classmethod

Auroraの凄さを振り返る

JAWS-UG 初心者支部#13　「AWS Night school」

2018/8/22

Classmethod, Inc. AWS Business Unit Consulting Div.

Hajime Oguri

Attention

スライドは後で入手することが出来ますので
発表中の内容をメモする必要はありません。

写真撮影をする場合は
フラッシュ・シャッター音が出ないようにご配慮ください

Who am I ?

大栗 宗（@maroon1st）

メーカー系SIerに10年ほど在籍して
クラスメソッドにジョイン

AM5:00に起きてAWSのアップデートを
ブログに書くのがメイン業務で、AWSの
導入コンサルティングやインフラ構築も
行っています

AWSの認定試験は全部取りました

好きなサービス：Amazon Aurora

月間170万PVを誇る技術ブログ「Developers.IO」

Developers.IO

https://dev.classmethod.jp/

月間170万PV、50万UUを誇る、社員が執筆するIT技術に特化したオウンドメディア

AWS、ビッグデータ、モバイル、IoT 、音声認識技術などの多岐にわたる記事を掲載中

- ユーザに有益な情報であれば社内のノウハウも余すところなく記事化
- 現在13,000本以上の記事を掲載（2018年7月現在）
- 憶測やセオリーだけでなく、実地検証に基づく「やってみた」記事を公開

過去の記事特集

AWS re:Invent 2016 (209本)
AWS re:Invent 2017 (393本)

毎年アメリカにて開催されるAWS re:Invent、2017年は社員36名が参加し、AWSの新サービスや使い方についていち早く記事を公開しました。

iOS 10 公開記念新機能を網羅する200tips(213本)

2016年9月iOS 10 のリリースを記念して、iOS 10 や Swift 3、Xcode 8 の新機能に関する記事を一挙にリリースしました。

Auroraのエントリーが現在99本！

Agenda

- 今日話すこと・話さないこと
- 基本的な特徴
- Well-Architectedに学ぶAuroraの凄さ
- まとめ

今日話すこと・話さないこと

話すこと

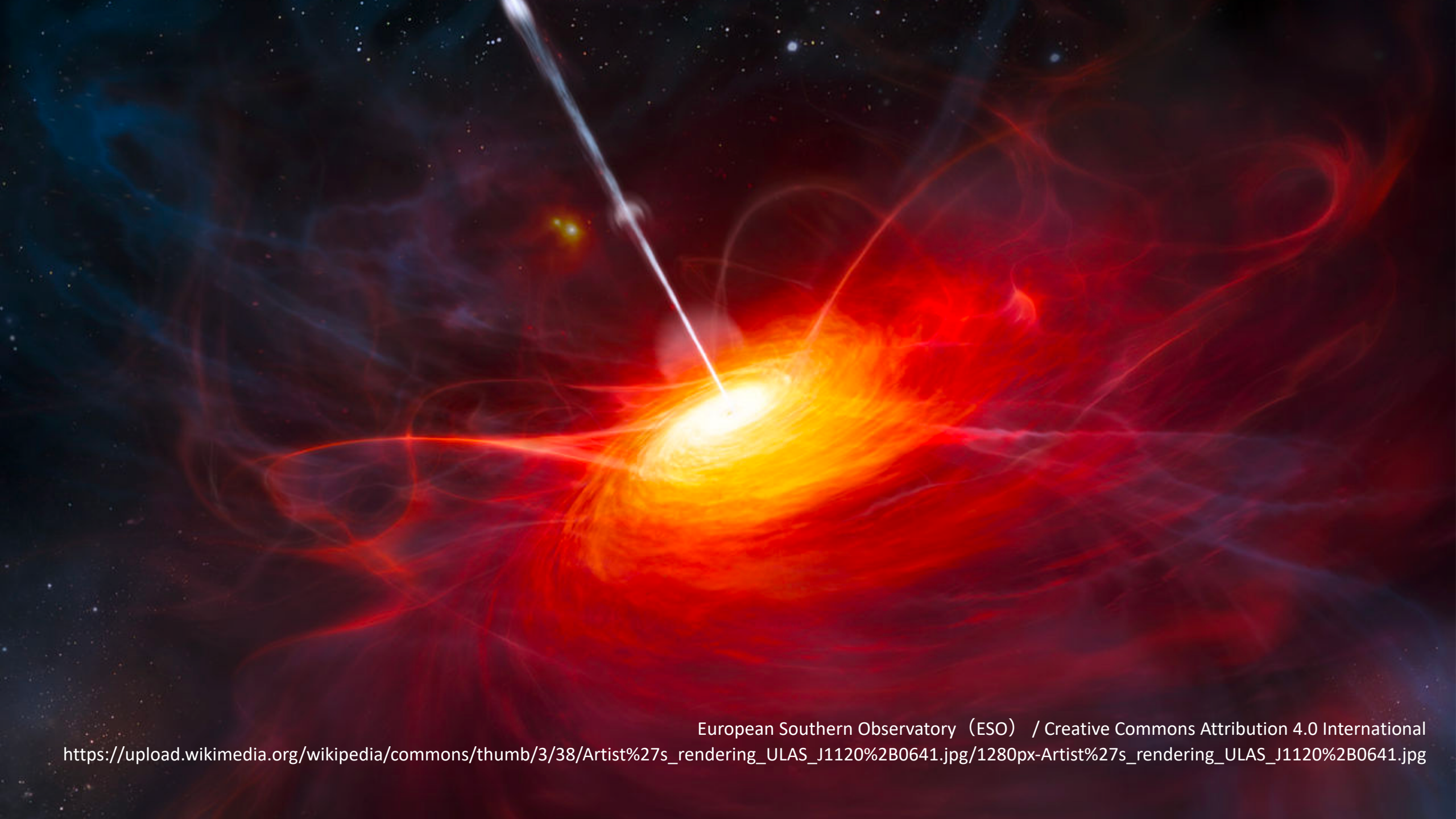
- **Auroraの特徴**
- **Auroraのアーキテクチャ**

話さないこと

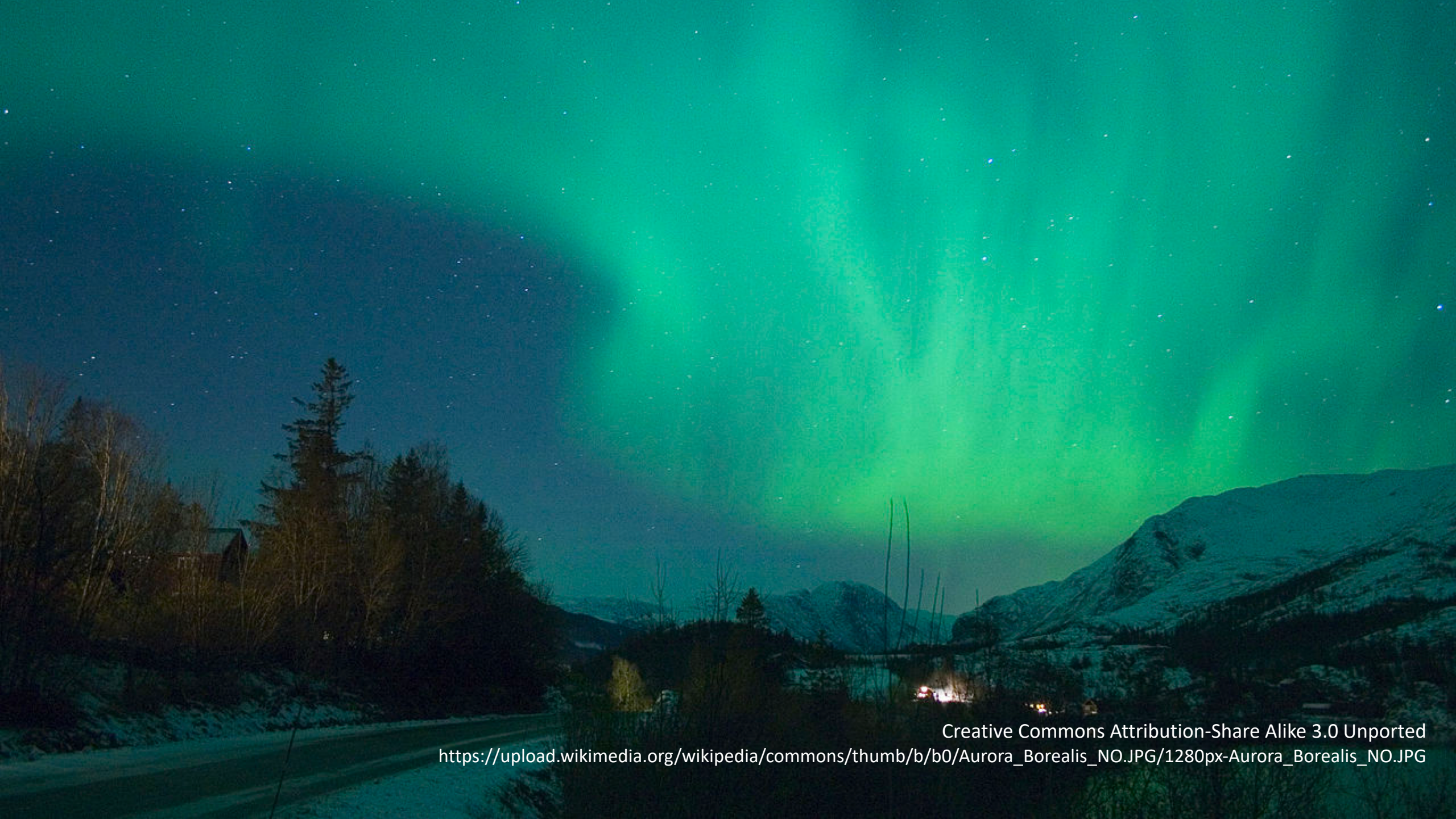
- **Aurora以外のRDSのこと**
- **Auroraの使いこなし方**

Auroraの特徴

Wrench, spanner PNG image http://pngimg.com/download/3124



European Southern Observatory（ESO） / Creative Commons Attribution 4.0 International
https://upload.wikimedia.org/wikipedia/commons/thumb/3/38/Artist%27s_rendering_ULAS_J1120%2B0641.jpg/1280px-Artist%27s_rendering_ULAS_J1120%2B0641.jpg



Auroraの話をします

re:Invent 2014で発表

AWS史上最も早く成長

MySQLとPostgreSQL

クラウド時代にAmazonが再設計したRDBMS

Auroraの特徴

- 優れたパフォーマンスとスケーラビリティ
- 高い可用性と耐久性
- 高い安全性
- MySQL および PostgreSQL との互換性
- 完全マネージド型
- 移行サポート

