracle / MS SQL Serverが選択可能



Amazon Aurora

- 現在はLimited Preview中
- Virginia / Oregon / Irelandリージョン
- 2015/5/20 よりpreviewがプロダクション環境へ移行
  - Beta環境はクローズ
  - Beta環境のSnapshotから起動可能
  - 活発に開発・デプロイが行われている

Amazon Aurora pricing

	vCPU	Mem	Hourly Price
db.r3.large	2	15.25	$0.29
db.r3.xlarge	4	30.5	$0.58
db.r3.2xlarge	8	61	$1.16
db.r3.4xlarge	16	122	$2.32
db.r3.8xlarge	32	244	$4.64

- ライセンス料金は不要
- ロックインもない
- 使った分だけ課金

- ストレージ: $0.10/GB/month
- IO課金: $0.20 per million IO
- Virginiaリージョンの価格

Amazon Auroraの特徴

クエリ性能の向上

スケーラブル

MySQL5.6互換

セキュリティにも配慮

コストパフォーマンスが良い

高可用性・高耐久性

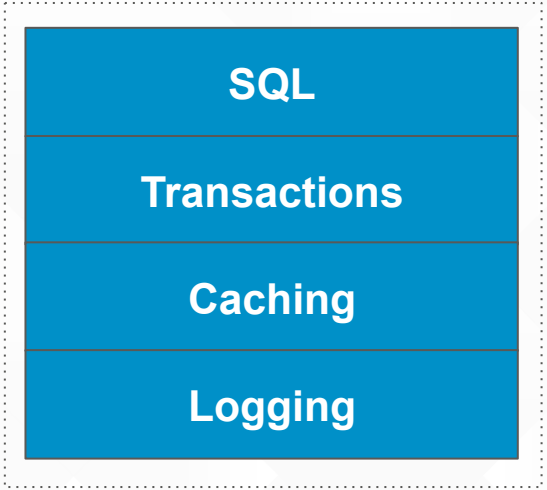
Amazon Auroraの特徴

- MySQL5.6と互換性があるため既存のアプリケーションを簡単に移行可能
- ストレージが10GBから64TBまでシームレスに拡張
- 3AZに2つずつ、計6つのデータのコピーを保持
  - S3にストリーミングバックアップを実施
- VPC内に起動
  - Security GroupやNACLを使用してアクセスコントロール可能
- Amazon Auroraは99.99%の可用性を実現するように設計されている

なぜAmazonがデータベースを再考したか

amazon web services

現在のモノリシックなDB

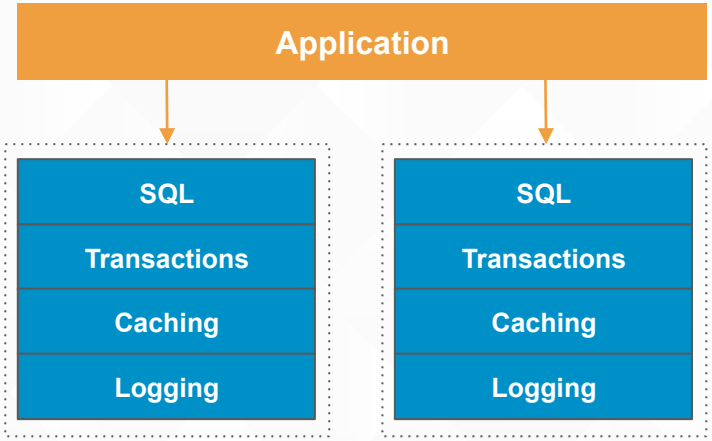


複数の機能レイヤーが
1つのアプリケーショ
ンになっている

現在のモノリシックなデータベース

Sharding

アプリケーションレイヤでシャーディング

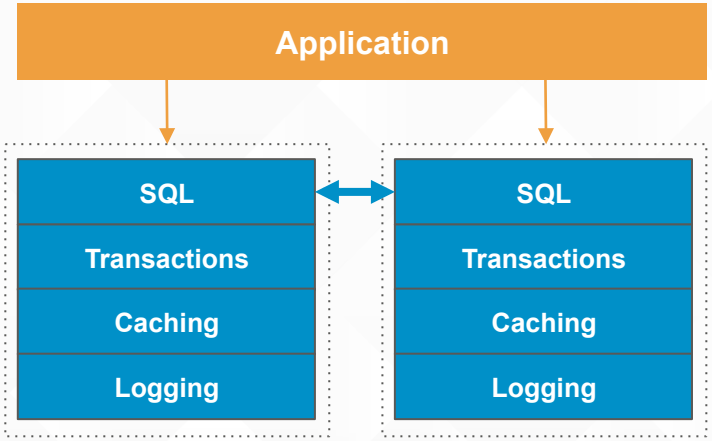


スケールアウトする場合は、このセットを増やしていく必要がある

現在のモノリシックなデータベース

Shared Nothing

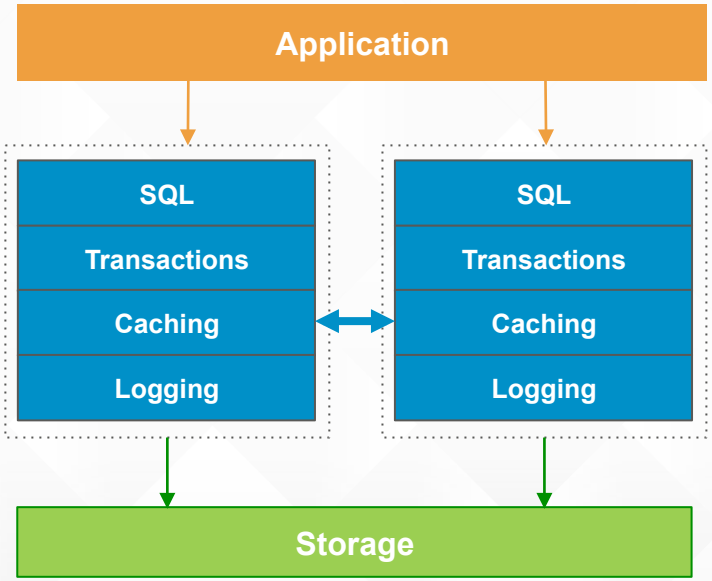
SQLレイヤ



スケールアウトする場合は、このセットを増やしていく必要がある

現在のモノリシックなデータベース

Shared Disk
キャッシュとストレージレイヤ



スケールアウトする場合は、このセットを増やしていく必要がある

コスト・可用性・柔軟性の面で問題

リレーショナルデータベースをもう一度考える

- 今、データベースを再度実装するならどうするか？
  - 少なくとも1970年代の方法で実装はしない
  - AWSサービスを活かすことができ、スケールアウトが簡単で、セルフヒーリングが出来るようなデータベースを作りたいと考えた

amazon web services

クラウド時代に適したリレーショナルデータベース

- ハイエンドデータベースの様なスピード 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果の高さ
- MySQLと互換性を保つ
- 利用した分だけお支払いいただく課金モデル
- AWSサービスと簡単に連携