優れたパフォーマンスとスケーラビリティ

MySQLの最大5倍, PostgreSQLの最大3倍

ストレージの自動スケーリング

低レイテンシのリードレプリカ

高い可用性と耐久性

インスタンスのモニタリングと修復

マルチAZとレプリカ

耐障害性と自己修復機能を備えたストレージ

99.999999999%の堅牢性

Auroraの特徴

高い安全性

- ネットワークの隔離
- リソースレベルのアクセス許可
- 暗号化
- 高度な監査

フルマネージド型

- 使用が簡単
- モニタリングとメトリクス
- ソフトウェアの自動パッチ適用
- DBイベントの通知

移行サポート

標準の移行ツールが使用可能

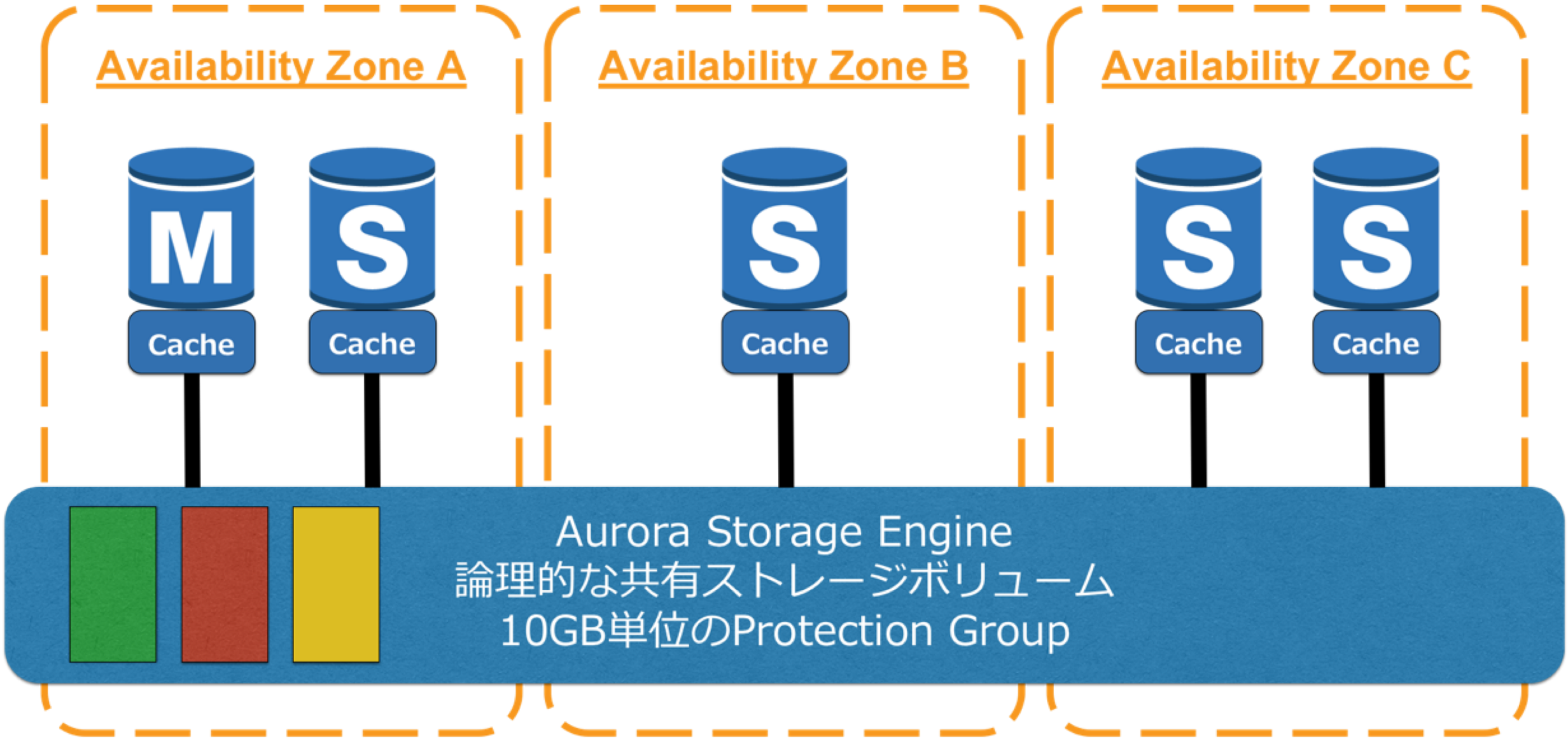
DMSによる移行

商用データベースの移行

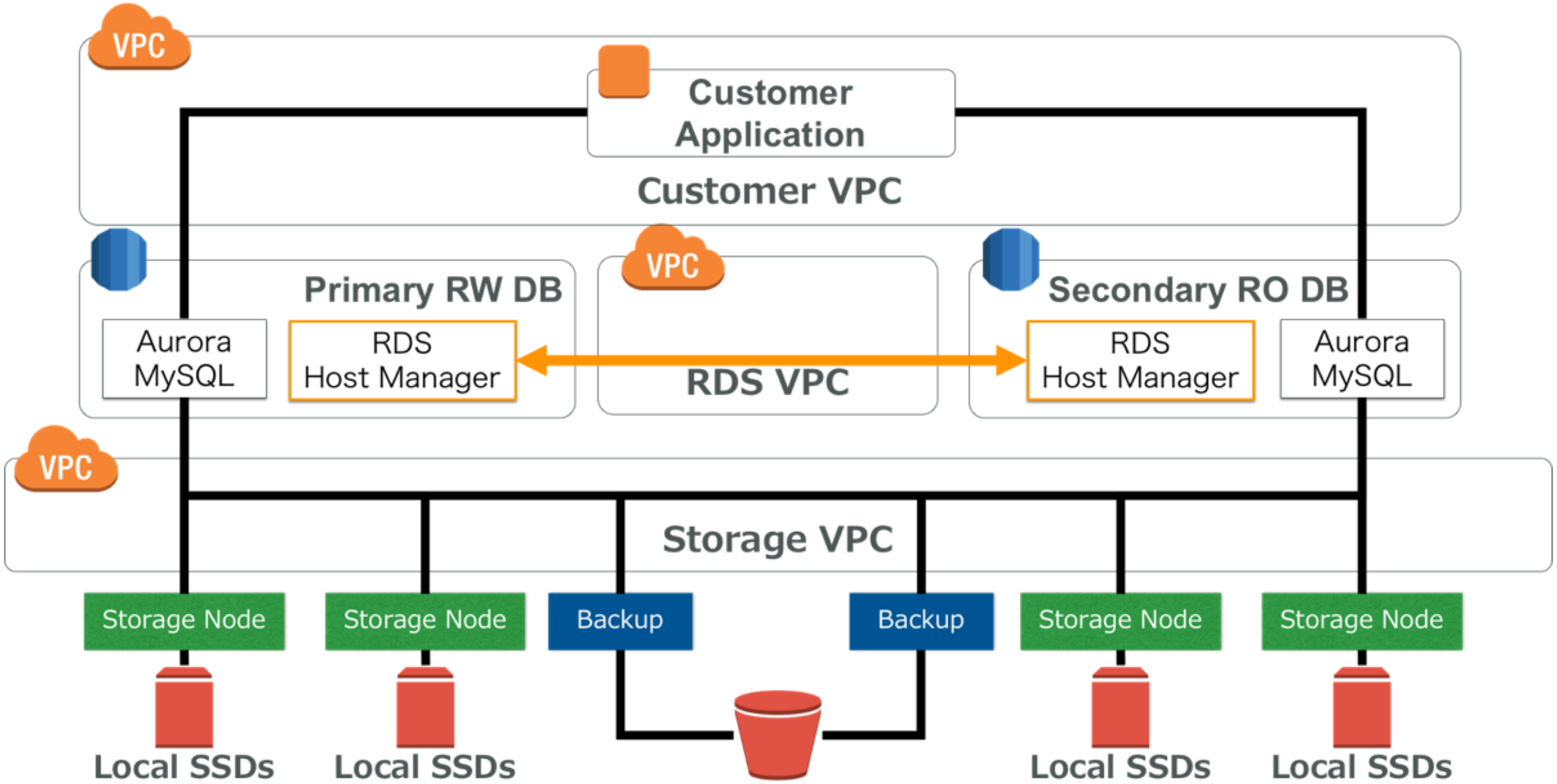
低コスト

実際に使用した分のみ料金が発生

Auroraの全体概要



Auroraの全体概要



Well-Architectedに学ぶAuroraの凄さ

Well-Architected Frameworkとは、クラウドアーキテクチャ設計・運用の考え方とベストプラクティス

設計原則と5つの柱で構成されています。

5つの柱の観点でAuroraの凄さを紹介してみます。

Well-Architected Frameworkでは以下の5つの柱があります。

- 信頼性
- パフォーマンス効率
- コストの最適化
- セキュリティ
- 運用上の優秀性

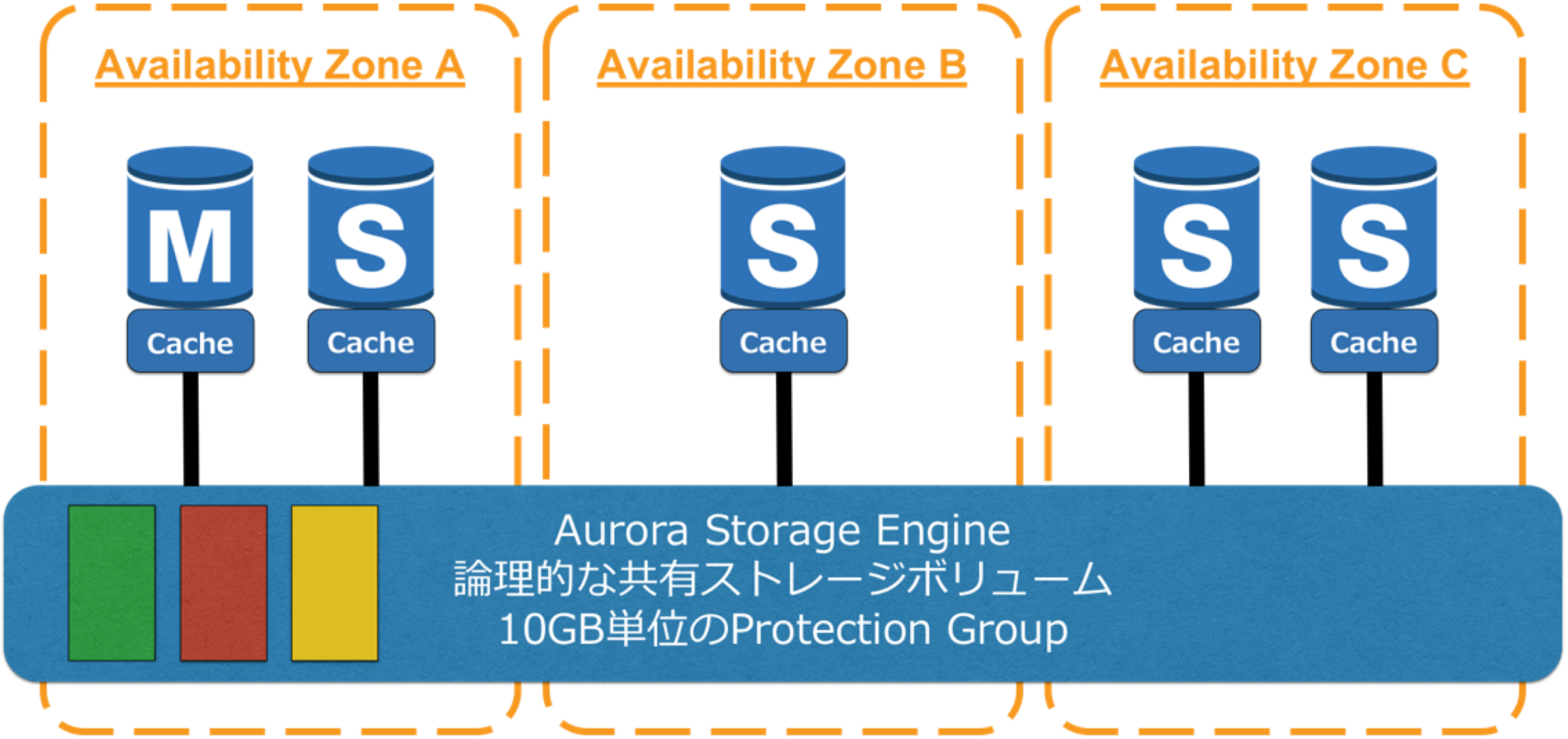
信頼性の柱

Auroraの信頼性の要は6重化された分散ストレージ

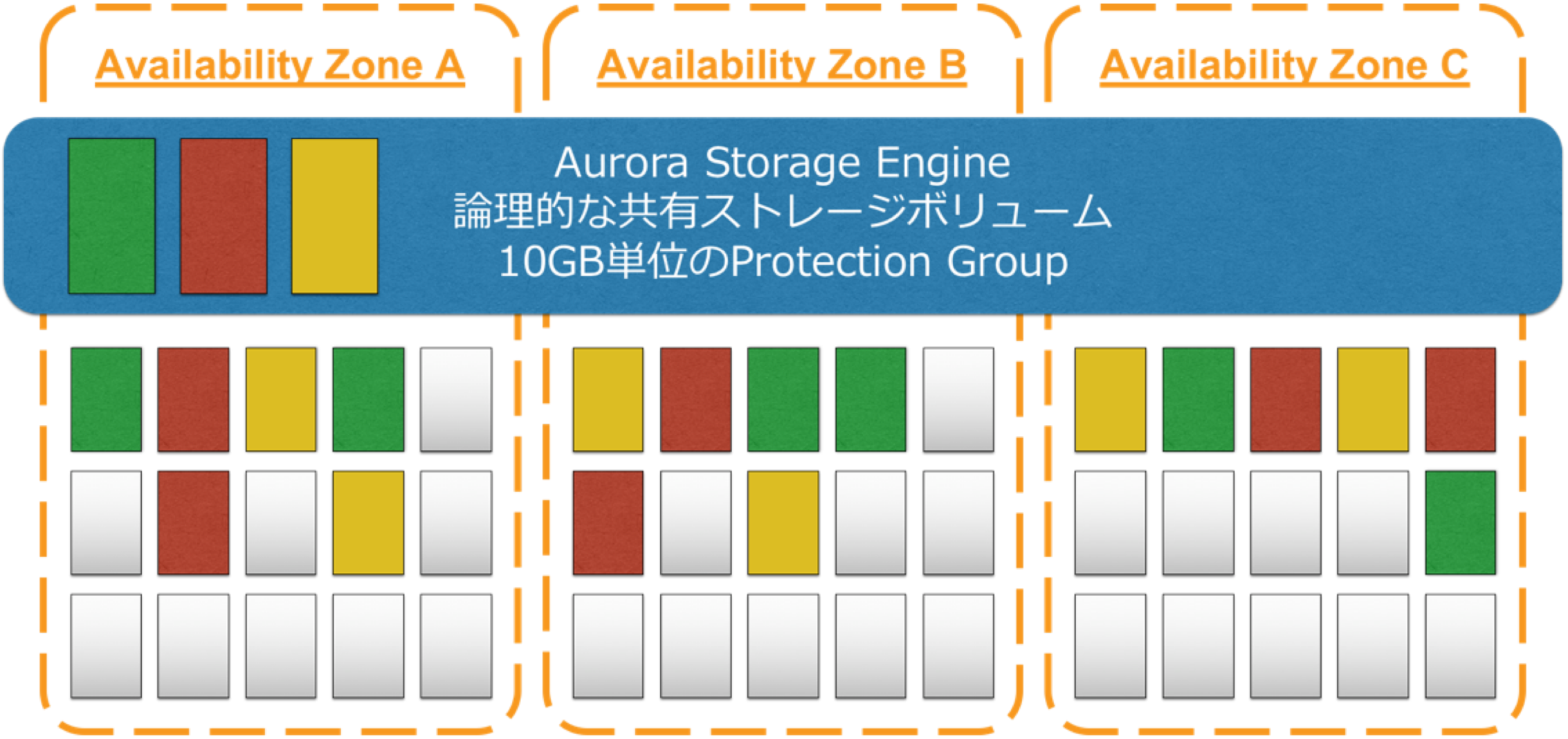
- 1つのAvailability Zone(データセンター郡)に2個のデータ、3AZに跨って合計6個のデータを保存
- ストレージがDBインスタンスから独立
- 全データが追記型になっているLog-Structured Storage

Auroraの凄いところは大体ストレージに集約されます

信頼性


Availability Zone A
Availability Zone B
Availability Zone C
M
S
S
S
S
Cache
Cache
Cache
Cache
Cache
Aurora Storage Engine
論理的な共有ストレージボリューム
10GB単位のProtection Group

信頼性


Availability Zone A
Availability Zone B
Availability Zone C
Aurora Storage Engine
論理的な共有ストレージボリューム
10GB単位のProtection Group

なぜ6重化？

クォーラムモデル。多数決だと思って下さい。

実は障害が発生しても動き続けるには3重化で大丈夫

ルール１：書いた数Vwと読んだ数Vrの合計が全体の数V(3)より大きいと、書いたノードと読んだノードが必ず重複する
→ Vw + Vr > V

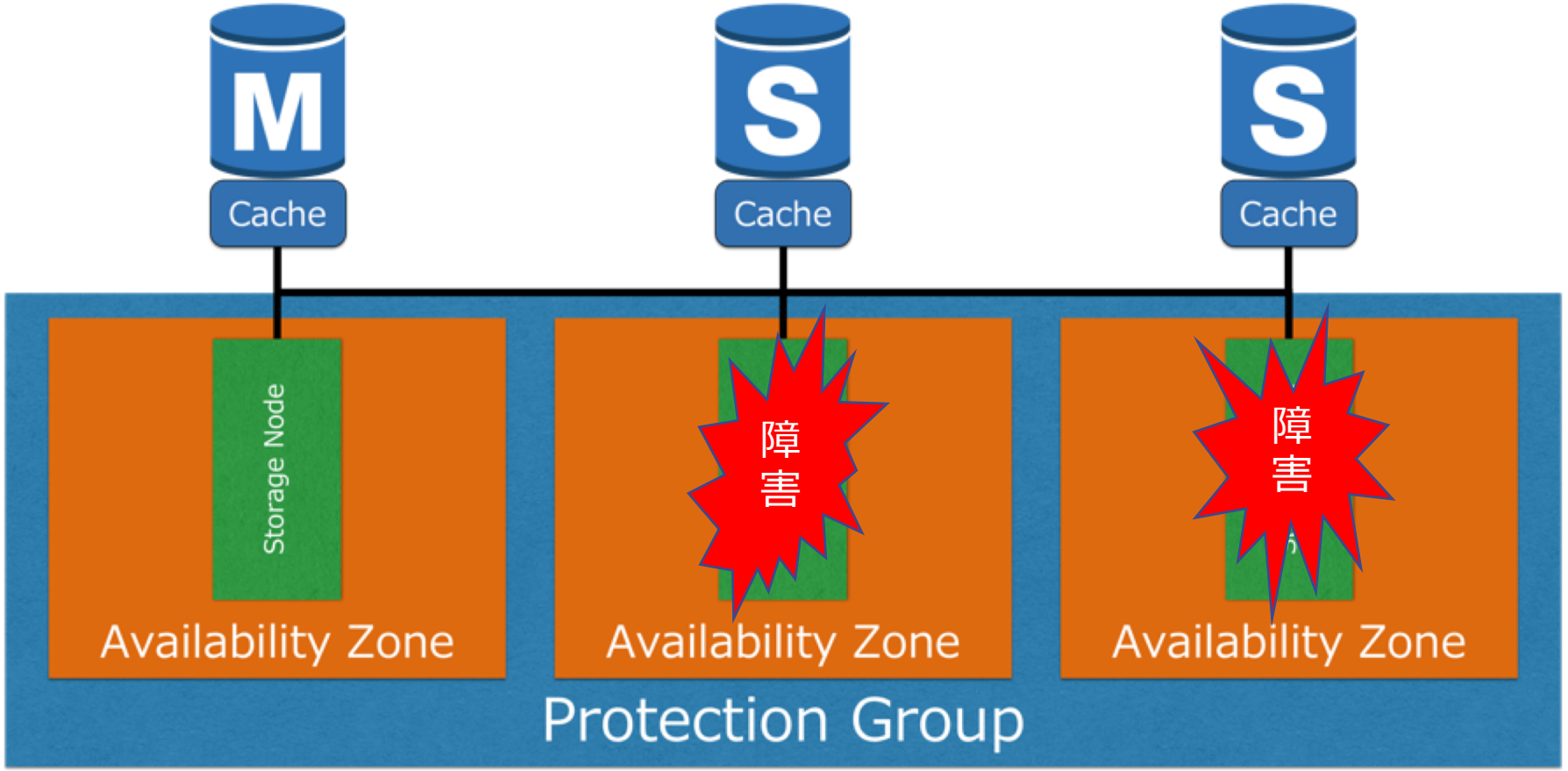
ルール２：全体に対して半分より多く書ける
→ Vw > V/2

障害が発生しても上記を満たす最小数は、

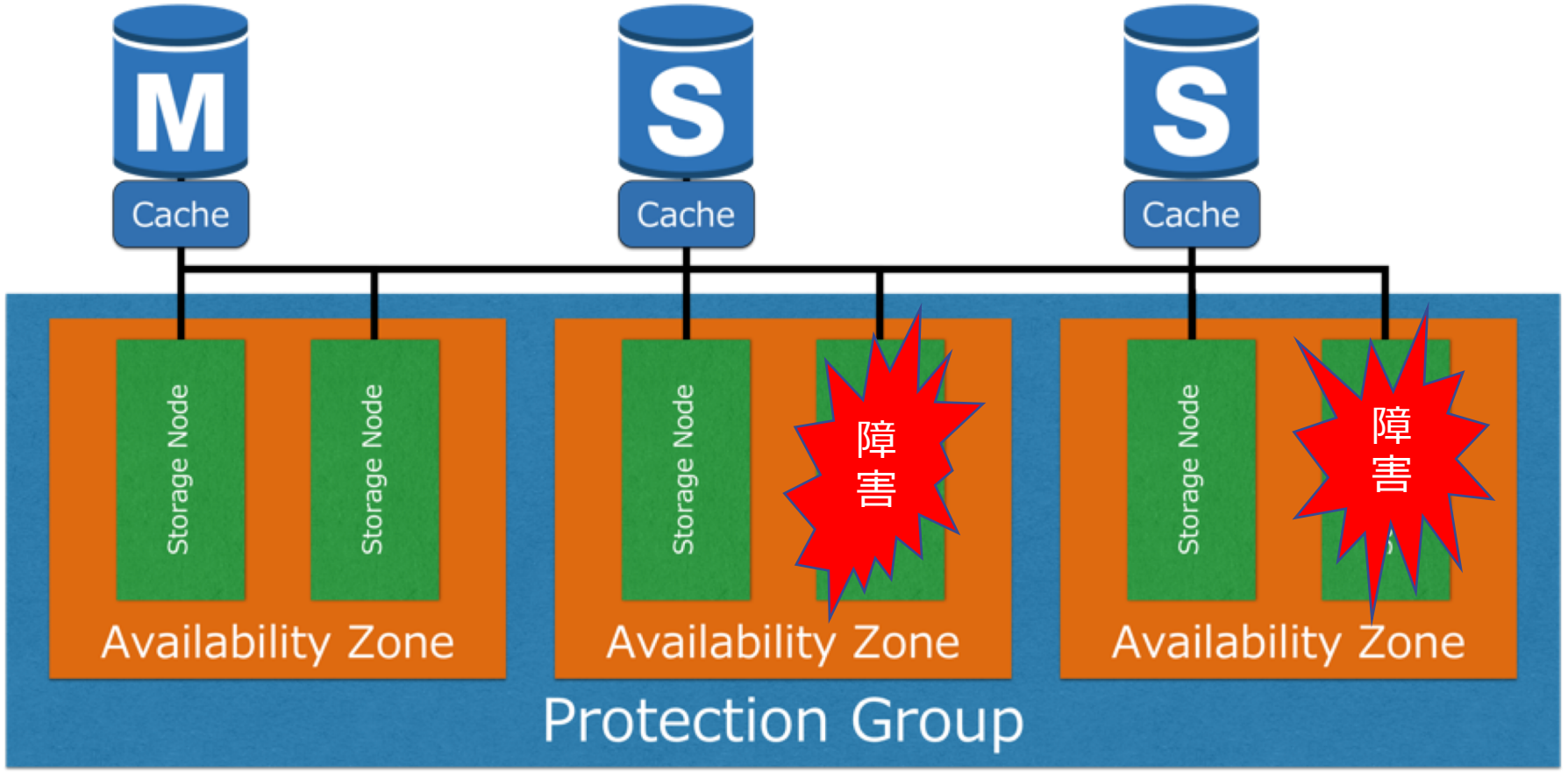
全体3、書き込み2、読み込み2

なぜ6重化？

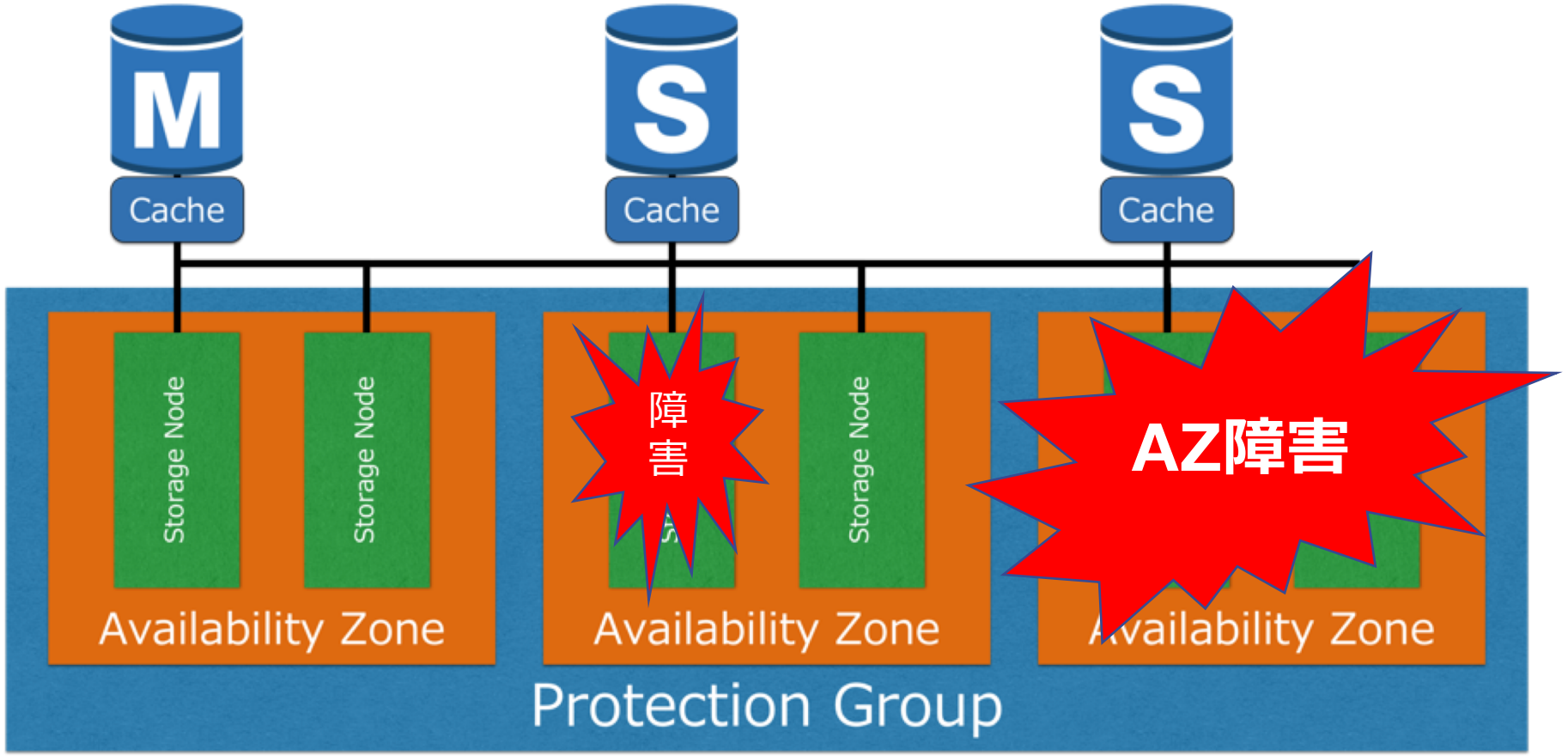
3重化だと多重障害やAZ停止のような大規模障害に耐えられないので3重化 × 2 で6重化を採用