マネージド・サービスとしてご提供

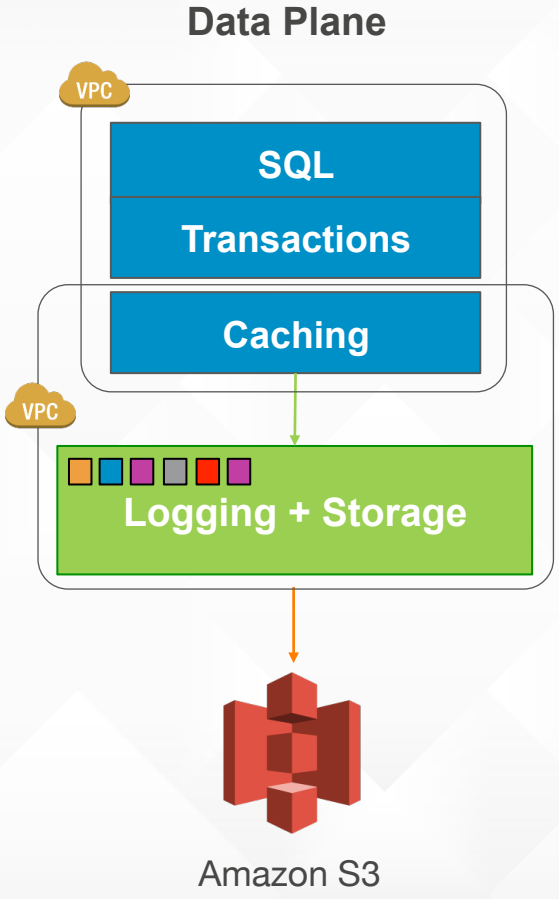
amazon web services

アーキテクチャ

amazon web services

Service Oriented Architecture

- ログとストレージレイヤをシームレスにスケールするストレージサービスに移動
- EC2, Amazon DynamoDB, Amazon SWFなどのAWSサービスを管理コンポーネントに採用
- Amazon S3を利用して99.999999999%の可用性でストリーミングバックアップ

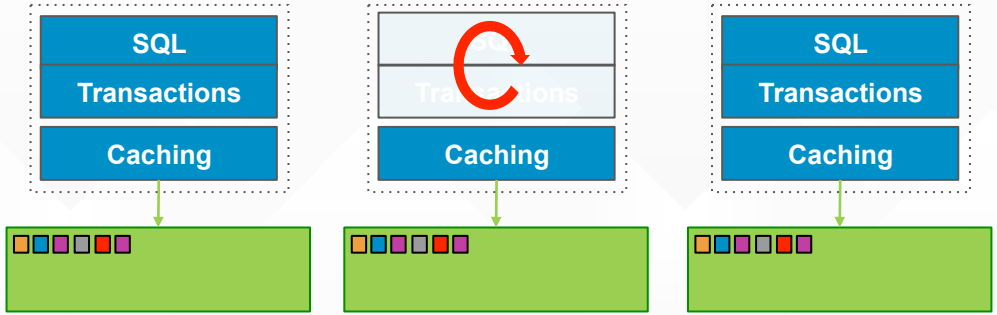


amazon web services

キャッシュレイヤの分離

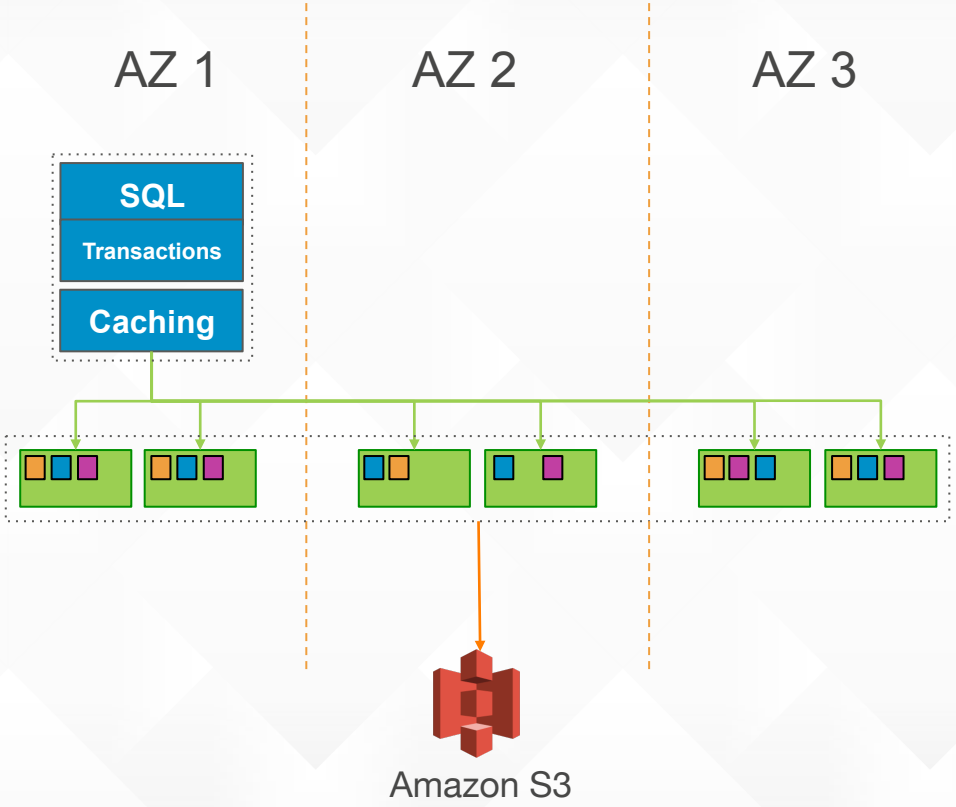
- キャッシュをデータベースプロセス外に移動させた
- データベースプロセスのリスタートが発生してもキャッシュが残った状態を維持可能
- サービスにすぐデータベースを戻すことが出来る

キャッシュプロセスをDBプロセス外におくことで
DBプロセスの再起動でもキャッシュが残る



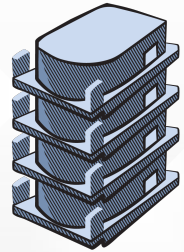
Auroraのストレージ

- SSDを利用したシームレスにスケールするストレージ
  - 10GBから64TBまでシームレスに自動でスケールアップ
  - 実際に使った分だけ課金
- 標準でHighly availableを実現
  - 3AZに6つのデータのコピーを作成
  - 2つのディスクが利用不能でも読み書き可能
    - 万が一1つのAZが利用不能になっても3本で読み書き可能な状態で稼働
  - 3つのディスクが利用不能の場合読み込みのみ可能
- Log structured Storage
  - redo logを複数の小さなセグメントに分割
  - Log pageによってData pageを作成



Auroraのストレージ

- Amazon Auroraは6本全てのディスクへの書き込みを待たずに、少なくとも4つのディスクに書き込みが完了するとすぐに次の処理を実行
- ホットスポットの影響を取り除き、非常に高い並列度を実現
- ストレージはSSDベースのディスクに10GBずつのブロック内に分散して書き込まれる

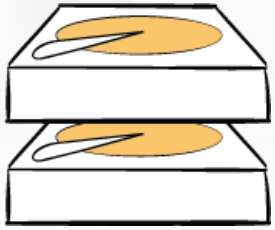


amazon web services

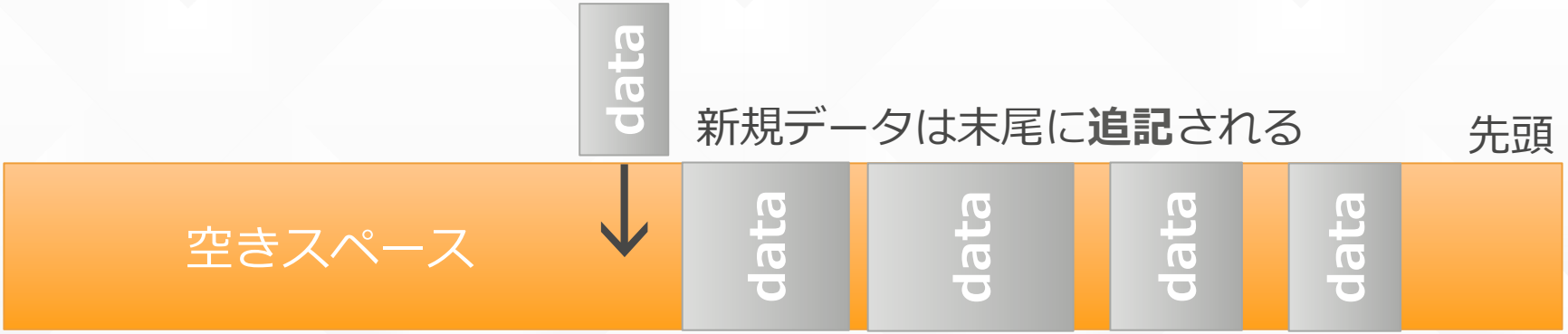
Auroraのストレージの特徴

- リードレプリカもマスタと同じストレージを参照
- Log Structured Storage
- 継続的なS3へ増分バックアップ
  - パフォーマンスへの影響なし
- 64TBまで自動でストレージがシームレスにスケールアップ
  - パフォーマンスや可用性に影響無し・利用開始時のプロビジョニング不要
- 自動で再ストライピング、ミラー修復、ホットスポット管理、暗号化

Log Structured Storage

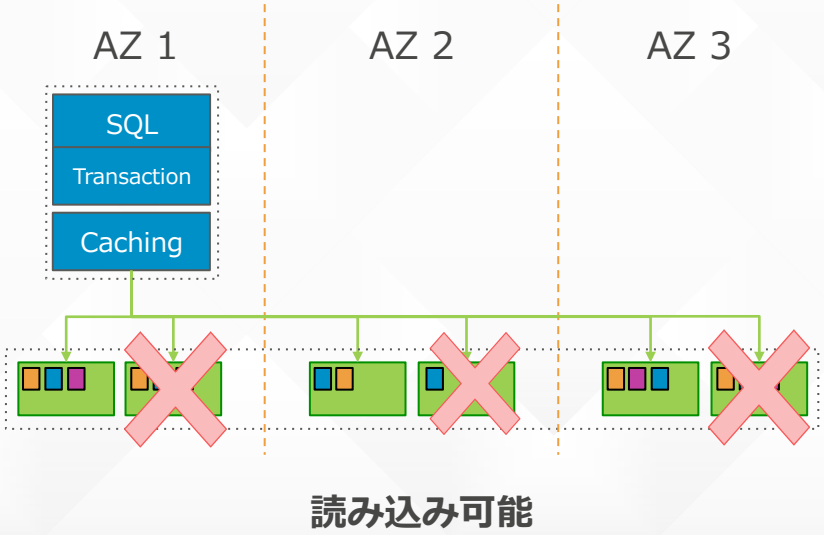


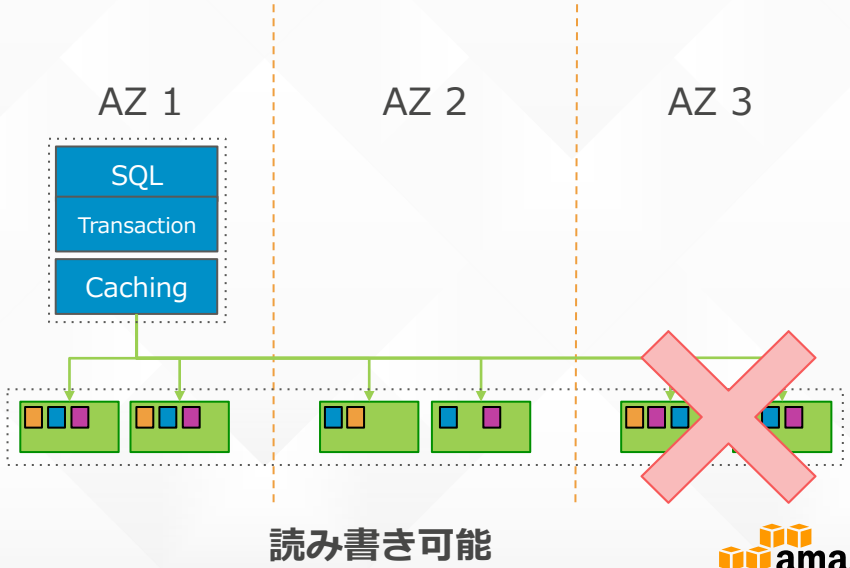
- 追記型のストレージ・システム
  - ログの様に常に末尾にデータを追加していくだけ
  - データが書き込まれているブロックを上書いたりはしない
  - GCによりデータを効率的に格納する
- シーケンシャルに読み出すことが出来る
- 常に最新のデータが末尾にある
- これらの特徴によりS3への継続バックアップや高速なリカバリ、書き込み性能の向上を実現



ディスク障害検知と修復

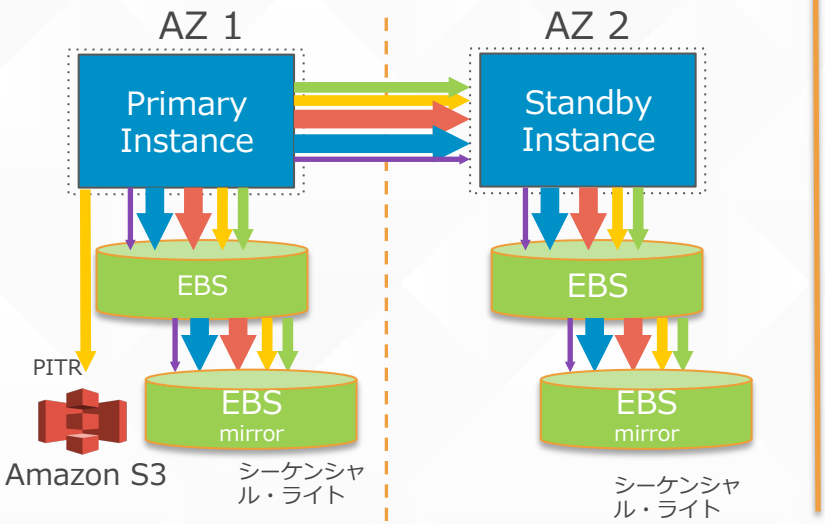
- 2つのコピーに障害が起こっても、読み書きに影響は無い
- 3つのコピーに障害が発生しても読み込みは可能
- 自動検知、修復