多重障害でも6重化であれば耐えられる

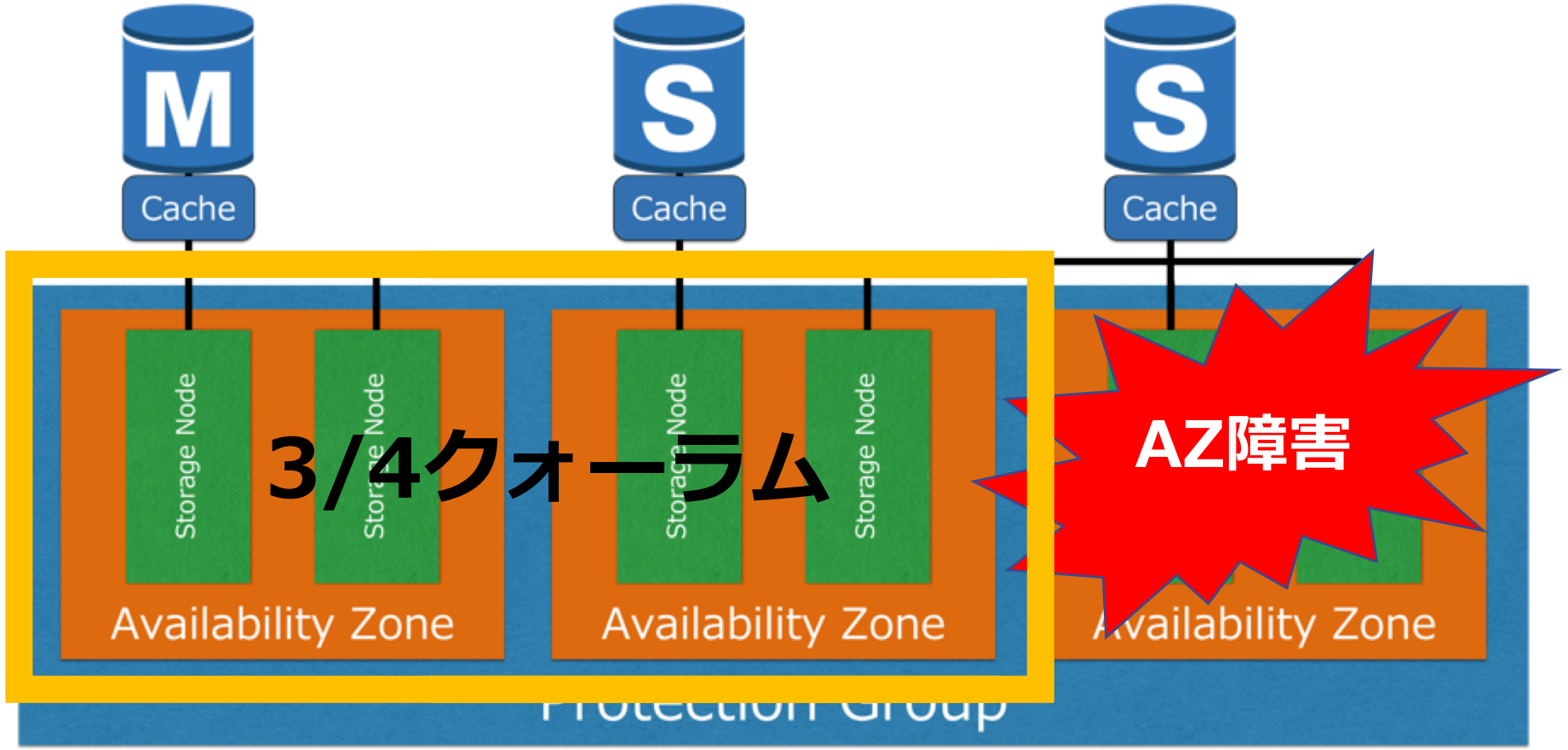


大規模障害でも6重化で事業継続が出来る



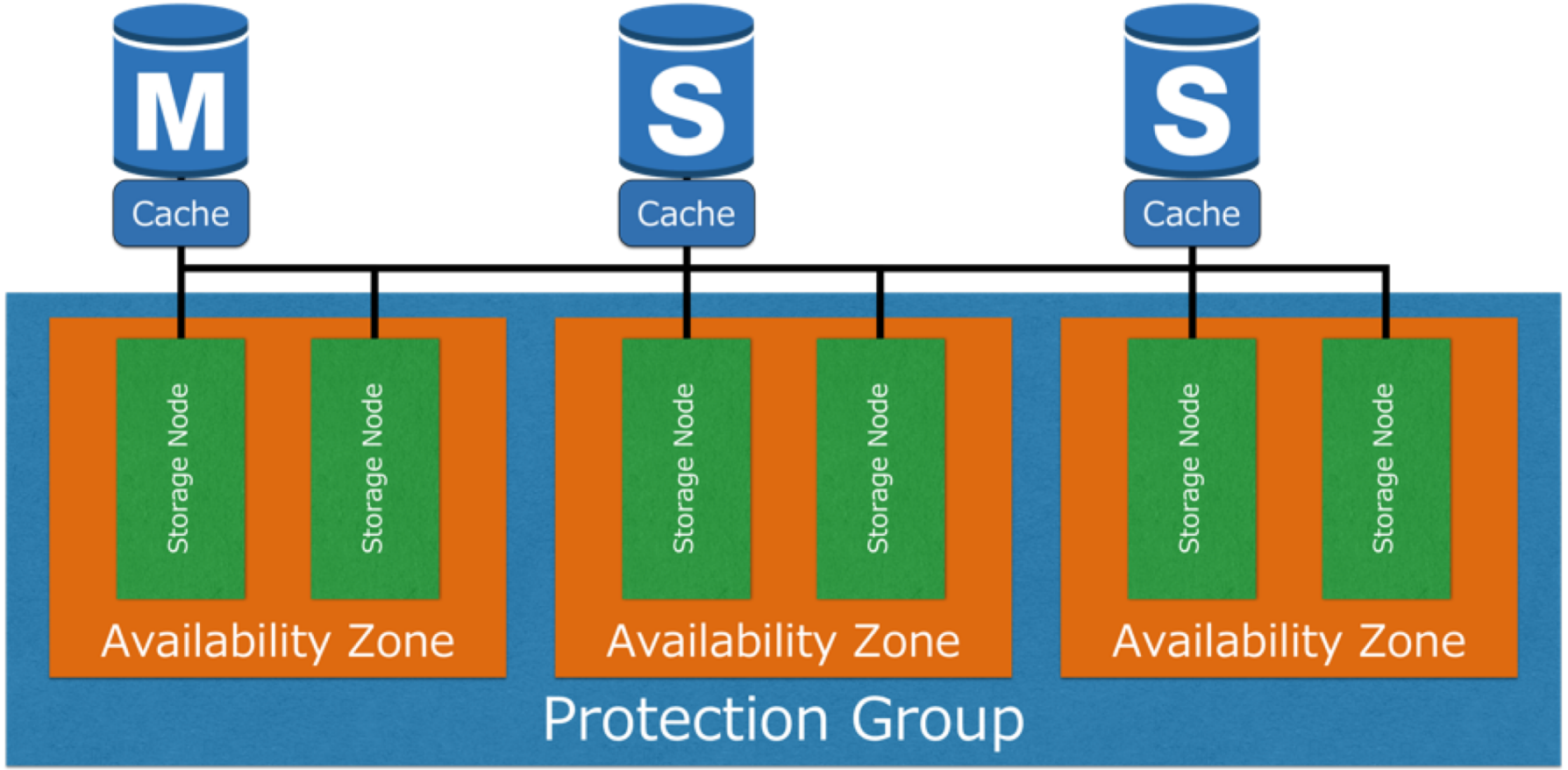
長期間の障害への対応

AZが長期間障害が続く場合はクォーラムから切り離し残りの4箇所でクォーラムを切り替える



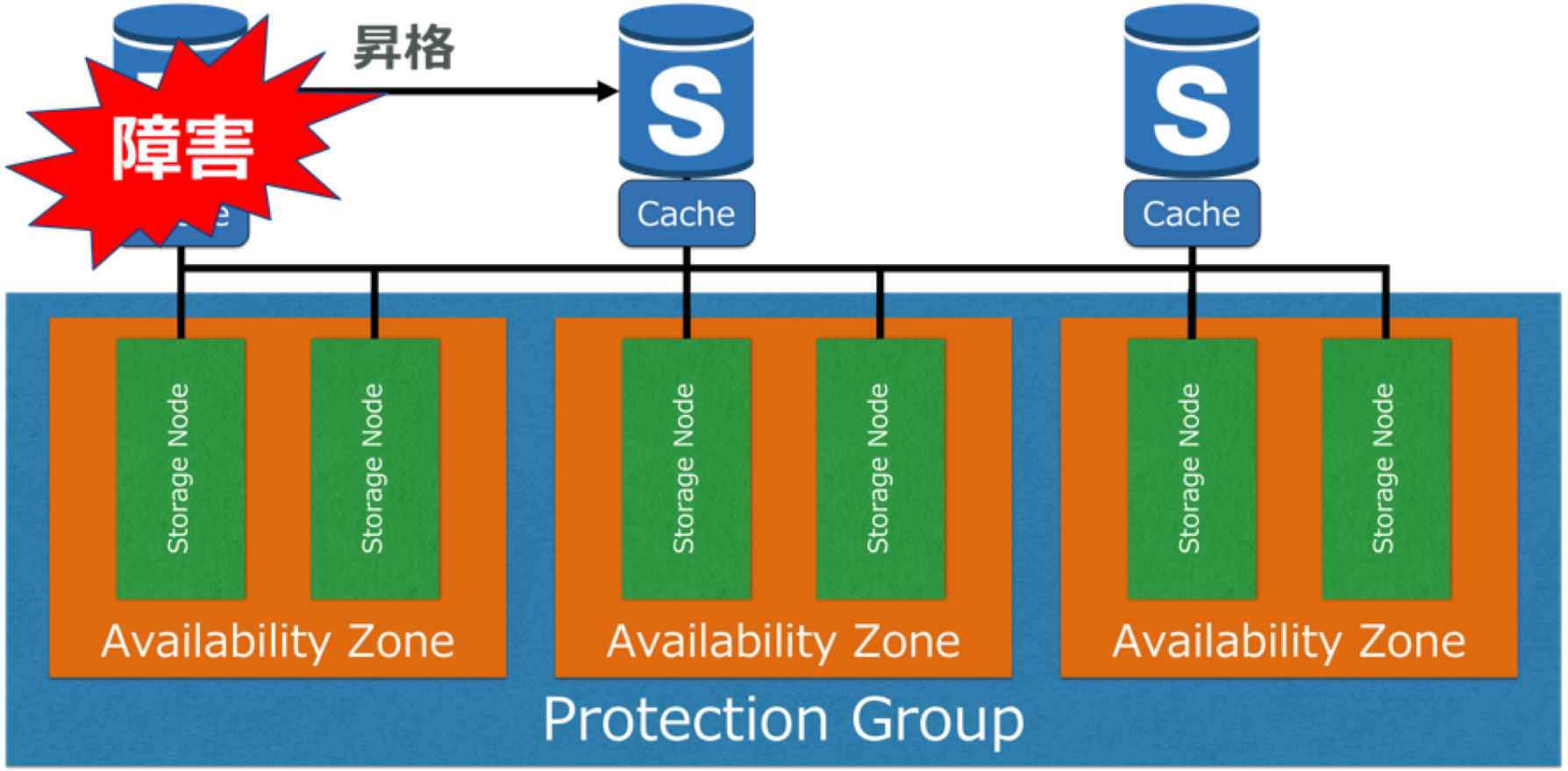
ストレージが独立

DBインスタンスがストレージから独立しているので
フェイルオーバーが高速(データリカバリが殆ど無い)



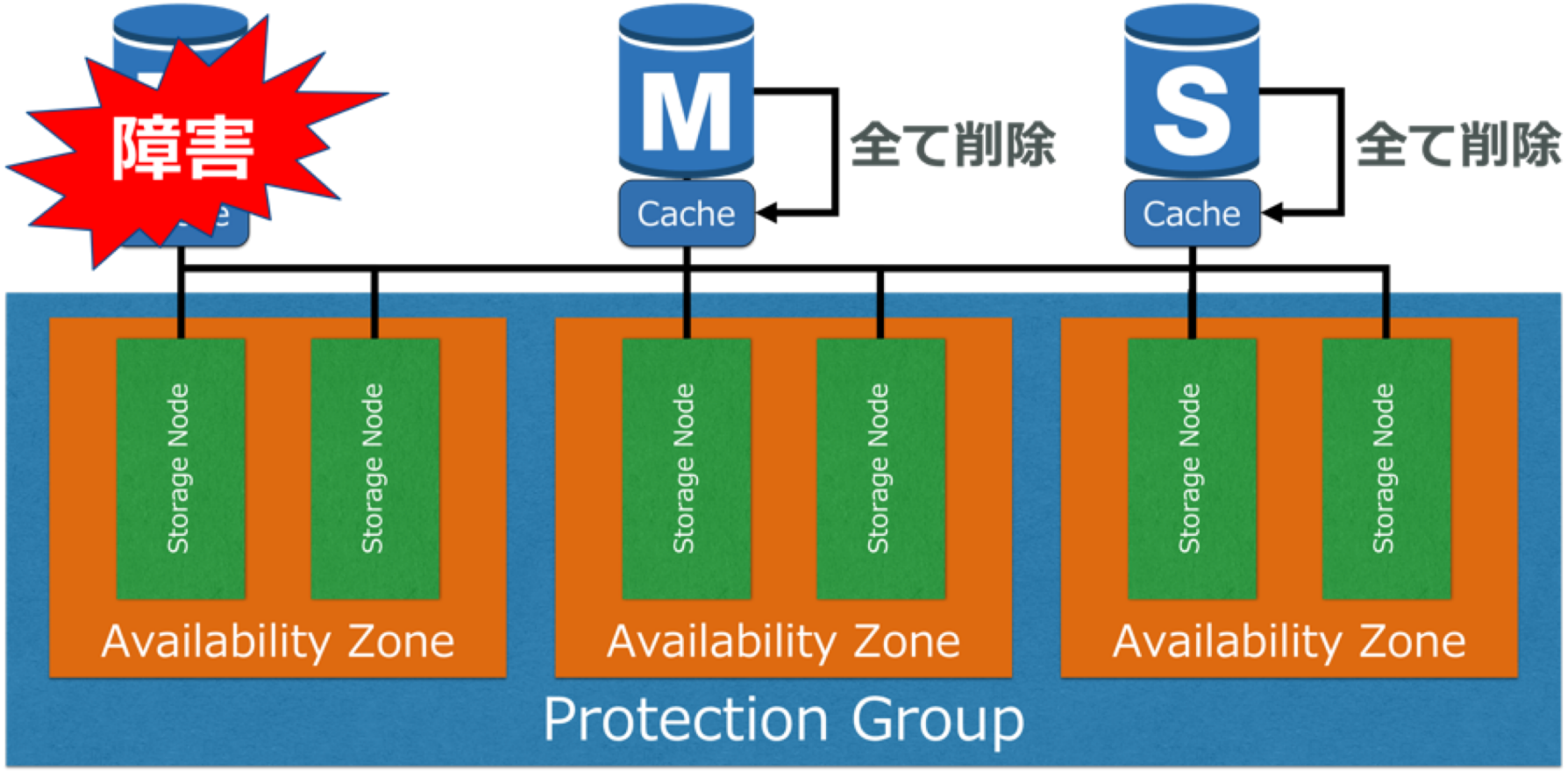
ストレージが独立

DBインスタンスがストレージから独立しているので
フェイルオーバーが高速(データリカバリが殆ど無い)



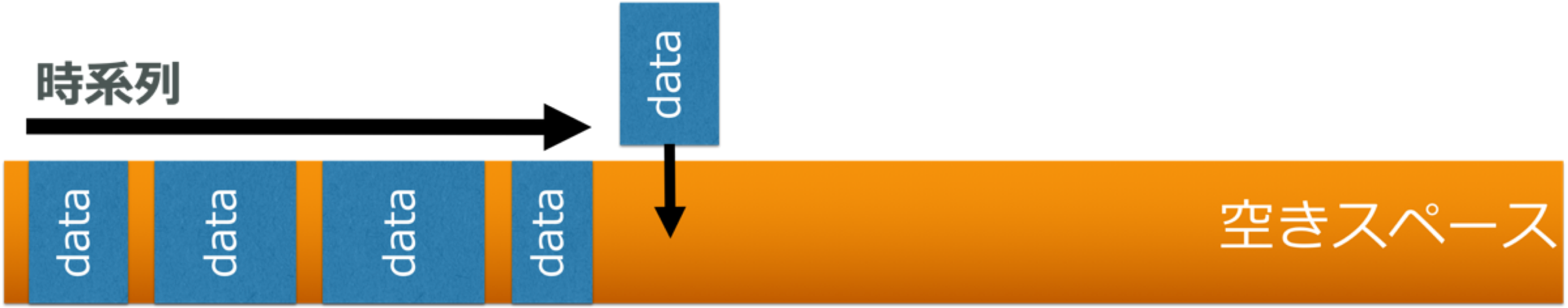
ストレージが独立

DBインスタンスがストレージから独立しているので
フェイルオーバーが高速(データリカバリが殆ど無い)



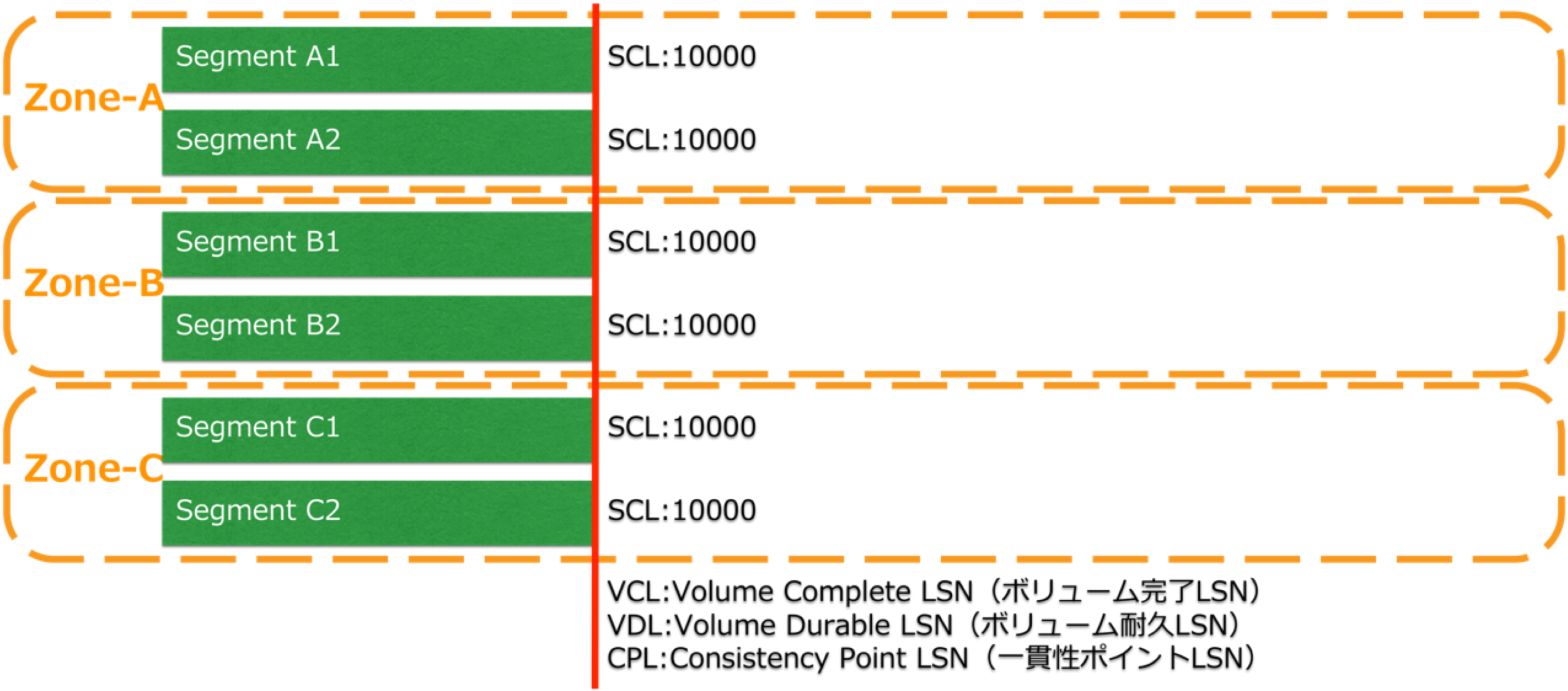
データをログのように先頭から追記する

分散ストレージとLog-Structured Storageを組合せ
安全に読み書きができたり、高速にリカバリが可能

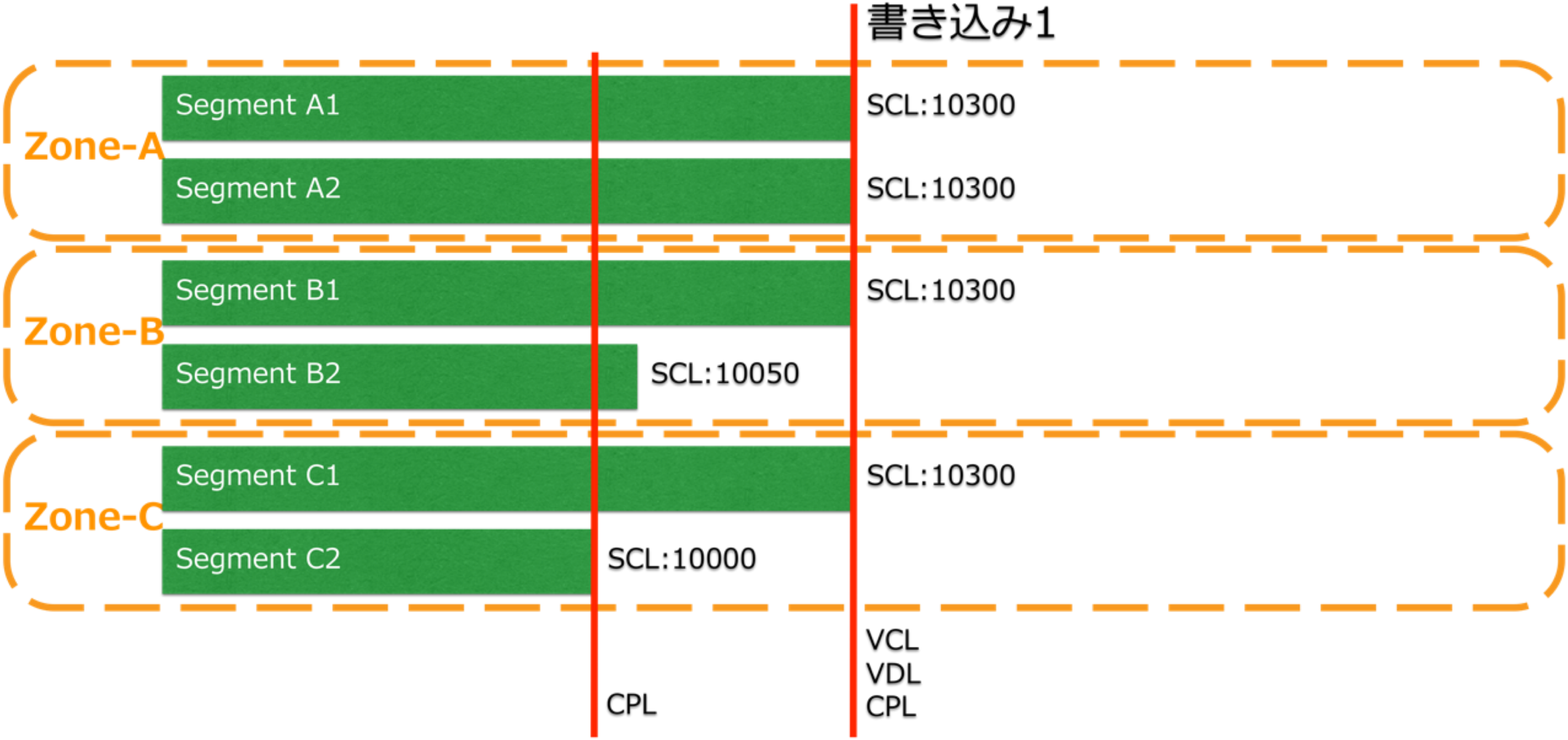


以降ストレージの動作説明が続きます
が全部すっ飛ばします！

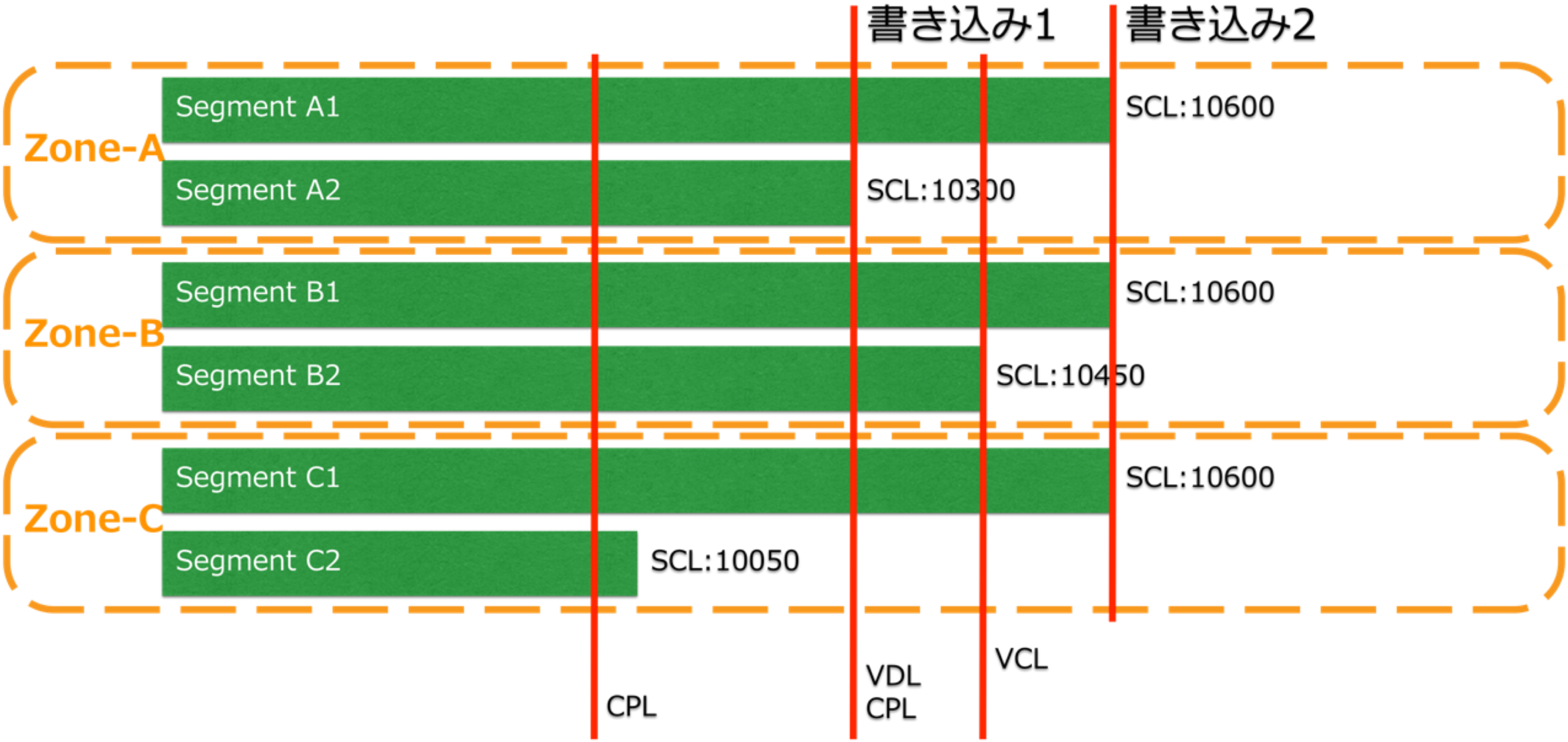
Log Structured Storage（Write）



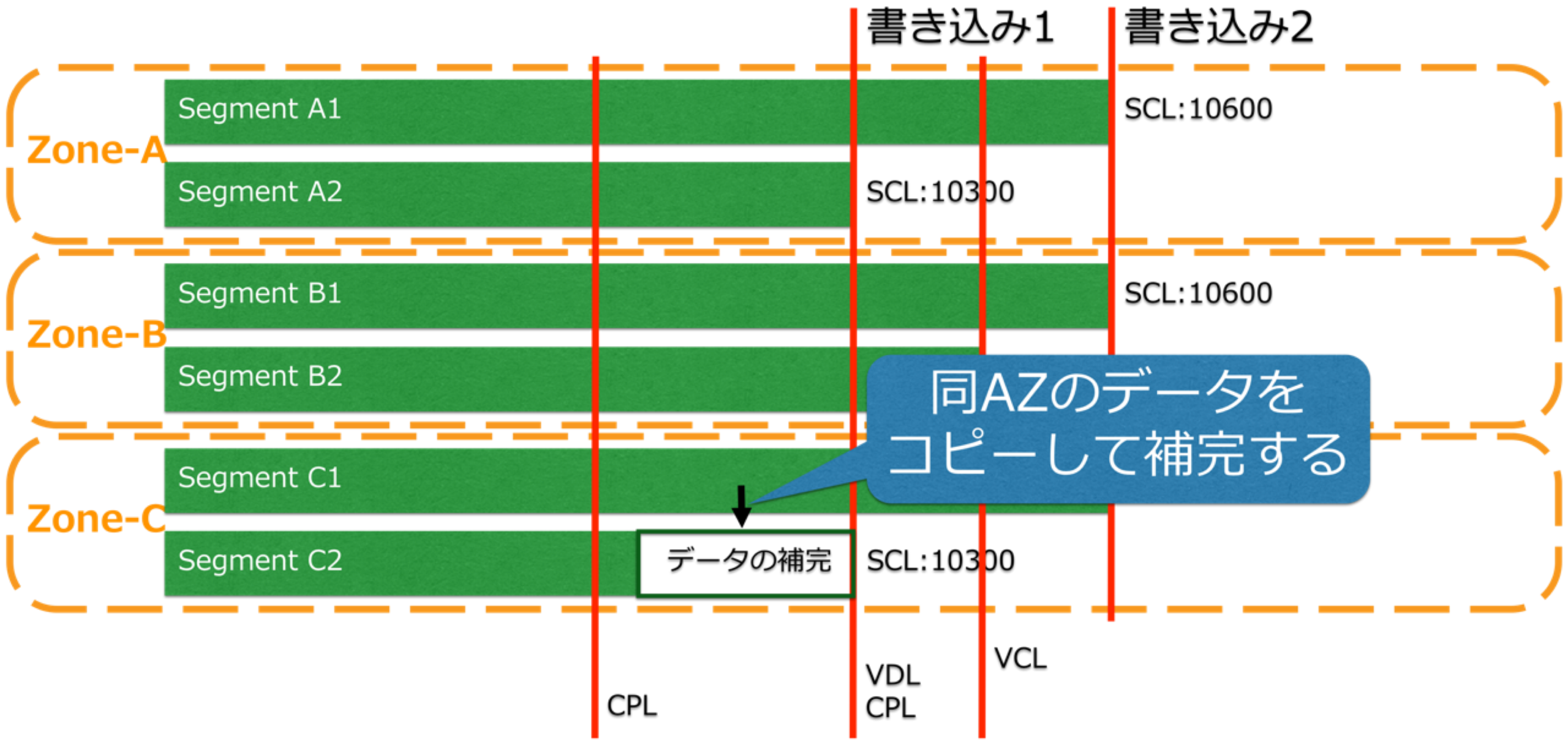
Log Structured Storage （Write）



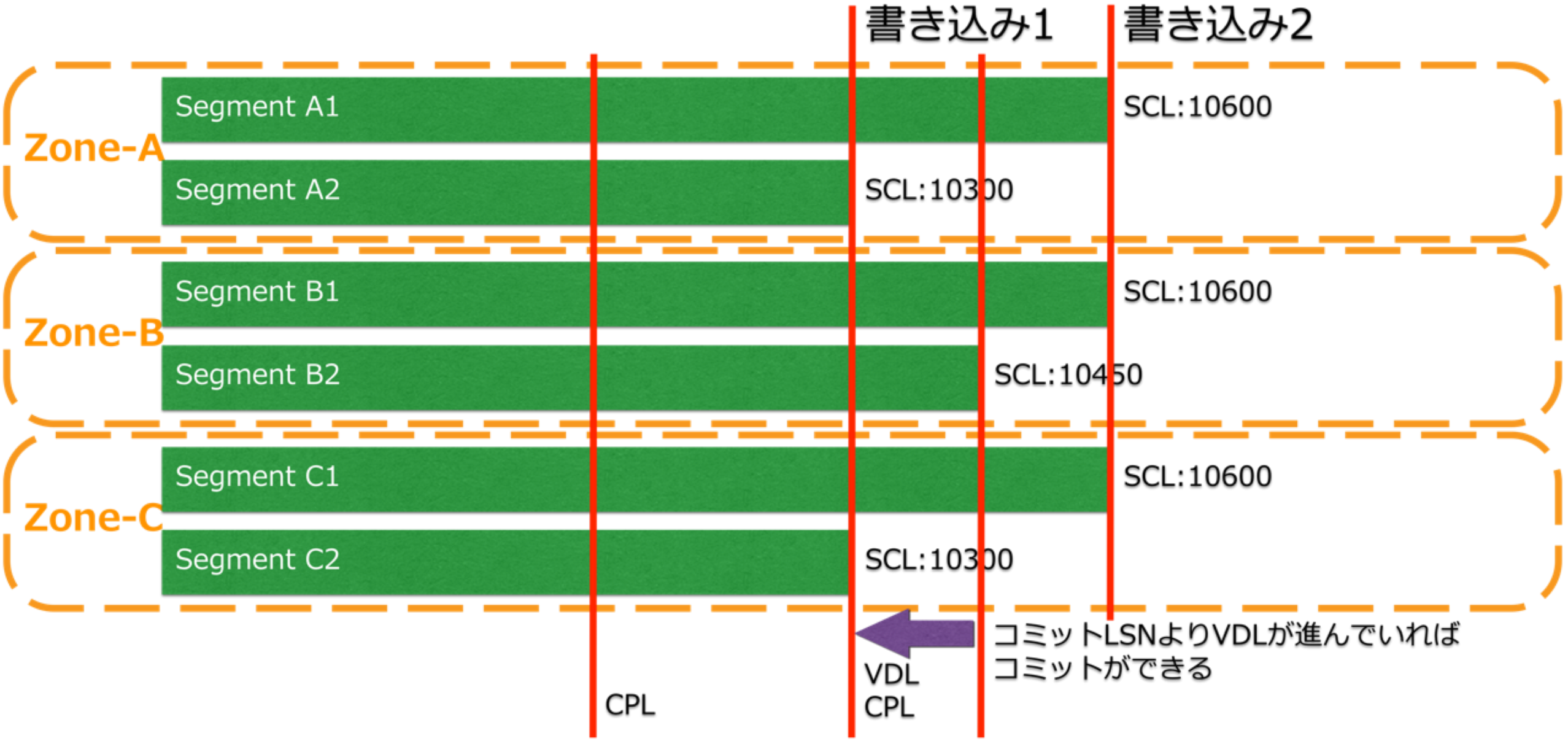
Log Structured Storage （Write）



Log Structured Storage （Write）