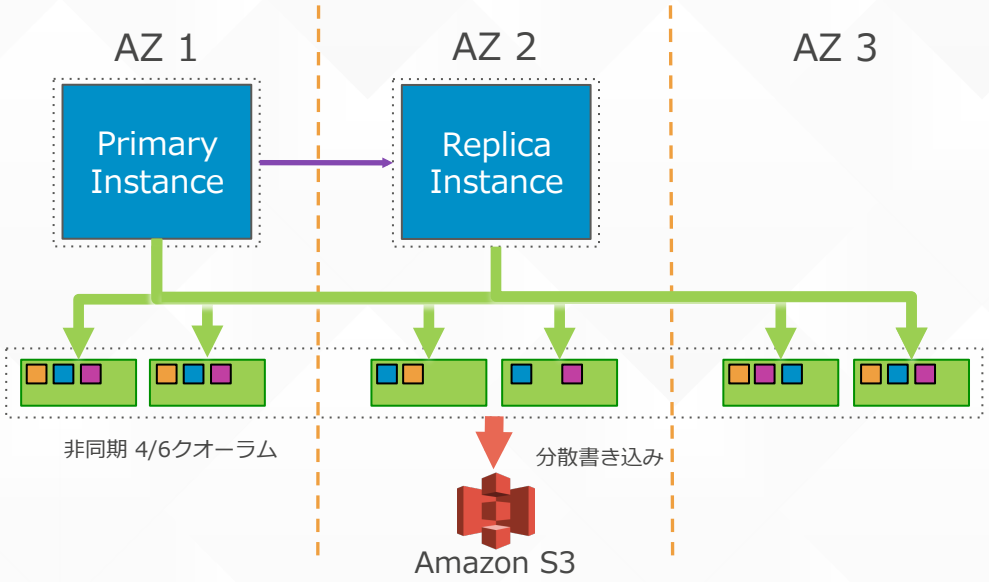


レプリケーション

MySQL レプリケーション



Amazon Aurora

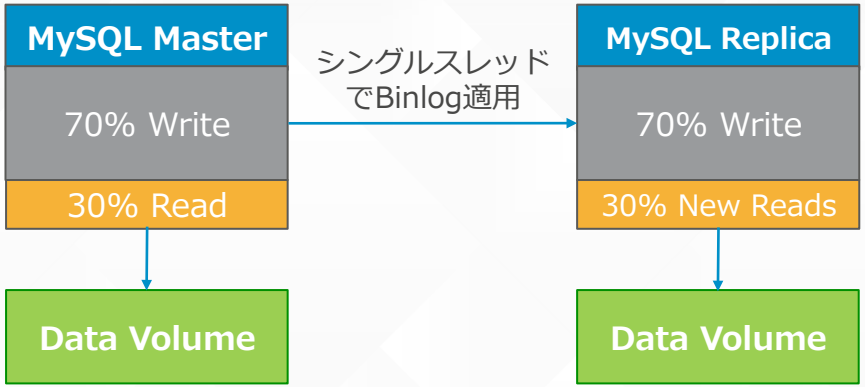


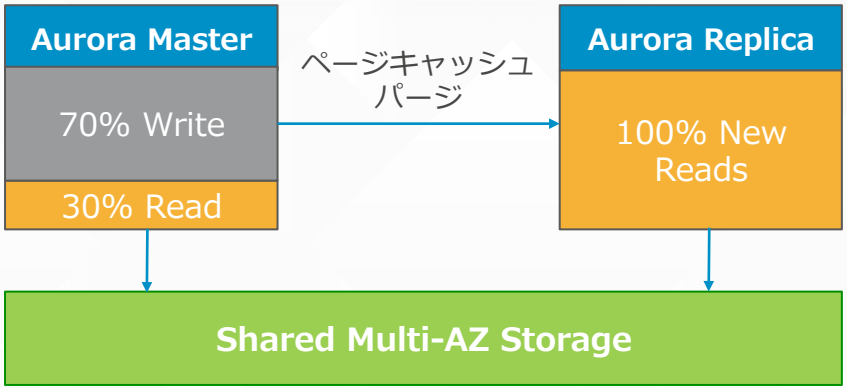
改善点

- Consistency – 異常を修復
- Latency – 同期 vs 非同期レプリケーション
- network I/Oを効率的に行う

amazon web services

レプリケーション





MySQL read scaling

- レプリケーションにはbinlog / relay logが必要
- レプリケーションはマスターへ負荷がかかる
- レプリケーション遅延が増加していくケースがある
- フェイルオーバでデータロスの可能性がある

Amazon Aurora read scaling

- Binlogによるレプリケーションではない
- マスターへの負荷を最小限に15台までリードレプリカを作成可能
- 100ms以内のレプリケーション遅延
- マスターとリードレプリカで同じディスクを共有しているため、フェイルオーバでデータロスが無い

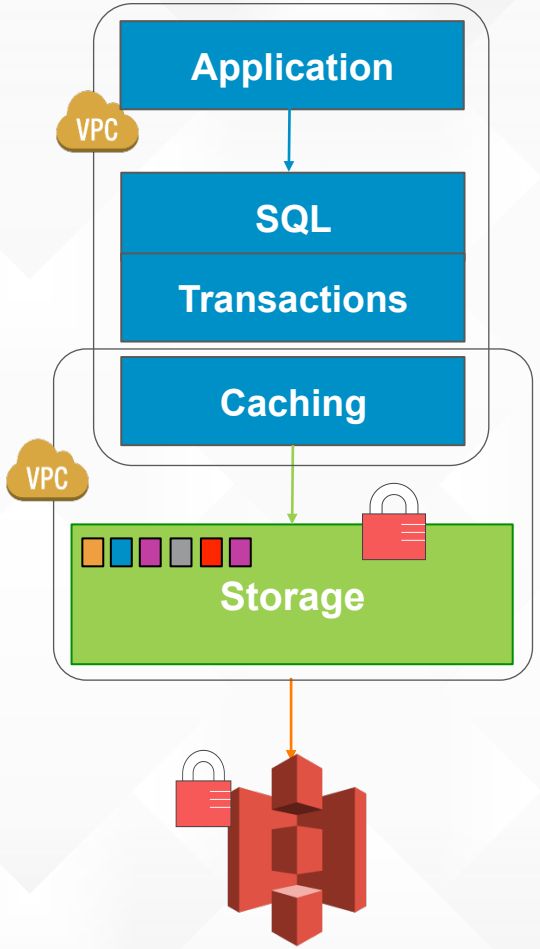
amazon web services

レプリケーション

- Amazon Auroraは、15台のリードレプリカを作成可能
  - リードレプリカはマスタサーバとストレージを共有しており、t低負荷で粒度の高いほぼ同期型のレプリケーションを行う
  - 最大100ミリ秒オーダーの遅延でレプリケーションされる
  - RDS for MySQLではリードレプリカは5つまで (孫リードレプリカを入れて30)

セキュリティ

- データの暗号化
  - AES-256 (ハードウエア支援)
  - ディスクとAmazon S3に置かれている全ブロックを暗号化
  - AWS KMSを利用したキー管理
- SSLを利用したデータ通信の保護
- 標準でAmazon VPCを使ったネットワークの分離
- ノードへ直接アクセスは不可能
- 業界標準のセキュリティとデータ保護の認証をサポート



amazon web services

DBクラスタ

- Amazon AuroraはDBクラスタという概念を持っている
  - マスタ (Writer)とリードレプリカ(Reader)をひとまとめにしたもの
  - Parameter GroupやMaintenance WindowもDBクラスタと各ノードそろぞれに存在する
- フェイルオーバが発生しても常にマスタを参照するエンドポイントがクラスタ毎に1つ存在する
  - アプリケーションからのWriteクエリは常にこのエンドポイントを参照するように設定