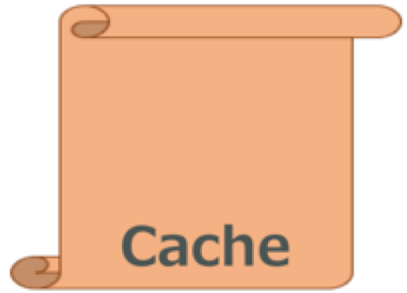


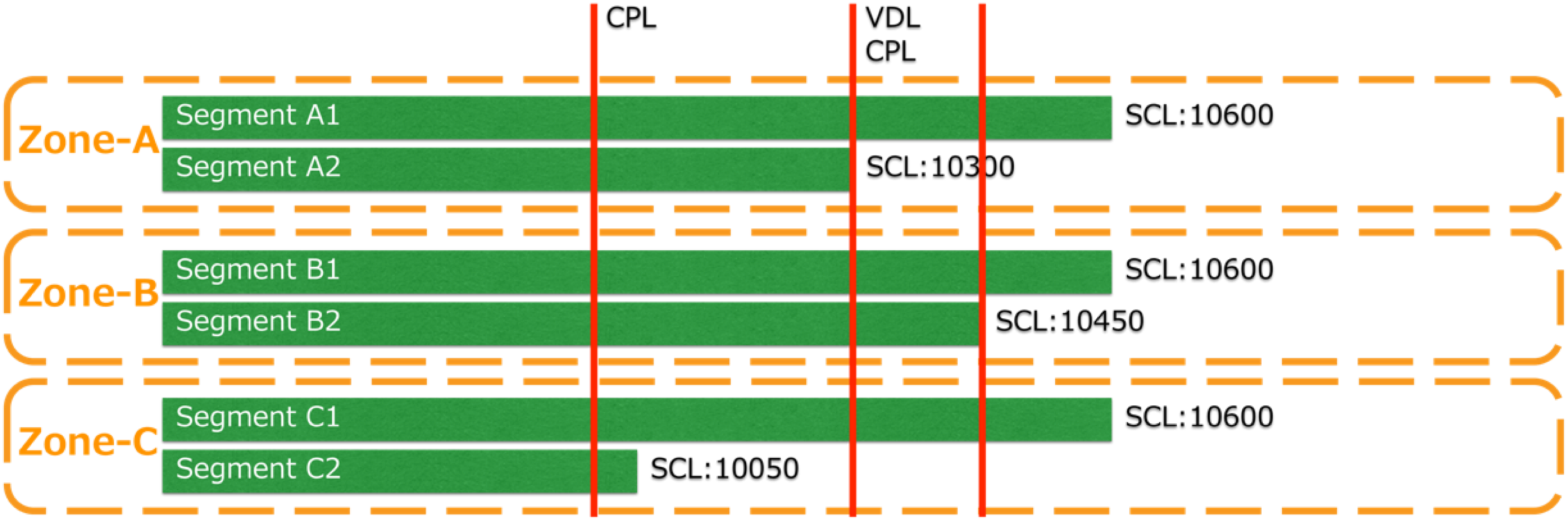
Log Structured Storage（Commit）



一般的なDBのページキャッシュ

普通はダーティページがあれば置き換え前に
ディスクにフラッシュ

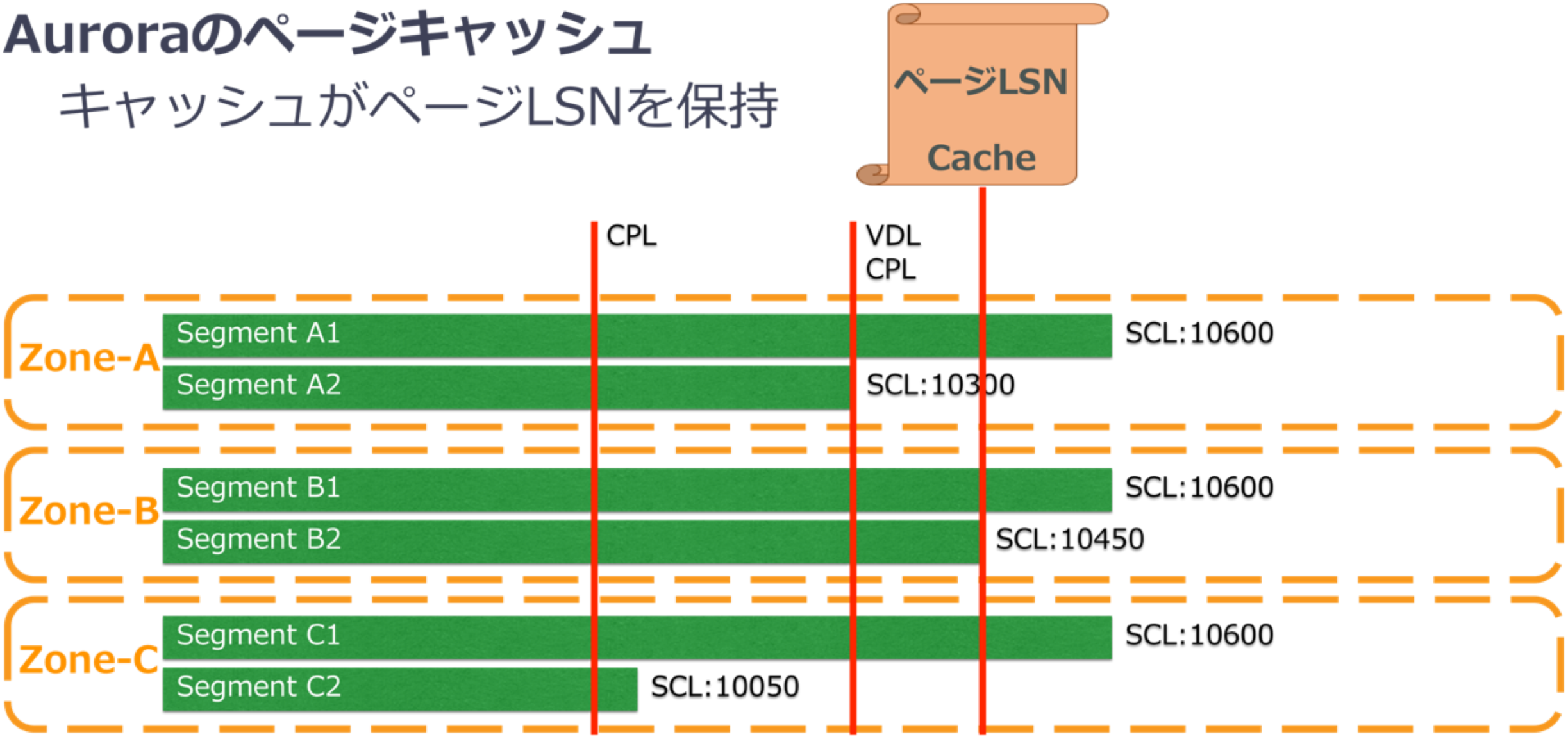




Log Structured Storage（Read）

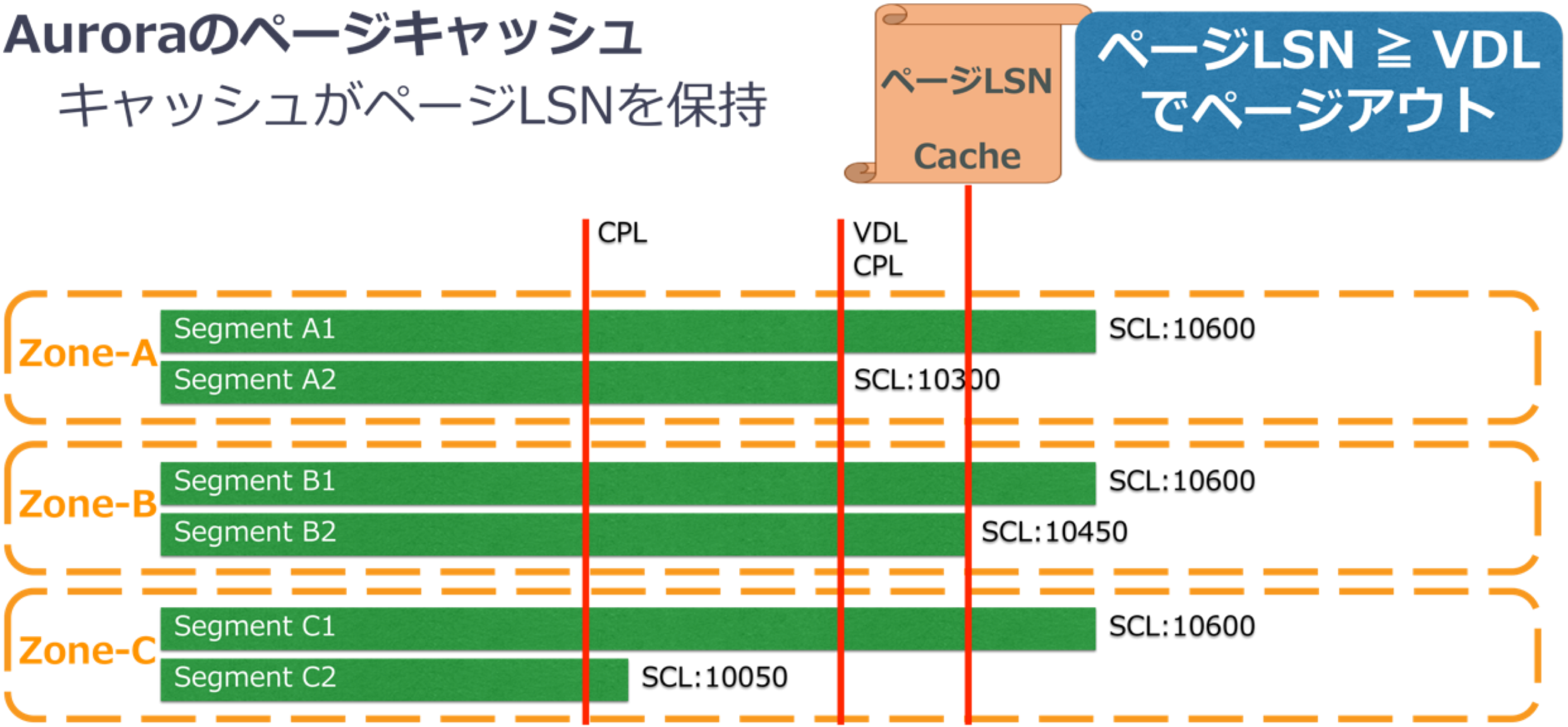
Auroraのページキャッシュ

キャッシュがページLSNを保持



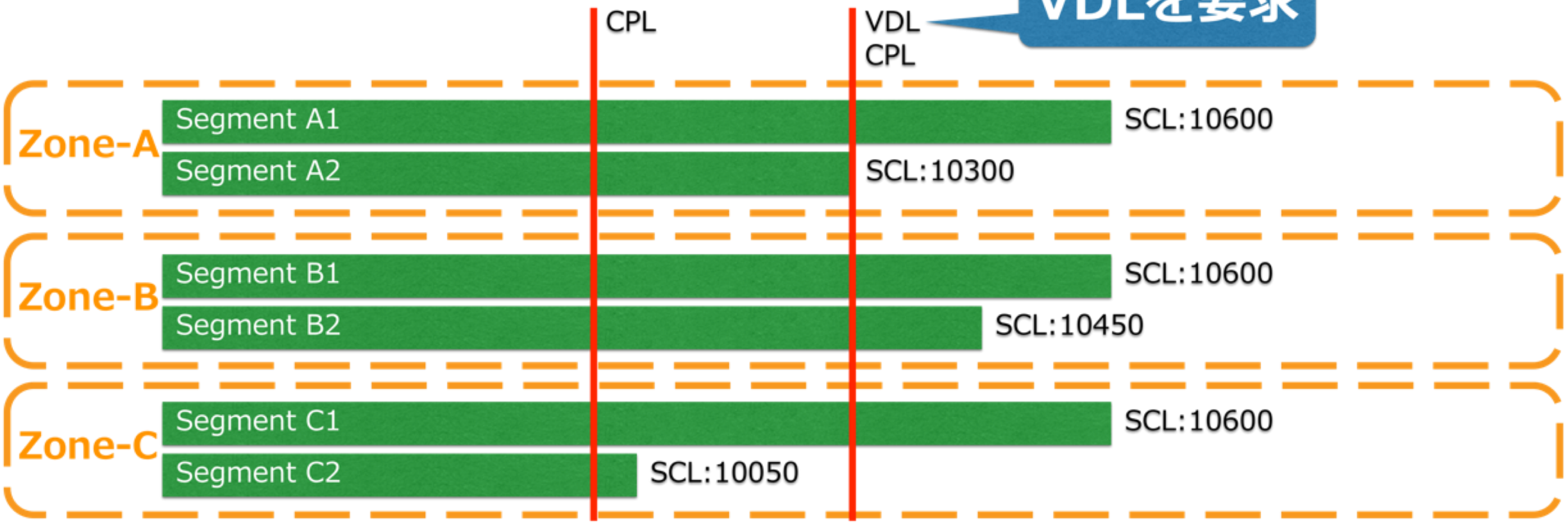
Auroraのページキャッシュ

キャッシュがページLSNを保持

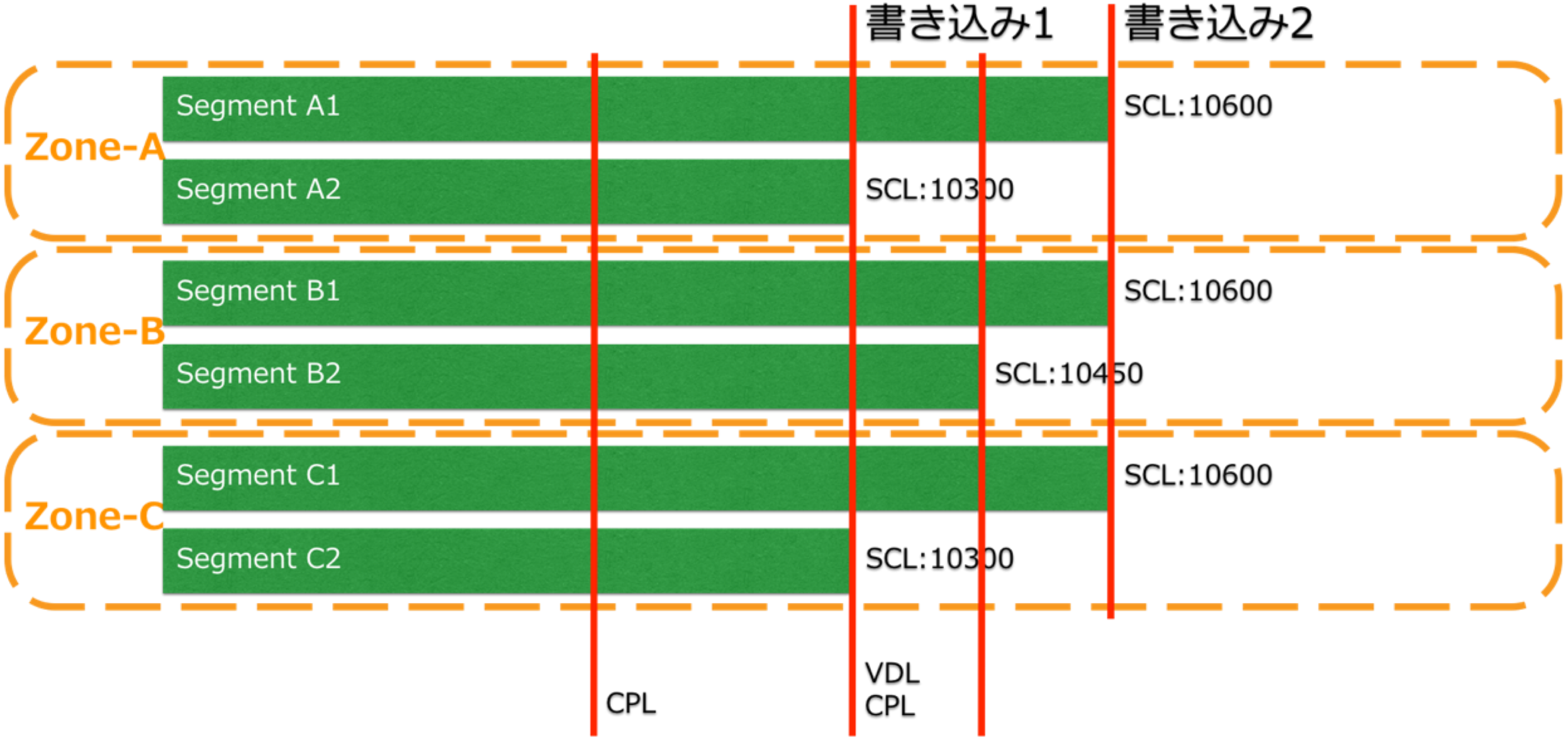


Auroraのページキャッシュ

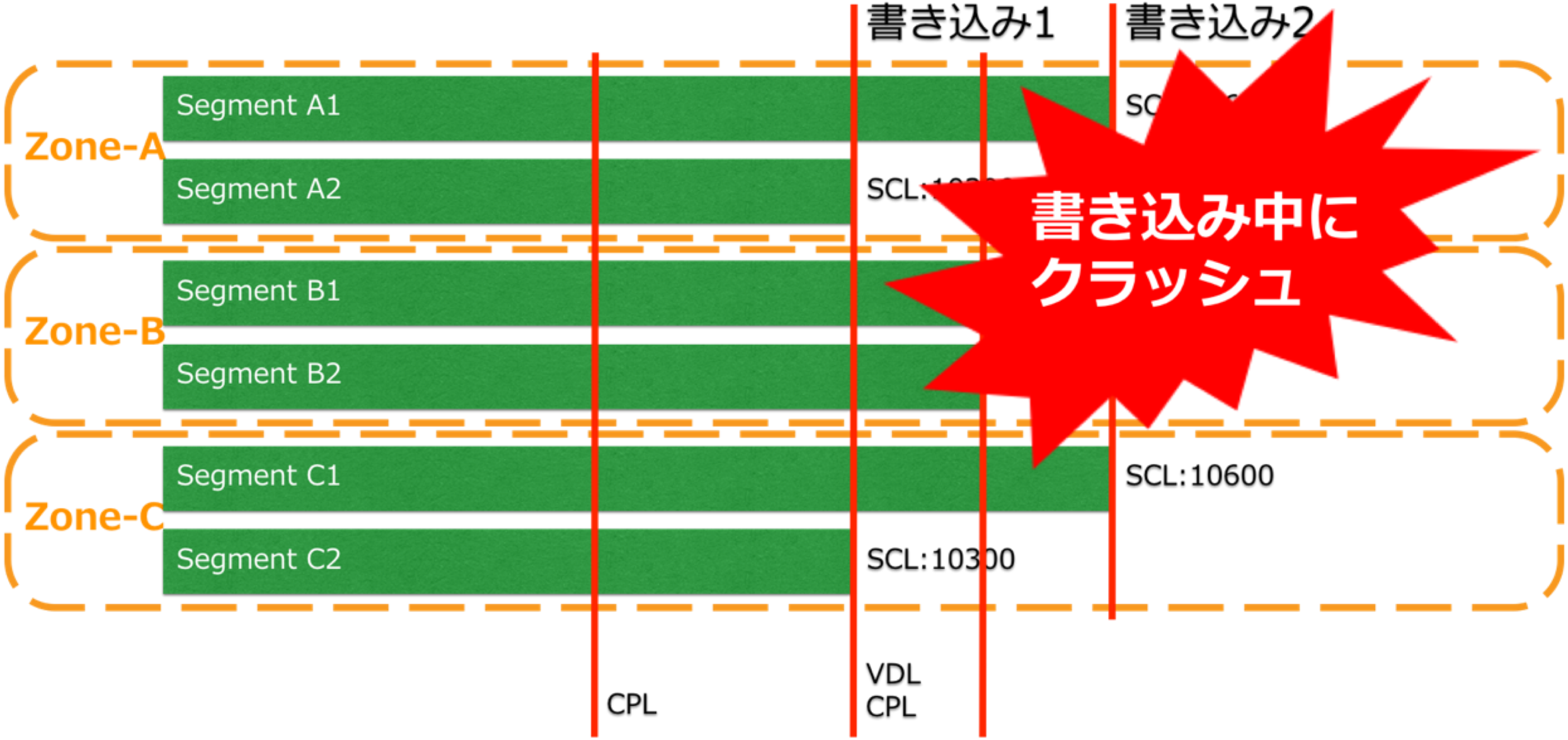
キャッシュミスの場合はVDLを要求



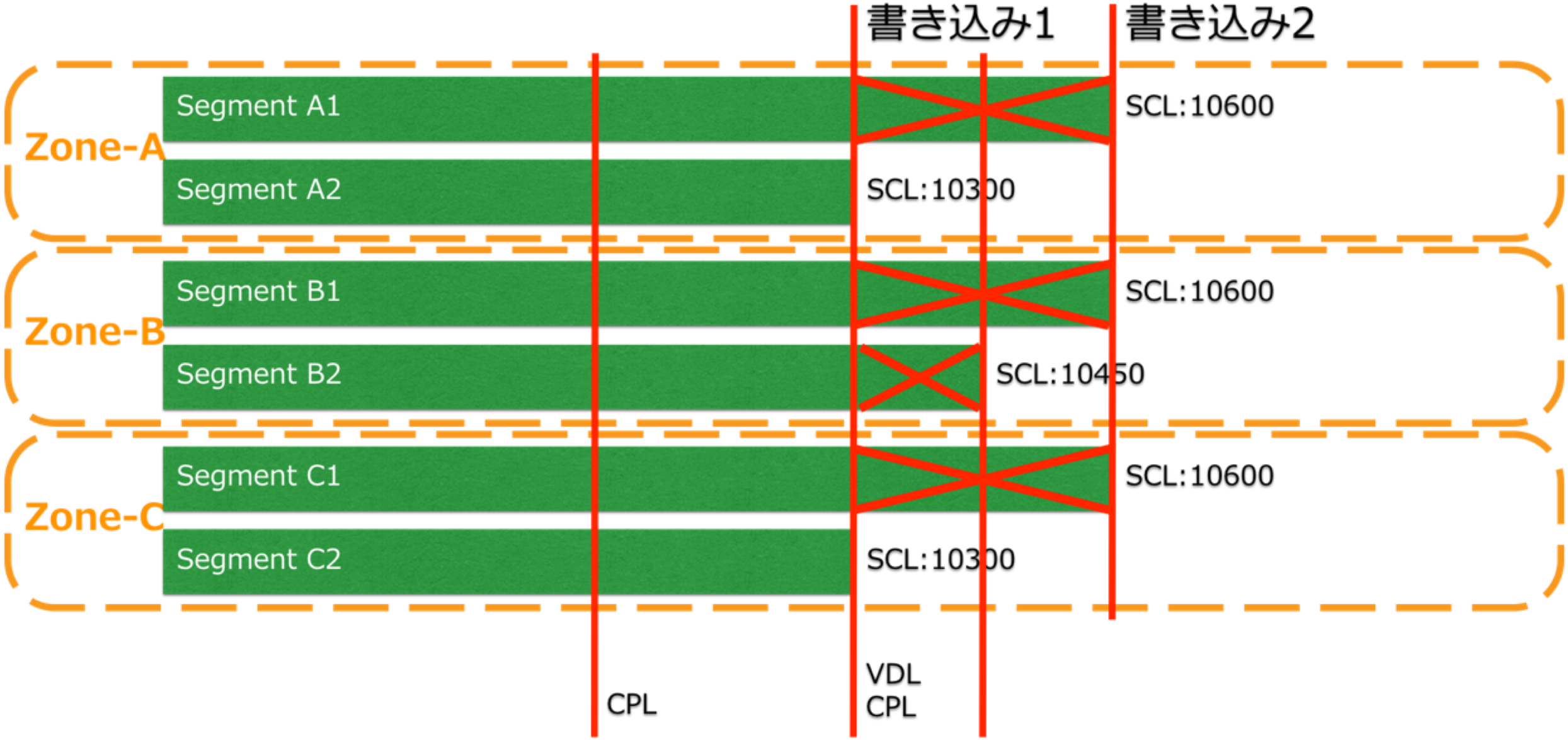
Log Structured Storage （Transaction Recovery）



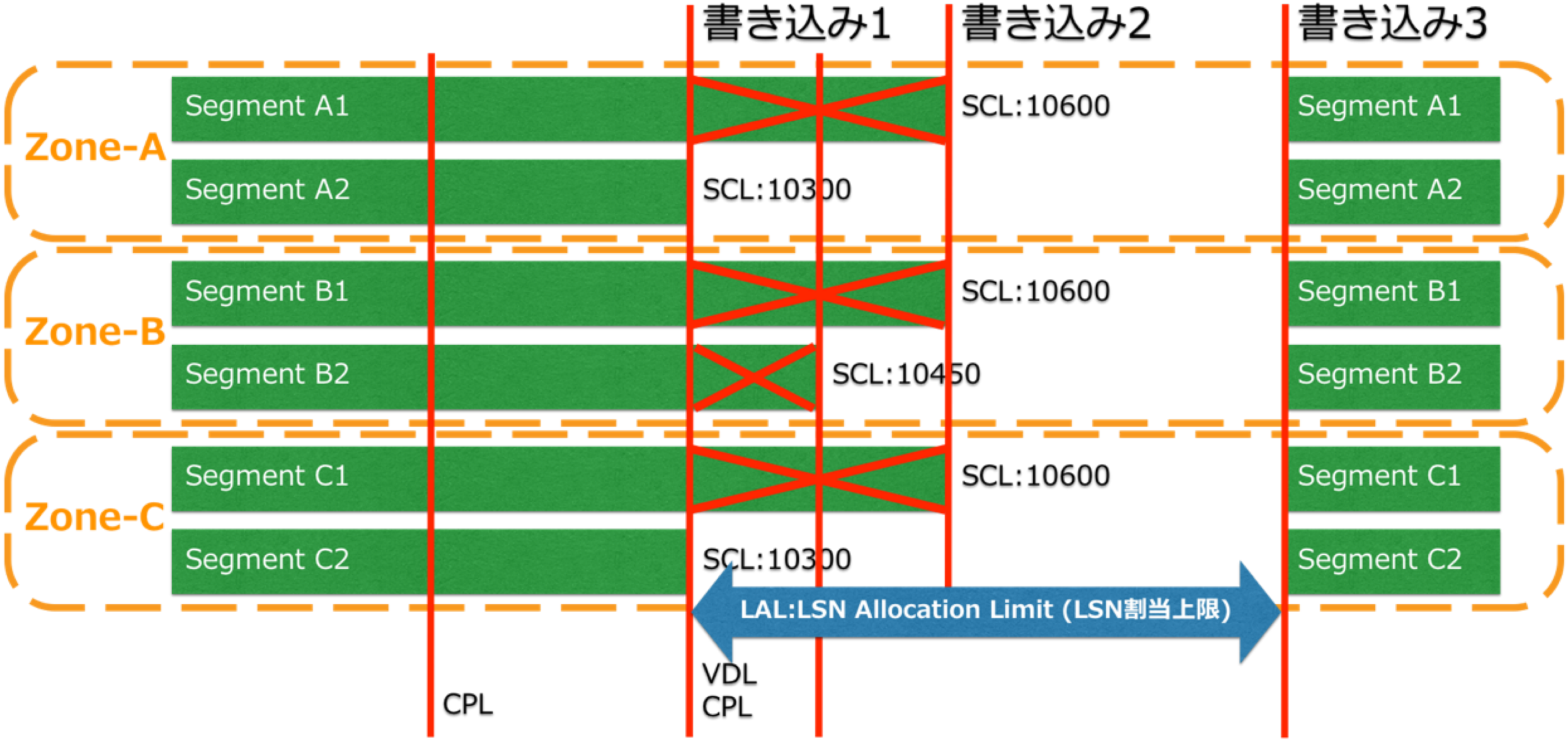
Log Structured Storage （Transaction Recovery） 46



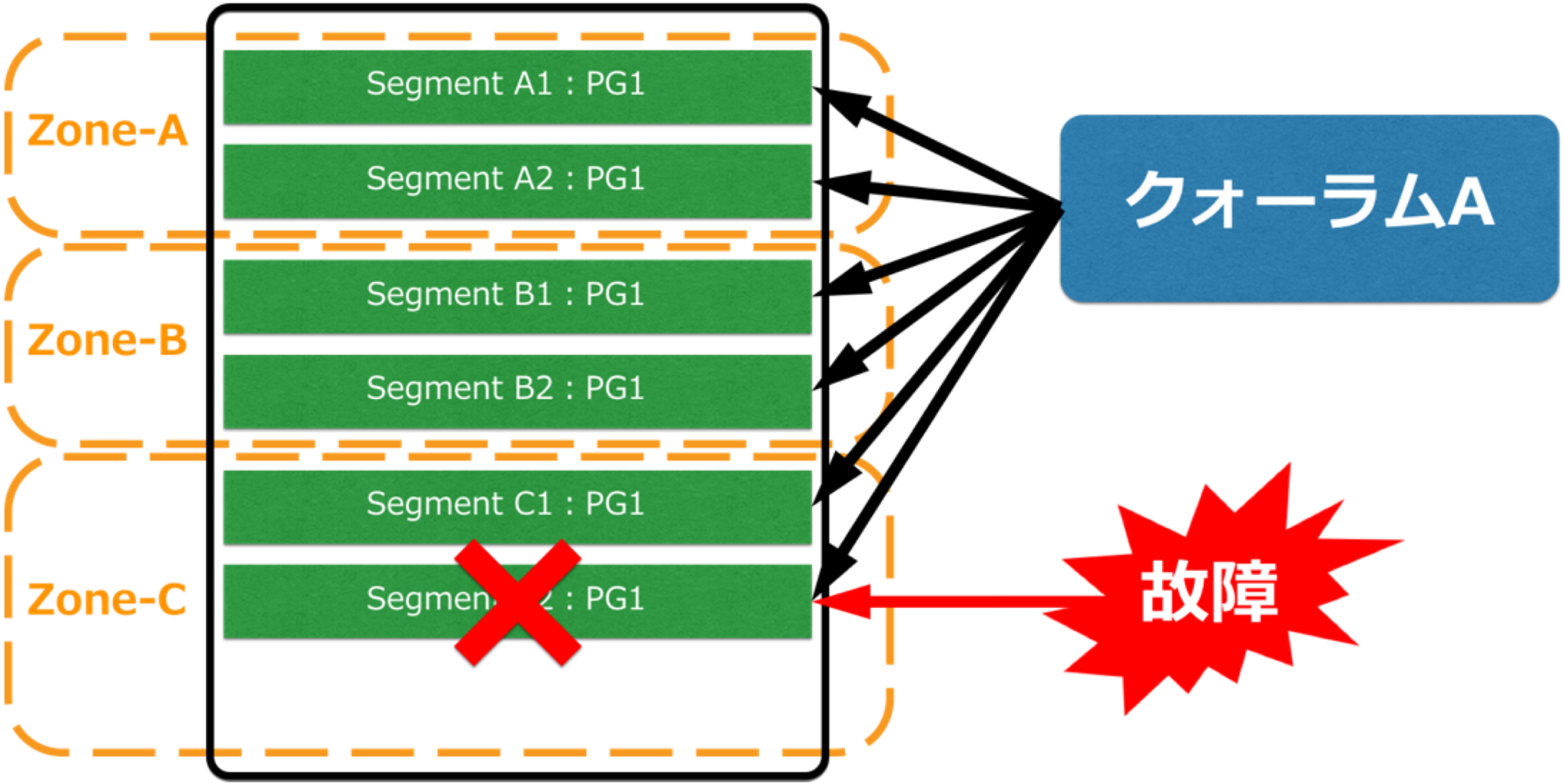
Log Structured Storage （Transaction Recovery） 47



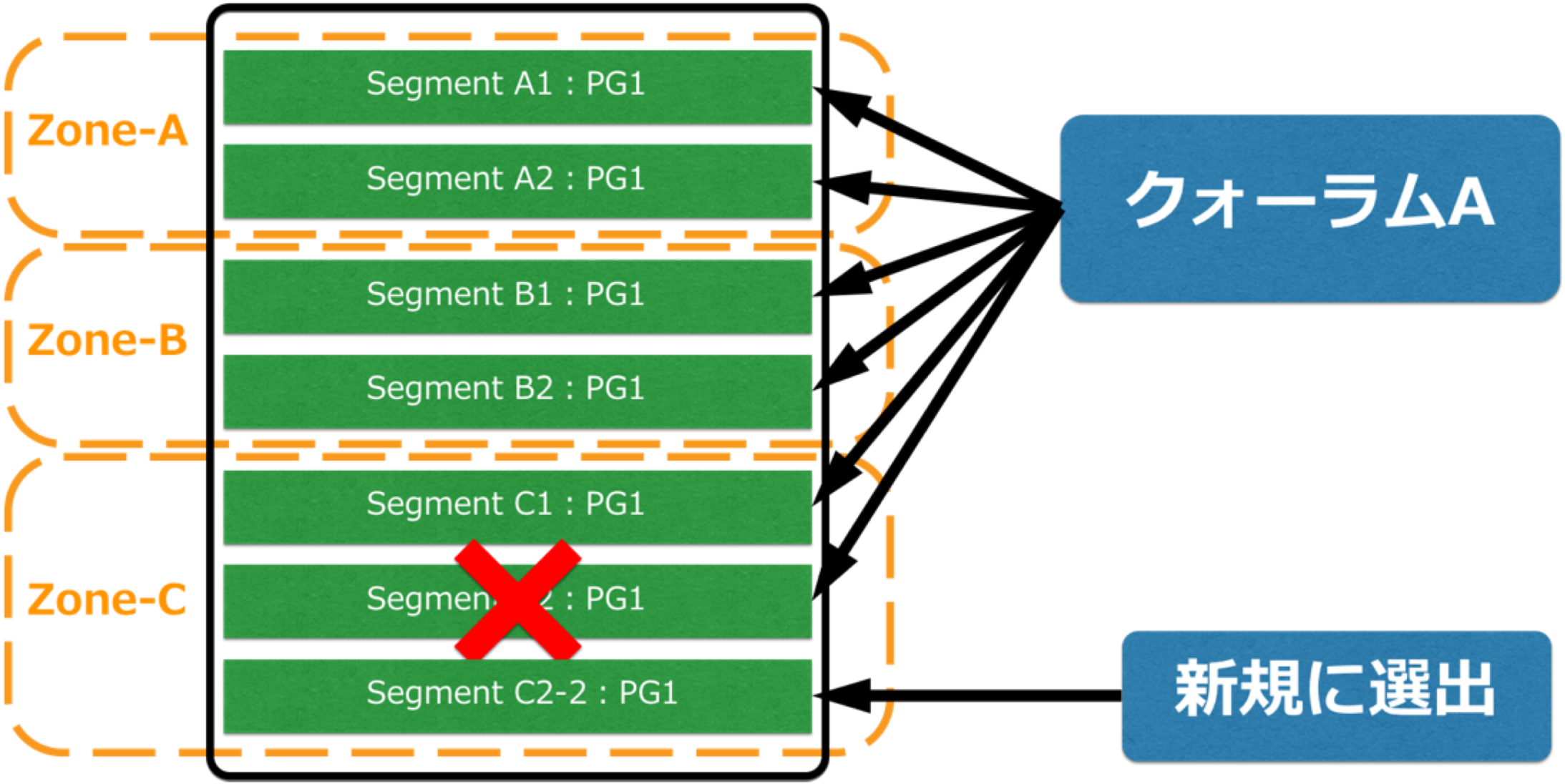
Log Structured Storage （Transaction Recovery） 48