DB Parameter GroupとDB Cluster Parameter Group

- RDS for MySQLではDB Parameter Groupのみ
- Auroraでは設定の適用範囲毎にグループを設定
  - DB Cluster Parameter Group: Auroraクラスタ内全ノードで共通
  - DB Parameter Group: 各Auroraノード個別の設定

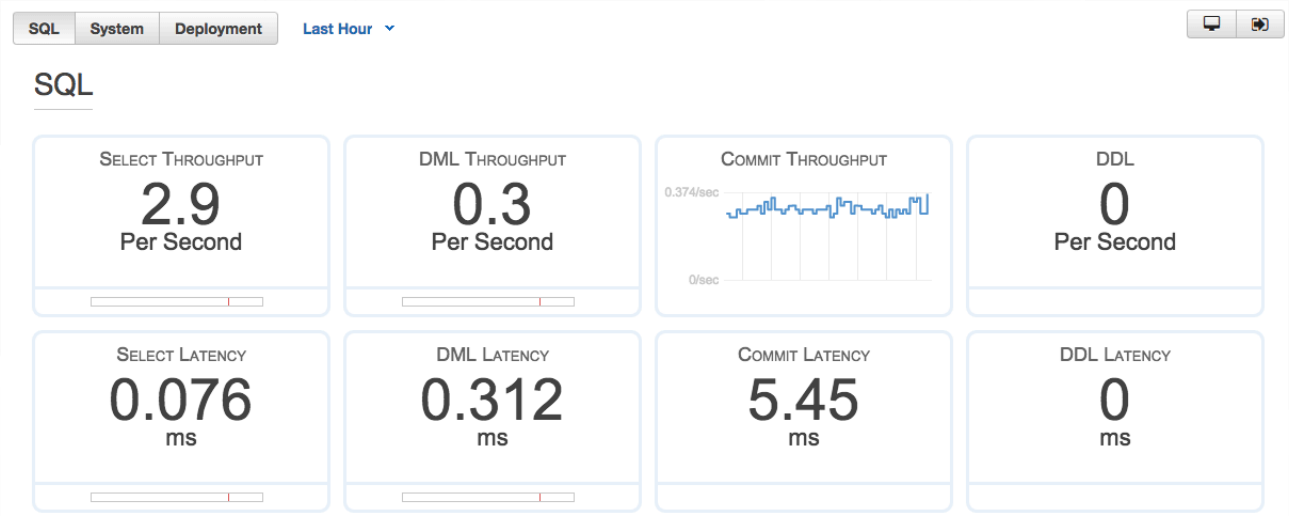
Configuration Details

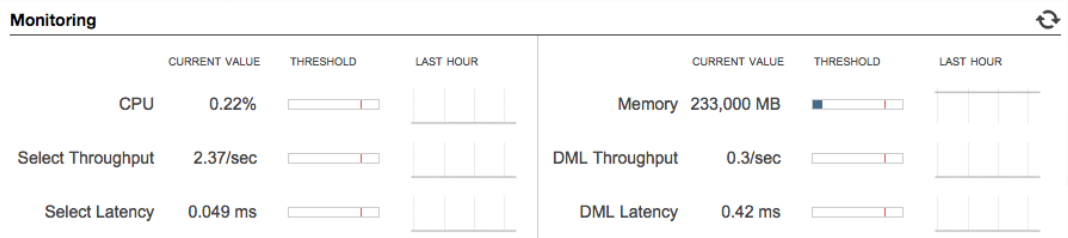
Engine	Aurora 5.6.10a
Created Time	May 14, 2015 at 5:17:17 PM UTC+9
DB Name	demo
Username	demo
Parameter Group	default.aurora5.6 (in-sync)
DB Cluster Parameter Group	default.aurora5.6 (in-sync)

Name	Family	Type	Description
default.aurora5.6	aurora5.6	DB Parameter Group	Default parameter group for aurora5.6
default.aurora5.6	aurora5.6	DB Cluster Parameter Group	Default cluster parameter group for aurora5.6

新しいメトリクス画面

- Throughput
  - Select
  - Commit
  - DML/DDL
- Latency
  - Select
  - Commit
  - DML/DDL
- Cache Hit Ratio
  - Buffer Cache
  - Result Set
- Deadlocks
- Login Failures
- Blocked Transactions



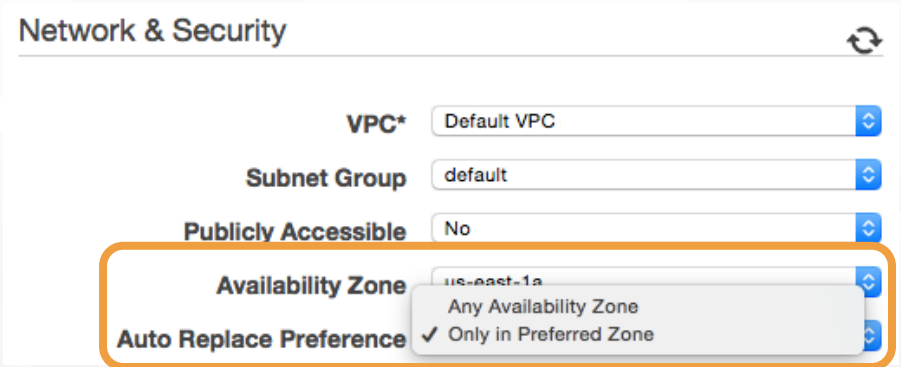


amazon web services

フェイルオーバとリカバリ

フェイルオーバ と リプレース

- リードレプリカが存在する場合は1分程でフェイルオーバ可能
  - RDS for MySQLよりも高速にフェイルオーバ可能
  - リードレプリカが存在しない場合は10分程
- 優先的にフェイルオーバさせるノードを1つ指定可能
  - Multi-AZ配置として別AZで起動する
  - RDS for MySQLと違いリードアクセス可能
- ノードリプレース時に新Auroraノードを起動するAZを指定可能
  - 指定したAZ
  - 問題のないAZの中から自動で選択



クラスタエンドポイント

- WriterとReaderのセットをクラスタと呼び、クラスタで常にWriter(マスタ)を指すクラスタエンドポイントが存在する
- 各Auroraノードは個別にエンドポイントを持っている

Cluster Endpoint: demo-cluster-1.cluster-[redacted].us-east-1-beta.rds.amazonaws.com:33((authorized)