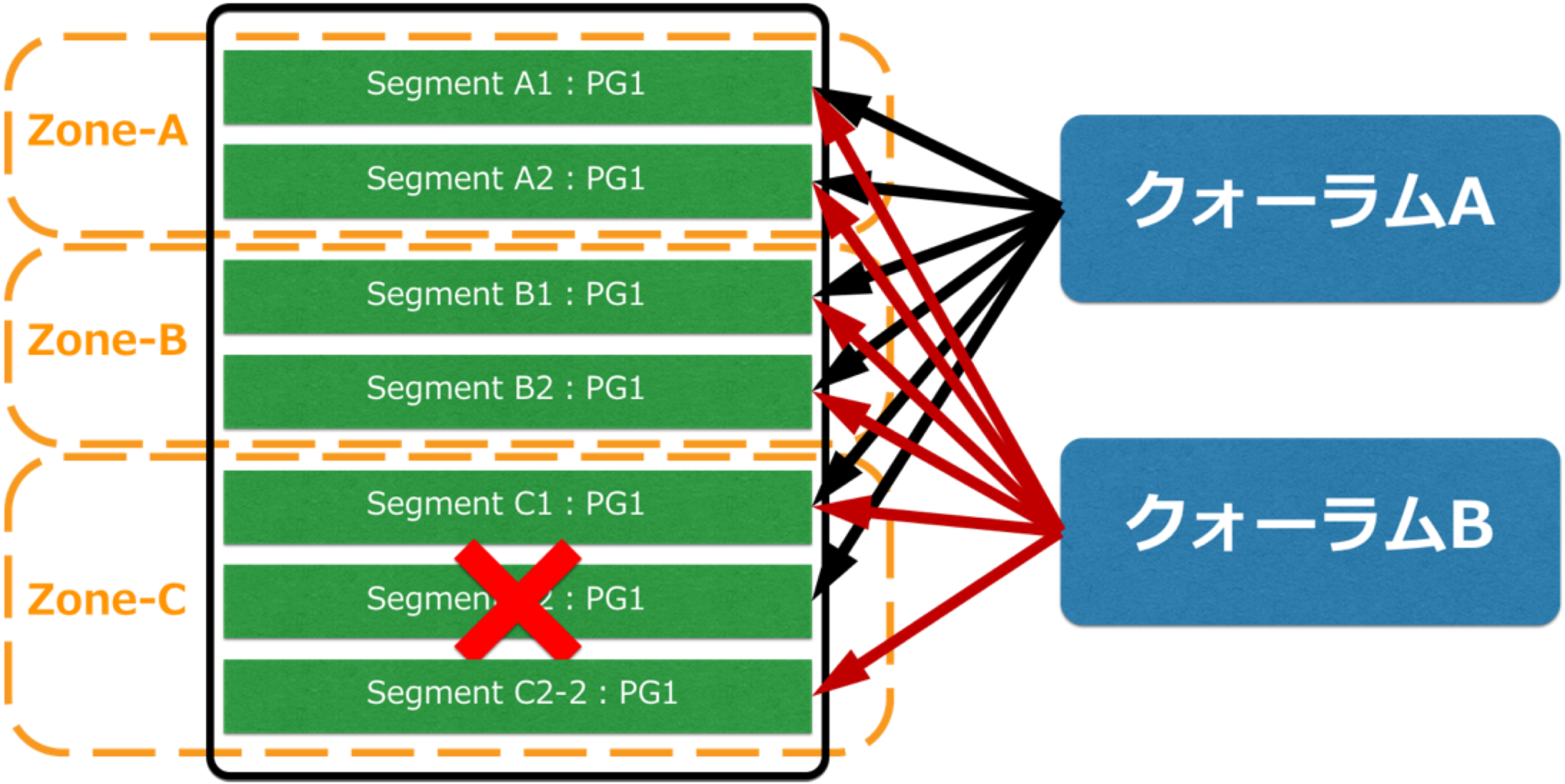
Log Structured Storage （Recovery）



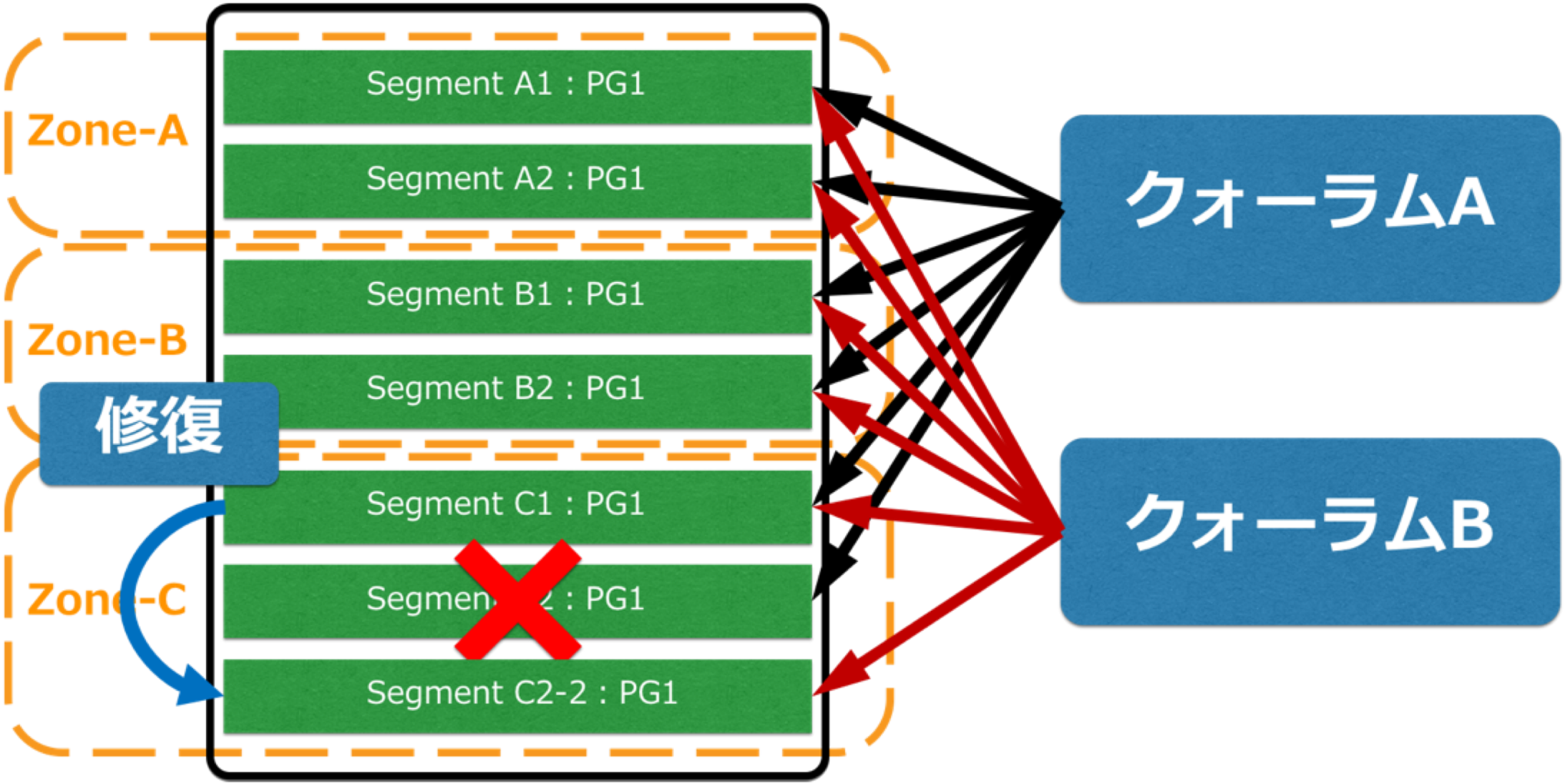
Log Structured Storage （Recovery）



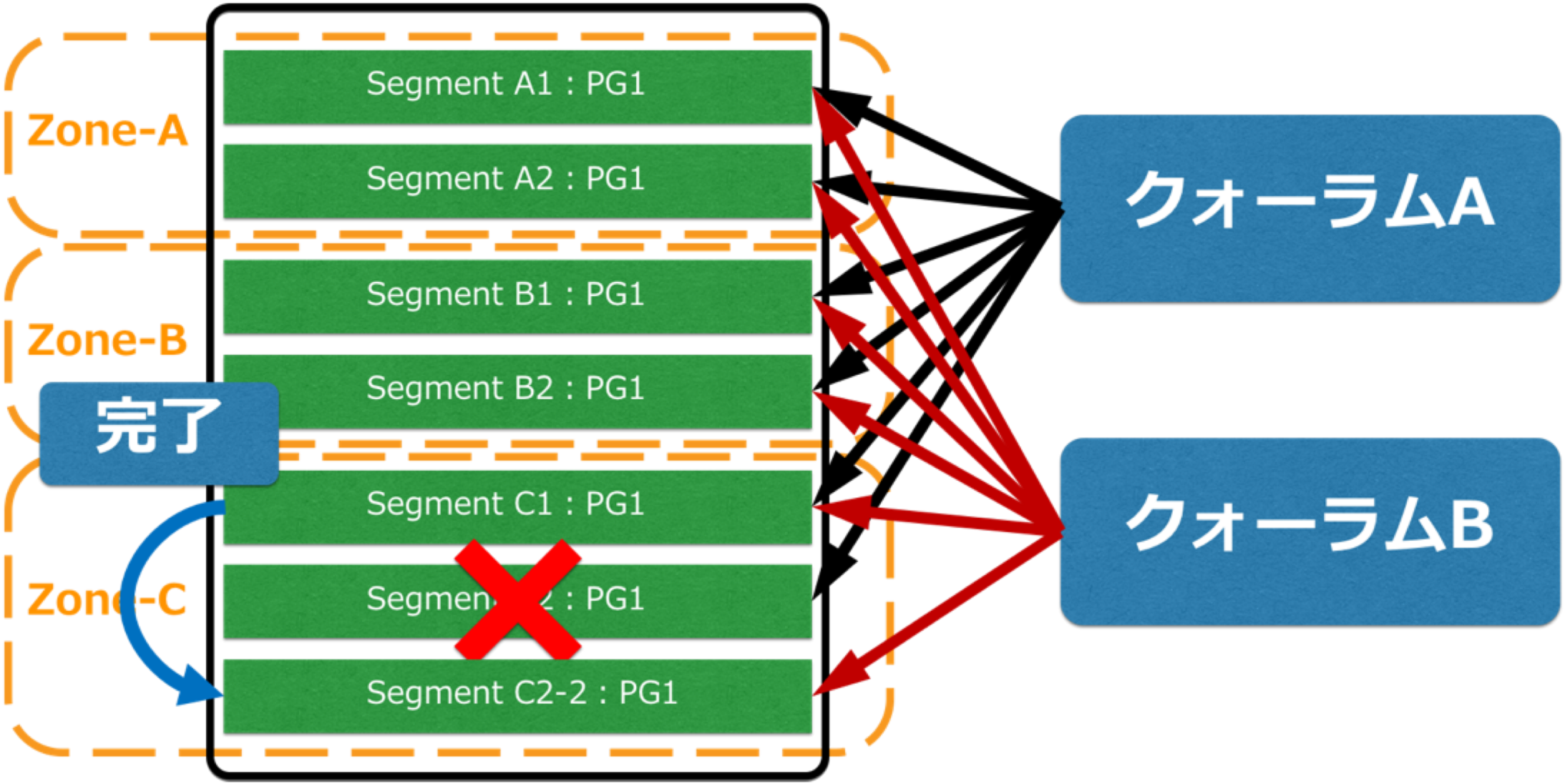
Log Structured Storage （Recovery）



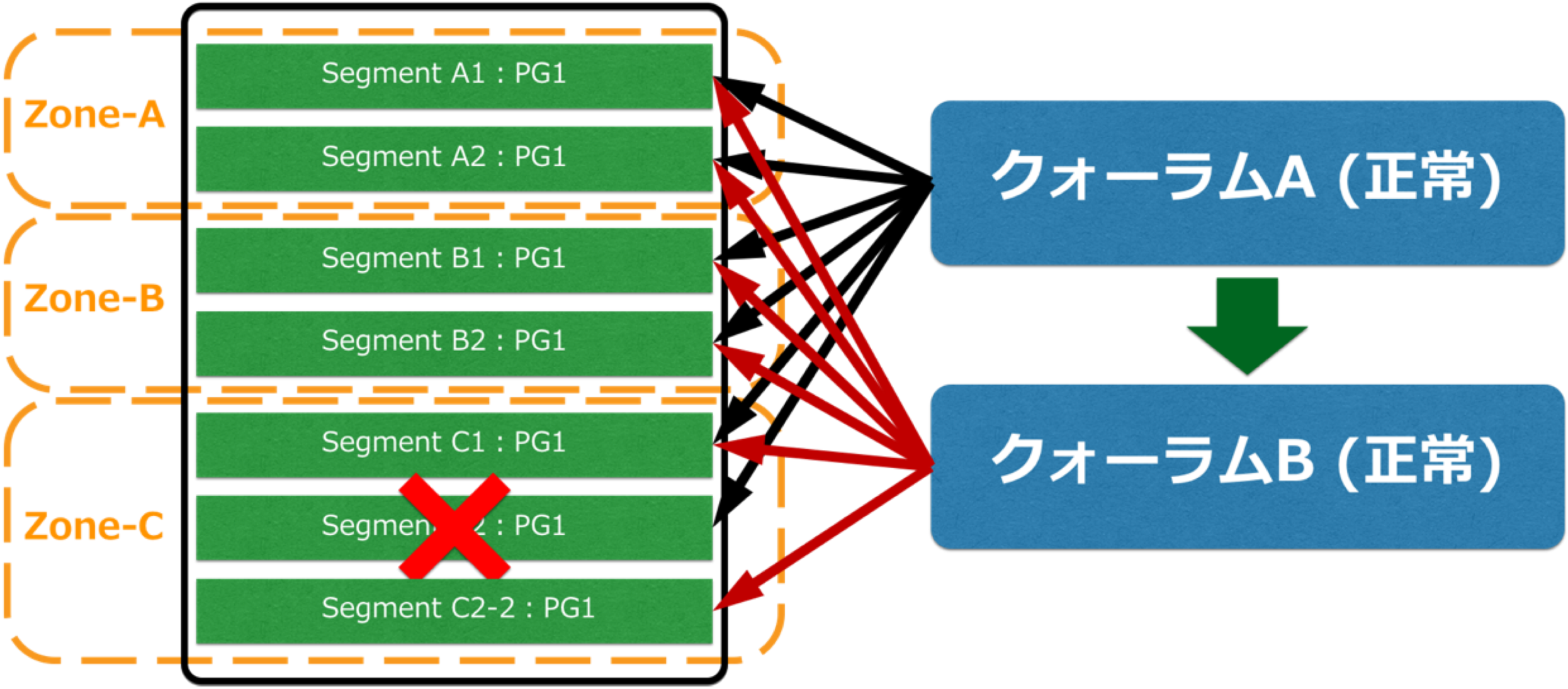
Log Structured Storage （Recovery）



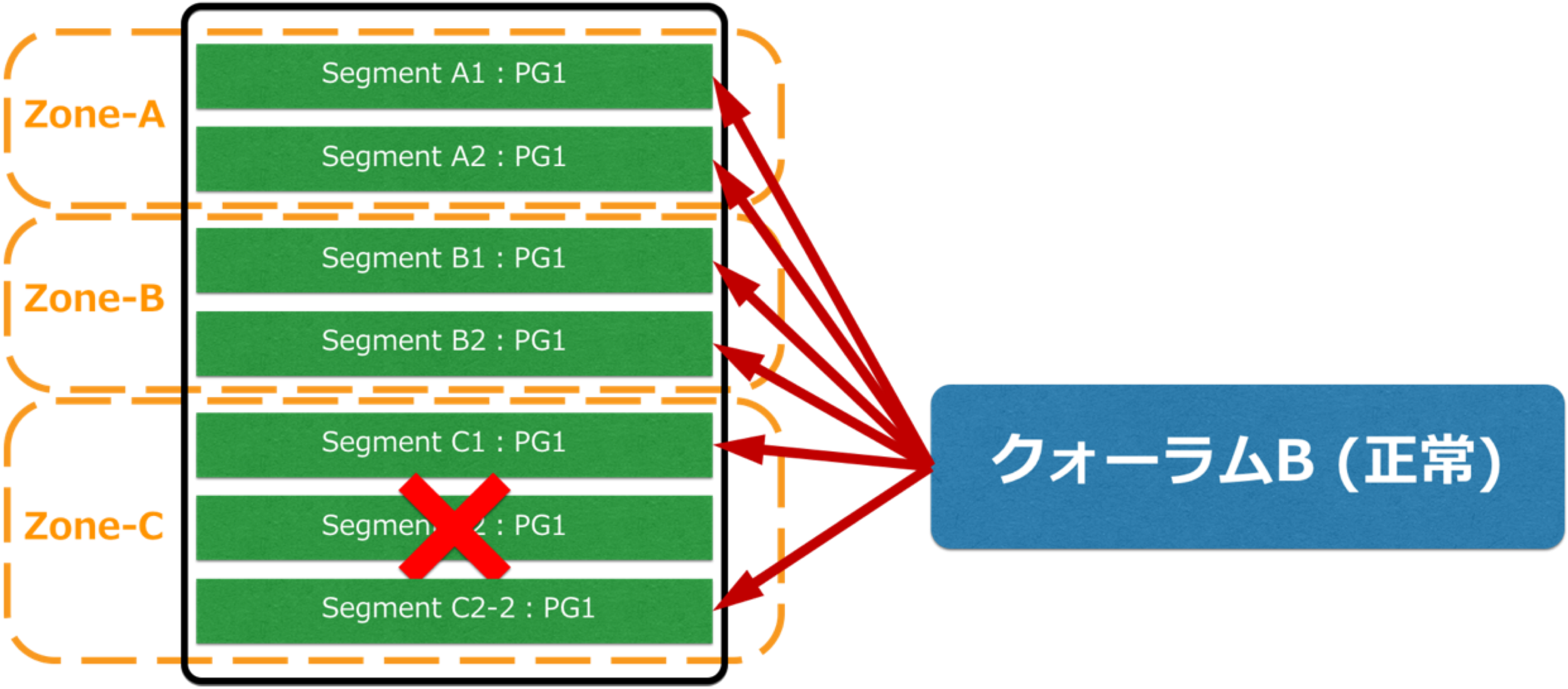
Log Structured Storage （Recovery）



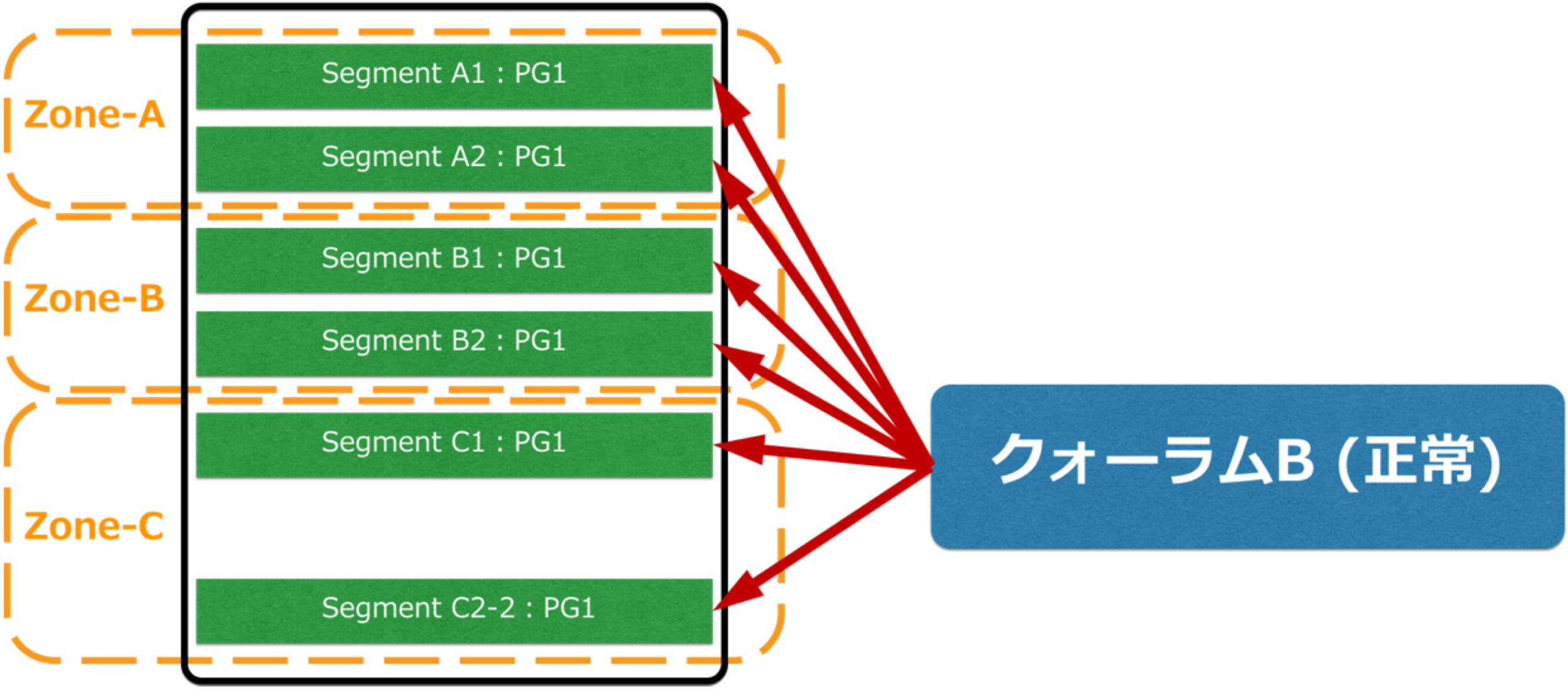
Log Structured Storage （Recovery）



Log Structured Storage （Recovery）



Log Structured Storage （Recovery）

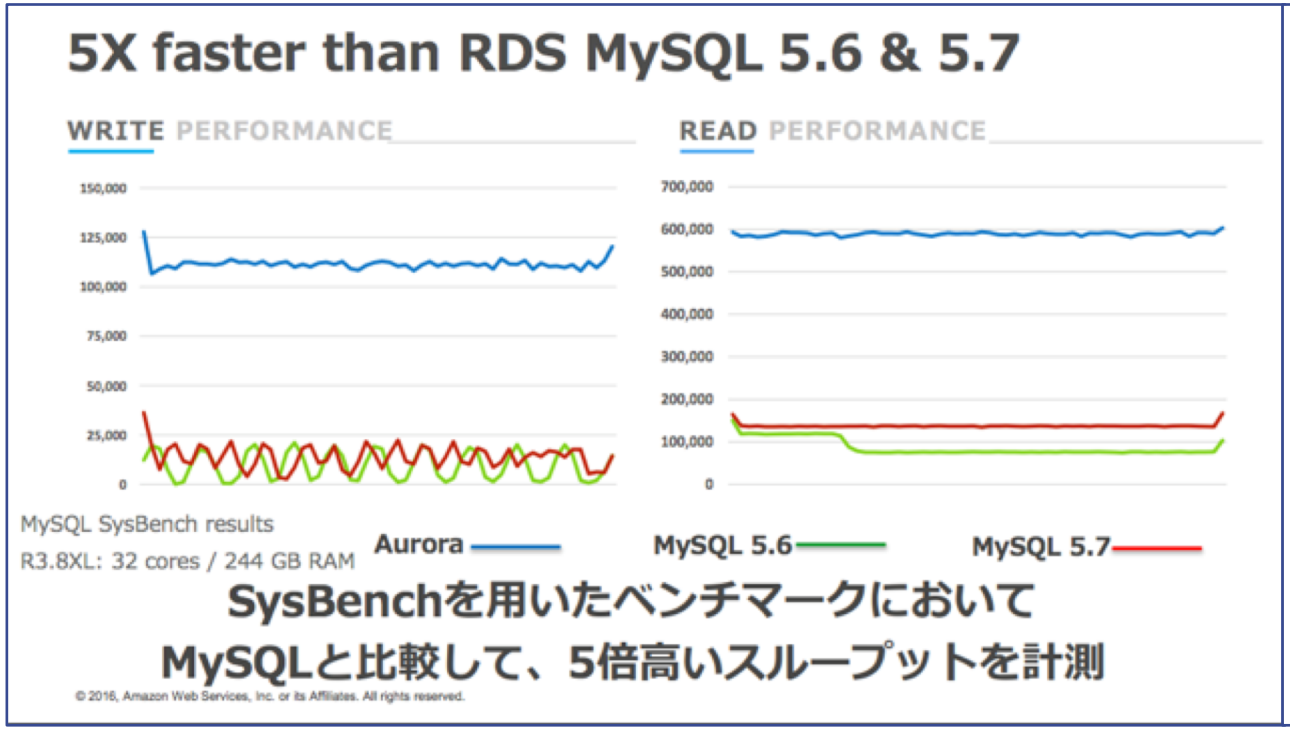


パフォーマンス効率の柱

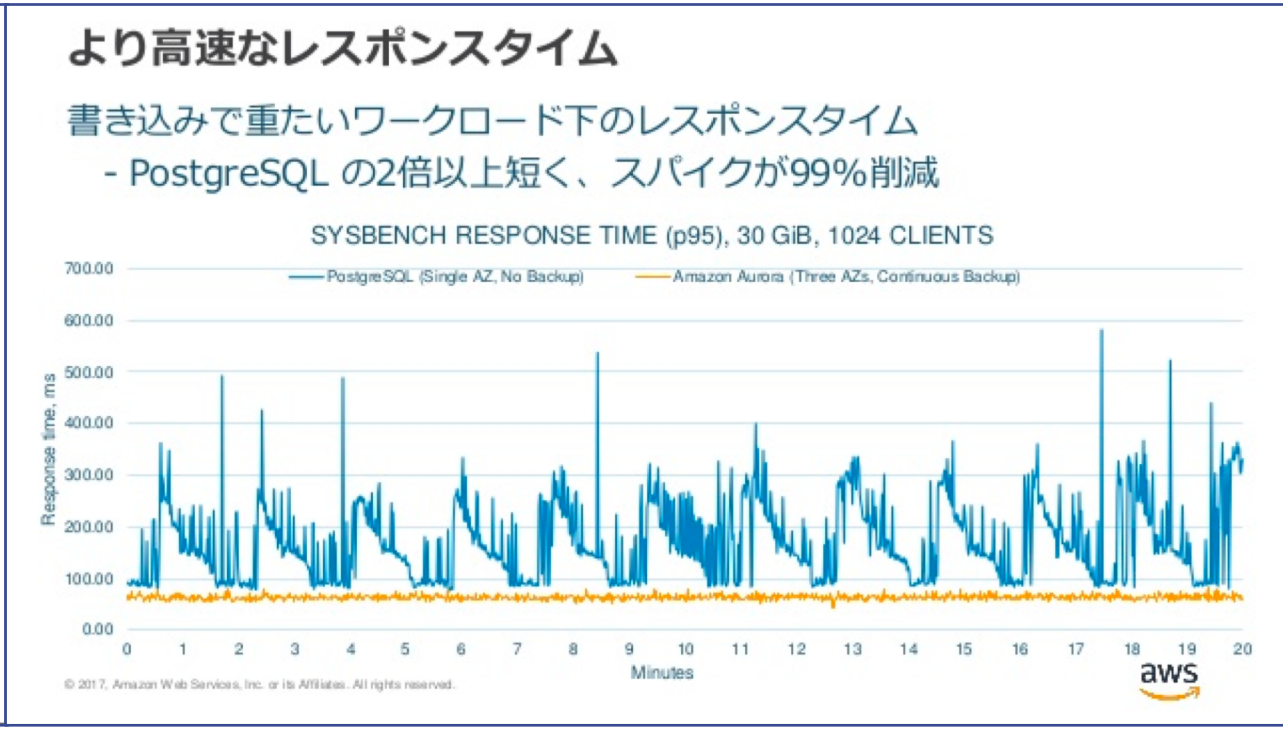
MySQLの5倍、PostgreSQLの3倍のパフォーマンス

- 独自スレッドプール、I/Oの非同期化、グループコミット
- Asynchronous Key PrefetchによるJoinの高速化
- Hash Join、Batch ScanによるJoinの高速化
- Parallel Queryでの分析クエリ高速化
- レプリケーションでSlaveに書き込み負荷がない
- etc

ワークロードによりますが、ベースのプロダクトから大幅なパフォーマンスアップをしています。



http://media.amazonwebservices.com/jp/summit2016/2D-02.pdf



https://image.slidesharecdn.com/20171222awsblackbelt2017aurorapostgresql-171220102238/95/aws-black-belt-online-seminar-2017-amazon-aurora-with-postgresql-compatibility-36-638.jpg