DB Cluster Details

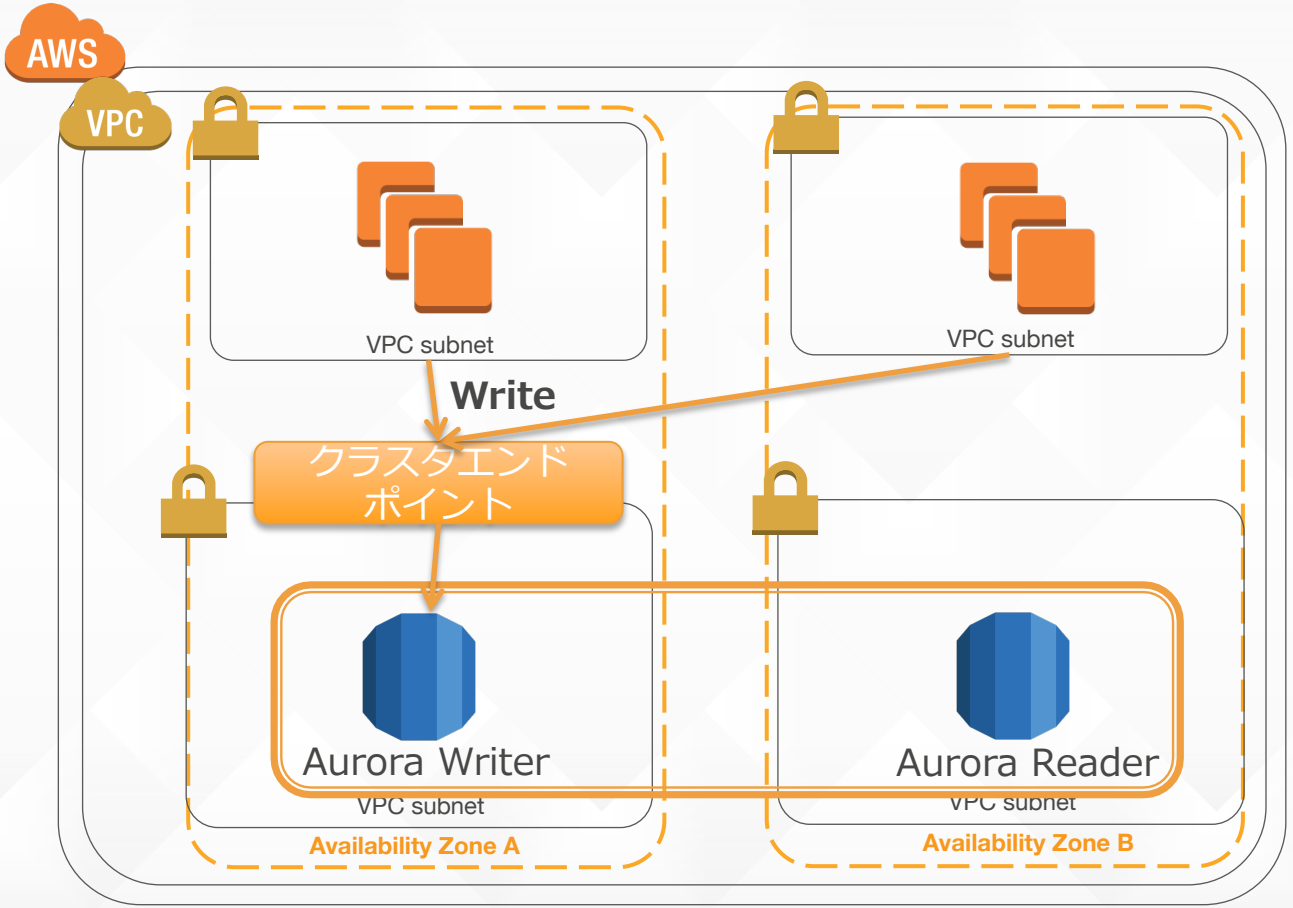
DB Cluster	demo-cluster-1 (available)
Endpoint	demo-cluster-1.cluster-[redacted].us-east-1-beta.rds.amazonaws.com
Port	3306
Automated Backups	Enabled (1 Day)
Earliest Restorable Time	May 5, 2015 14:49:22 PM UTC+9
Latest Restore Time	May 5, 2015 15:48:20 PM UTC+9
Backup Window	09:42-10:12
Maintenance Window	fri:05:35-fri:06:05
DB Cluster Parameter Group	default.aurora5.6

Deployment DB Instances In Region

DB INSTANCE	ROLE	ZONE	REPLICATION SOURCE	REPLICA LAG
demo	writer	us-east-1a	demo-cluster-1	-
demo-us-east-1b	reader	us-east-1b	demo-cluster-1	20.466 ms
demo-us-east-1d	reader	us-east-1d	demo-cluster-1	19.032 ms

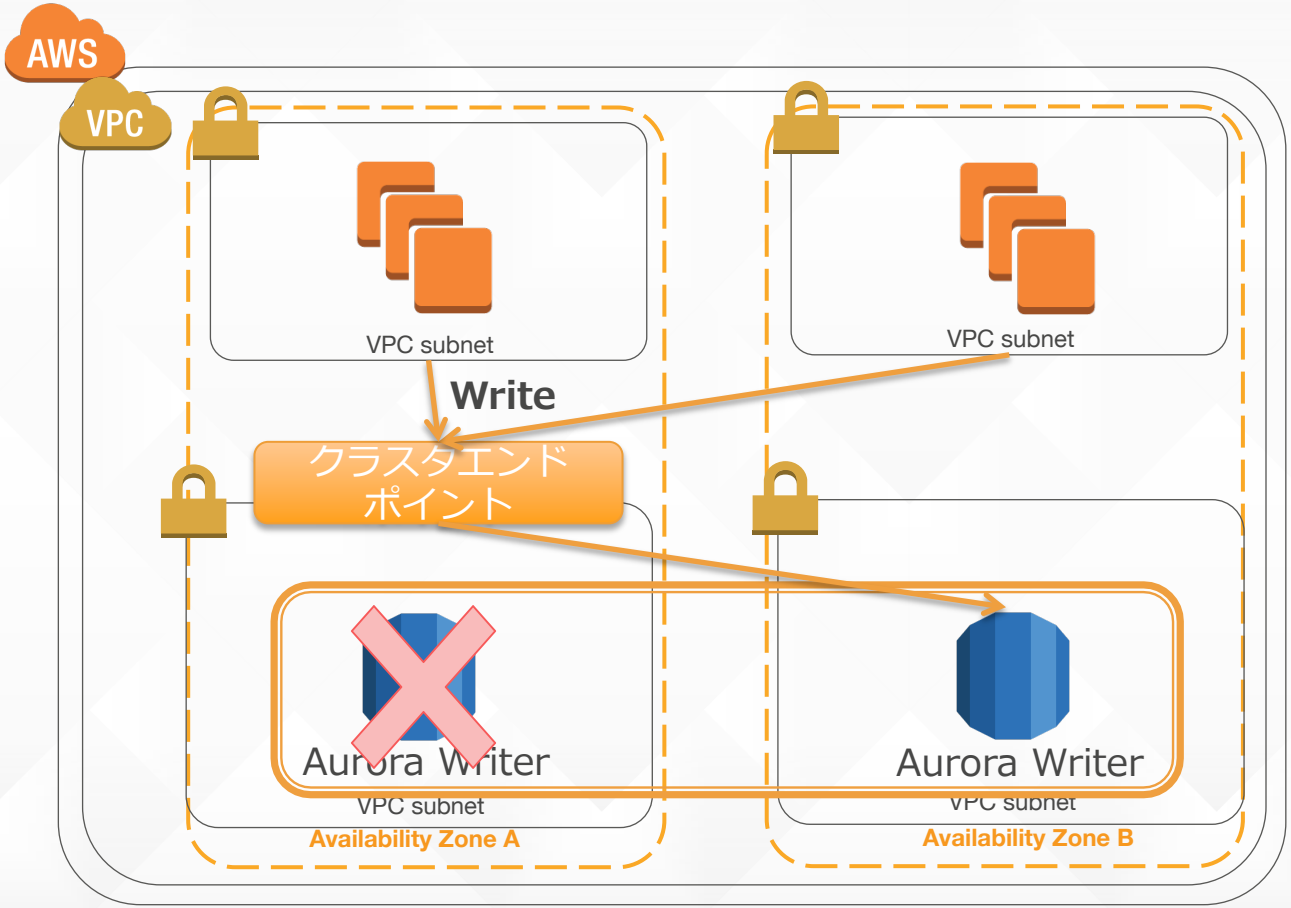
amazon web services

クラスタエンドポイント



- 各Auroraノードは個別にエンドポイントを持っている
- クラスタエンドポイントは、その時アクティブなAurora WriterノードのCNAME
- Readは各Readerを参照する

クラスタエンドポイント



- フェイルオーバが発生すると、Auroraノードの昇格が行われ、クラスタエンドポイントの指し先が変わる

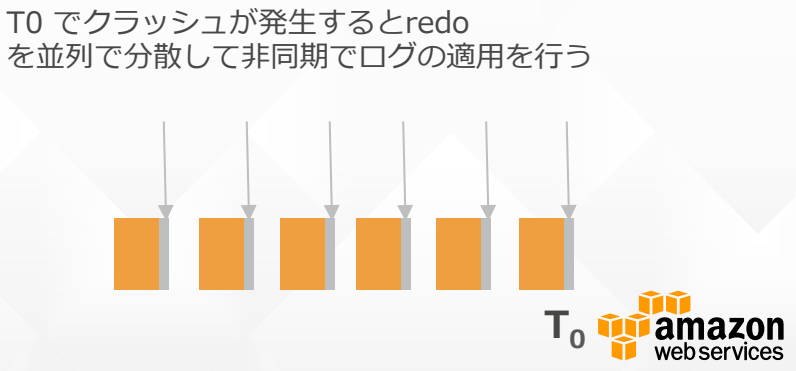
高速なデータ修復

既存のデータベース

- 最後のチェックポイントからログを適用していく
- MySQLではシングルスレッドなため適用完了までの時間が増加

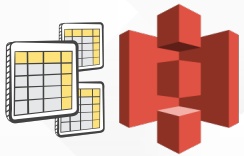
Amazon Aurora

- Disk readの一環として、オンデマンドでredo logの適用を行う
- 並列、分散、非同期で行われる



Streaming snapshotとPITR

- Amazon Auroraでは各セグメント毎にAmazon S3へ継続的に増分バックアップを取得している
  - Backup retention periodでバックアップを残す期間を指定可能
- Amazon Auroraが使用しているディスクの仕組みによりパフォーマンスへ影響を与えない
- PITRで5分前からBackup Retention Periodまでの任意の位置に秒単位で復元可能



amazon web services

SQLによるフェイルオーバのテスト

SQLによりノード・ディスク・ネットワーク障害をシュミレーション可能

- **データベースノードのクラッシュをシュミレート:**
  ALTER SYSTEM CRASH [{INSTANCE | DISPATCHER | NODE}]

- **レプリケーション障害をシュミレート:**
  ALTER SYSTEM SIMULATE percentage_of_failure PERCENT
  READ REPLICA FAILURE [TO ALL | TO "replica name"]
  FOR INTERVAL quantity [YEAR | QUARTER | MONTH | WEEK| DAY | HOUR | MINUTE | SECOND];

- **他にも**
  - **ディスク障害をシュミレート**
  - **ディスク障害(遅延)をシュミレート**
  - **ネットワーク障害をシュミレート**

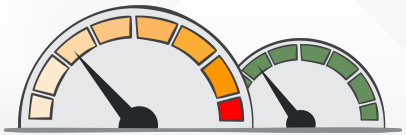
パフォーマンス

amazon web services

Auroraのパフォーマンスを引き出すために

- クエリ並列度が高い、データサイズが大きいケースで効果を発揮

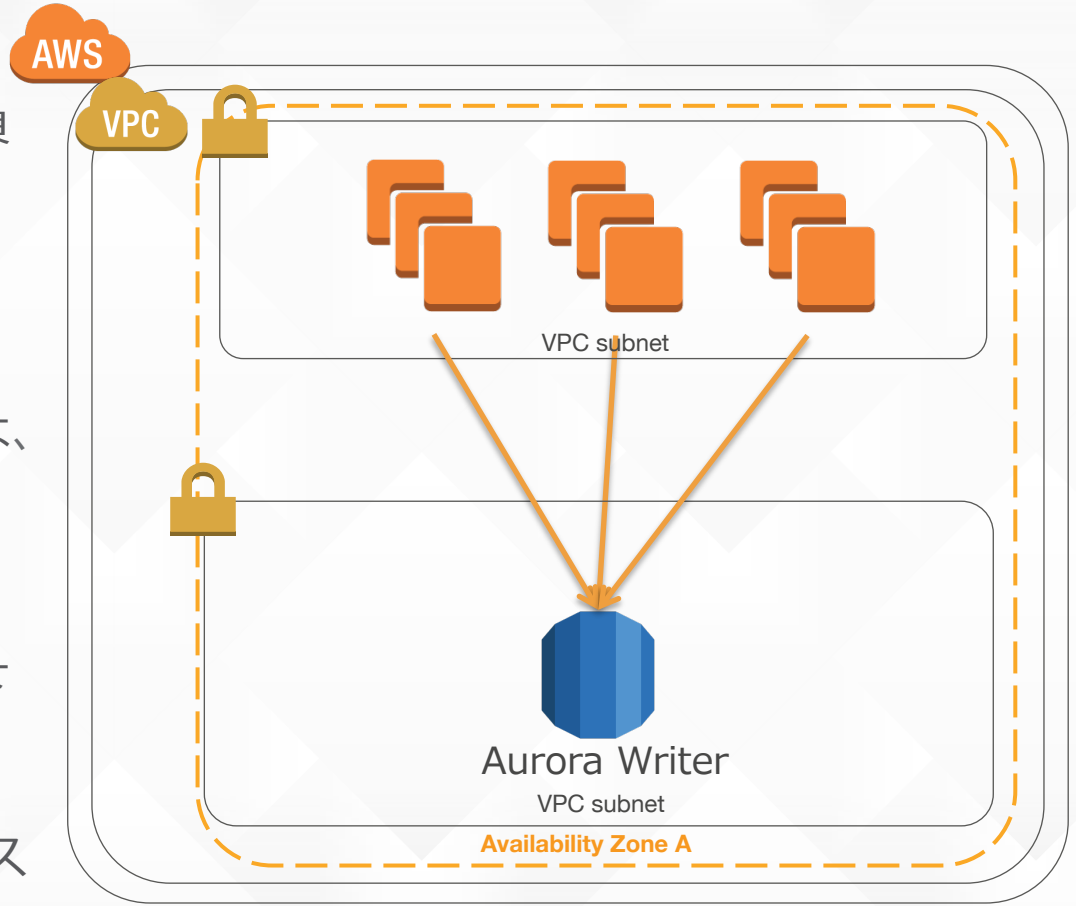
- ロック機構やQuery cacheなどに手を入れて性能向上を行っている
  - write heavyな環境ではoffをおすすめ
  - CPUを効率的に利用する改善により、CPU利用率がMySQLと比較して高くなるが、性能が落ちにくくなっている



amazon web services

パフォーマンス測定

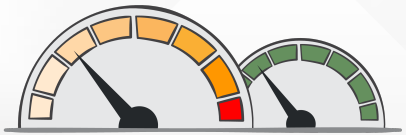
- 複数のインスタンスから同時に負荷をかけ並列度を上げる
  - NWの影響を抑えるために同一リージョンで行う
- 単一のインスタンスからだけでは、インスタンス毎のNW帯域の制限に達する可能性がある
- 高負荷環境でもスループット低下を抑える改善が入っているためCPU/メモリ利用率がRDS for MySQLと比較して高くなるケースがある



amazon web services

パフォーマンス

- 性能が5倍というのはどのようなケースか
  - 性能面は最大5倍
  - re:Invent で発表された5倍という性能はSysbenchを4インスタンスからr3.8xlargeのAuroraインスタンスに実行した場合の結果
- TPC-C をr3.8xlargeに実行した場合は約2.5倍の性能を観測している



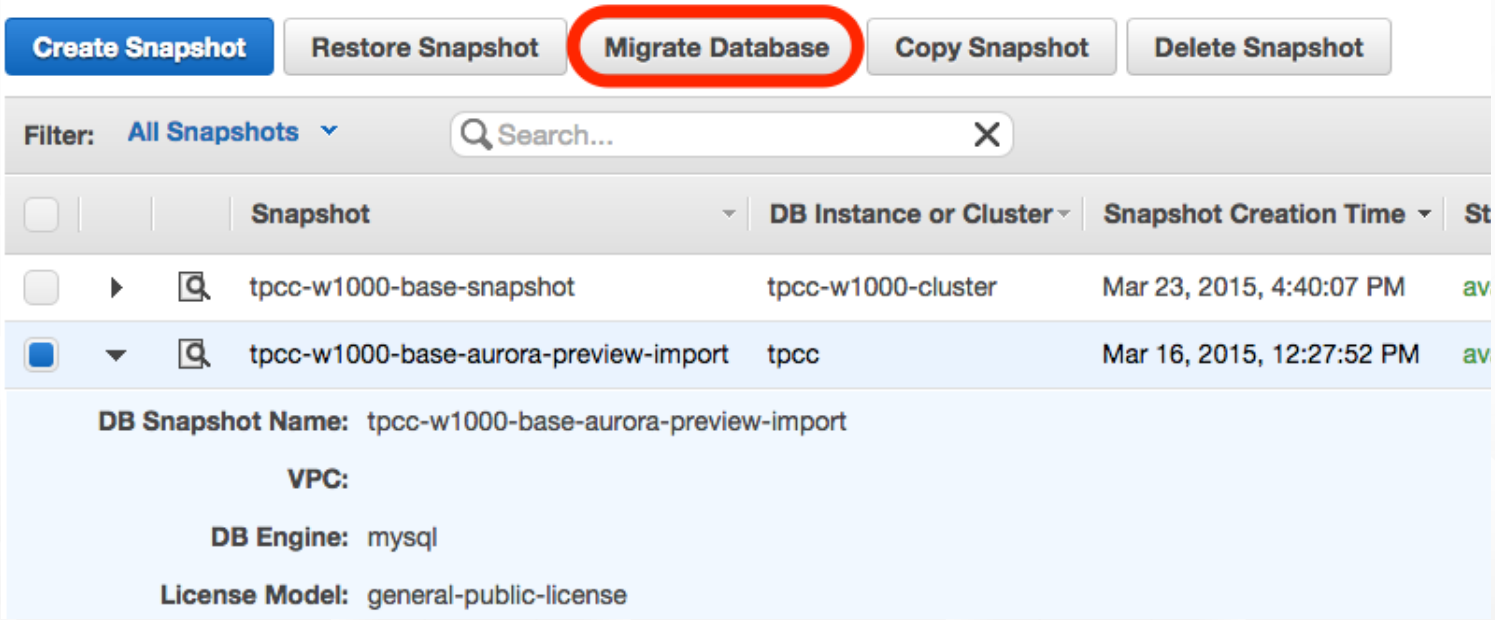
amazon web services

Amazon Auroraへの移行

amazon web services

RDS for MySQLからマイグレーション

- マネージメントコンソールから数クリックでAmazon Auroraへ移行可能
  - RDS for MySQLのスナップショットからAmazon Auroraへマイグレーション可能
  - RDS for MySQLは5.6を使う必要がある



マイグレーション時の注意

- RDS for MySQLとParameter Groupで設定出来る項目や規定値などが異なる
  - 例: max_connection / innodb_buffer_pool_size / query_cache_*など
- マイグレーションに必要なディスクスペース
  - スナップショットをインポートする場合、インポート前にEBSボリュームを使用しデータをフォーマットする
  - データをフォーマットするための追加容量が必要になる場合がある

マイグレーション時の注意

- MyISAM形式のテーブルが含まれない場合
  - 移行前のディスクで3TBまで容量を利用可能
- MyISAM形式のテーブルが含まれる場合
  - マイグレーションを行うテーブルで1.5TBを超えるものが無いことを確認する

マイグレーション時の注意

- Amazon AuroraはInnoDBのみサポート
  - MyISAMなどのストレージエンジンは非対応

MySQLからレプリケーション

- MySQL5.6からAmazon Auroraへレプリケーションを行うことが可能
  - Amazon AuroraからMySQLへは現状未対応
- 専用のProcedureを使用

```
mysql > CALL mysql.rds_set_external_master (DB Hostname or IP address',
3306,’user', ‘password', ’Binlog', position, 0);

mysql > CALL mysql.rds_start_replication;
```

MySQLからレプリケーション

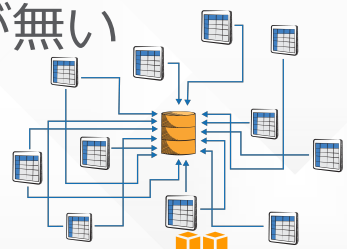
- RDS for MySQLやMySQL on EC2、オンプレ環境のMySQLからAmazon Auroraにレプリケーション可能
  - バックアップからAuroraにインポートを行い、レプリケーションを実行
  - 移行時にアプリケーションのメンテナンスを入れ、書き込みがなくなり、レプリケーションが追いついたタイミングでアプリケーションの書き込み先などをAuroraに変更

Amazon Auroraの使いどころ

amazon web services

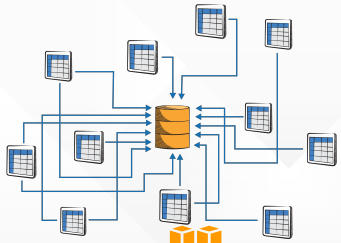
クエリ同時実行数やテーブルサイズが大きい

- Amazon Auroraに移行することで、クエリスループットの向上などが見込まれる
  - マルチコア環境でCPUを効率的に利用
  - 分散ロック機構やquery cacheの改善による性能向上
- ディスク
  - データ量の増加に応じてディスク容量を気にする必要が無い
  - 性能に影響を及ばさずバックアップ



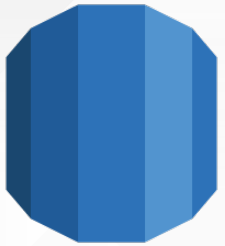
複数のサーバにシャーディングしている

- 複数の小さいDBを1つにまとめる
  - コスト効果増大と管理コストの軽減
  - シャーディングををするデータベースを減らすことでアプリケーションの設計を簡略化出来る
  - 障害時の影響を考慮する必要はある



まとめ

Amazon Aurora



- クラウド時代にAmazonが再設計したRDBMS
  - MySQL5.6と互換があり既存の資産を活かしやすい
- 高いクエリ実行並列度・データサイズが大きい環境で性能を発揮
- 高可用性・高速なフェイルオーバ・PITRを実現するための多くのチャレンジ
  - Log Structured Storage
  - SOA

Thank You

amazon web services