MySQLの5倍、PostgreSQLの3倍のパフォーマンスを発揮するために細かい改善を実施

- 独自実装のスレッドプール
- I/Oの非同期化
- グループコミット
- etc...

AKPによるJoinの高速化

MySQL限定ですが特定のJoinが速くなっています。

MySQL 5.6ではMulti-Range Read(MRR)最適化とBatched Key Access(BKA)というJoinアルゴリズムでJoinの高速化が図られています。

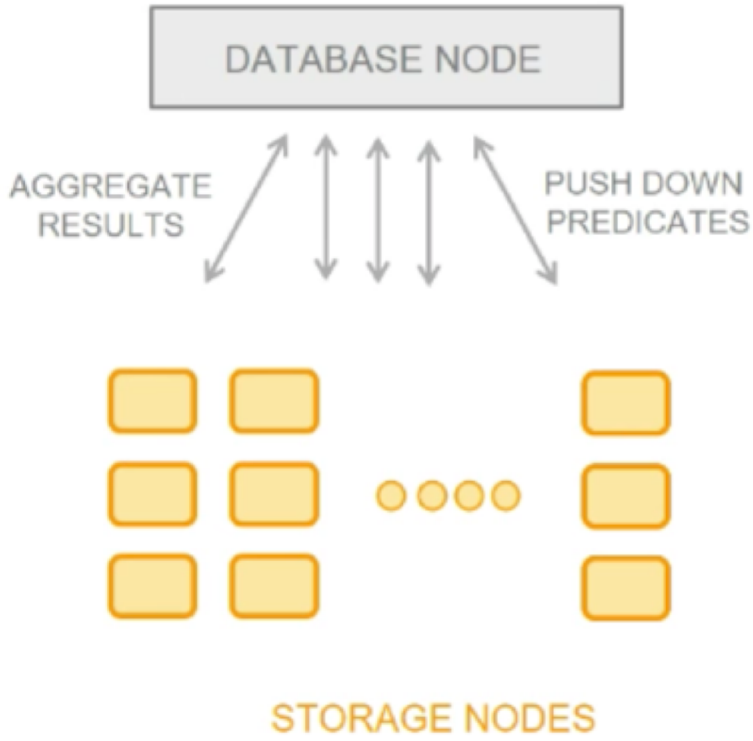
Asynchronous Key Prefetch(AKP)はキーを非同期で先読みすることで更に高速化する機能

MySQLからJoinの種類やスキャン方法を追加

- MySQLはNested Loop Join(NLJ)とその変形のBatched Key Access Join(BKAJ)が実装されています。
  Hash Joinを独自に実装しており比較的大きな表同士などのJoinが高速になっています。
- Batch Scanも実装されてMySQLでは苦手とされているフルスキャンが高速されています。

ストレージノードの潤沢なCPUを使用した並列クエリ

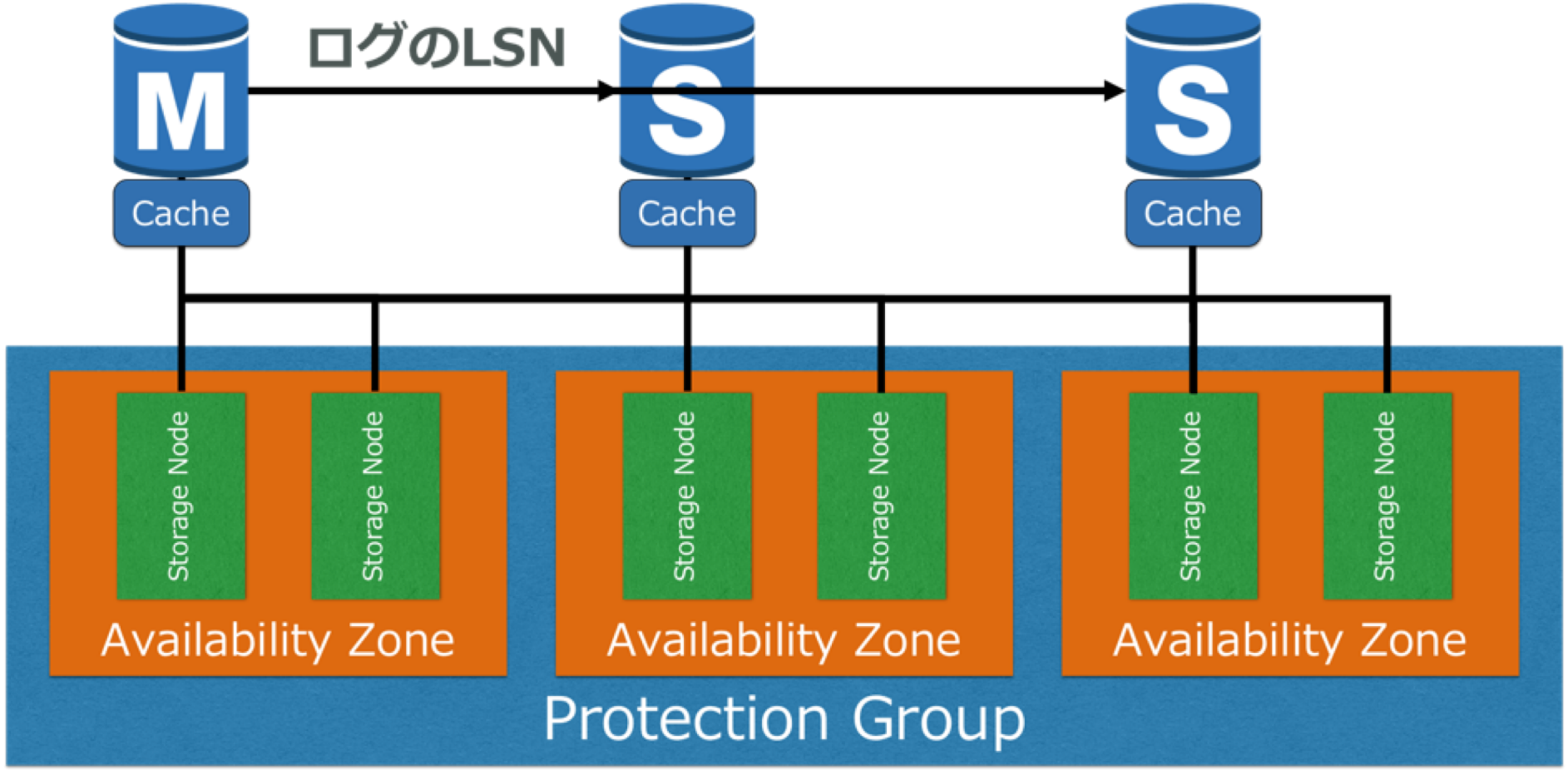
- ヘッドノードがパラレルクエリプランを生成してストレージノードへ送信
- ストレージノードでスキャン対象のページリストを基にスキャン
- ヘッドノードがデータを集約



https://www.slideshare.net/AmazonWebServices/deep-dive-on-the-amazon-aurora-mysqlcompatible-edition-dat301-reinvent-2017

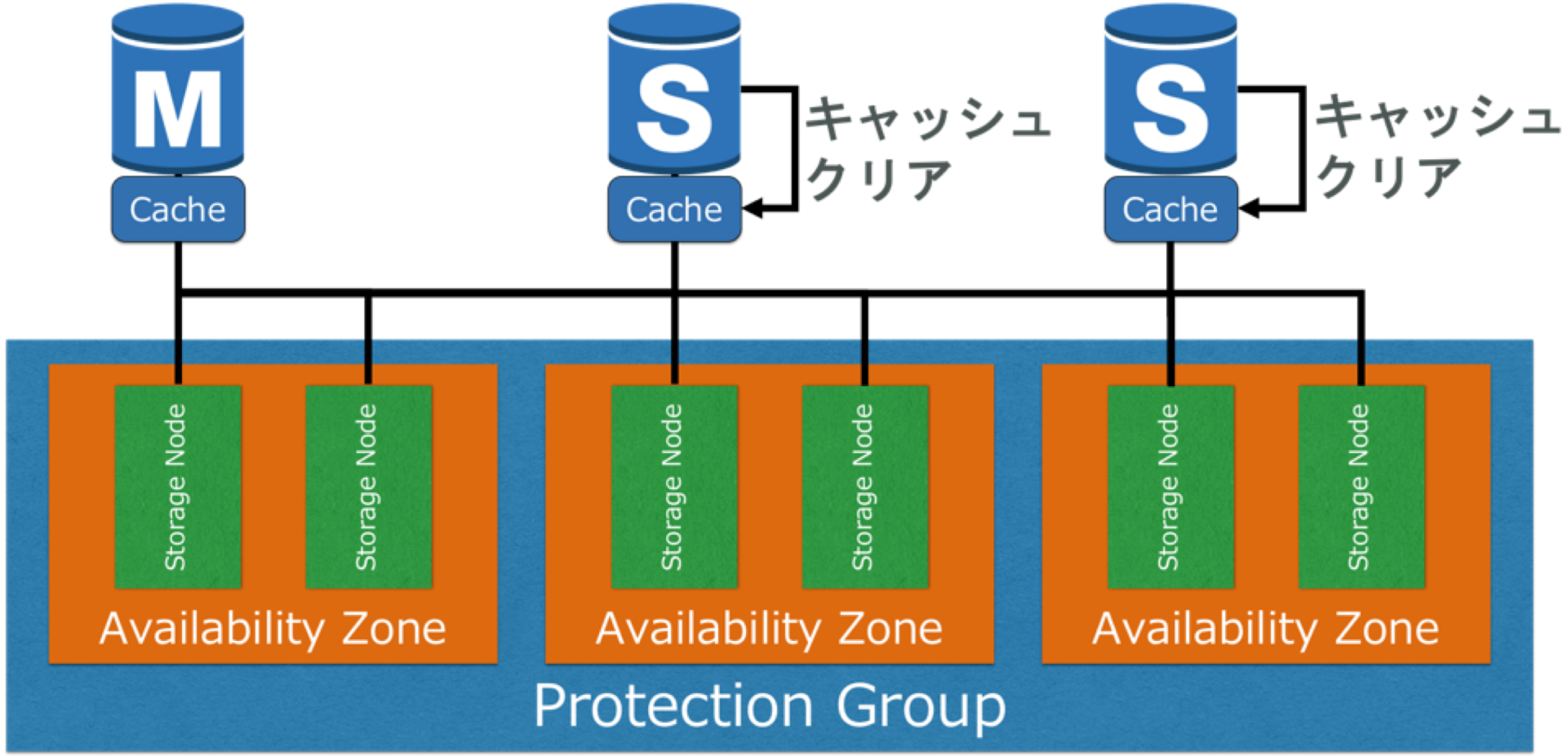
レプリケーションでSlaveの書き込み負荷が無い

書き込みはストレージに行って、Slaveへはログの位置情報のみ伝えて、キャッシュクリアのみ行う



レプリケーションでSlaveの書き込み負荷が無い

書き込みはストレージに行って、Slaveへはログの位置情報のみ伝えて、キャッシュクリアのみ行う



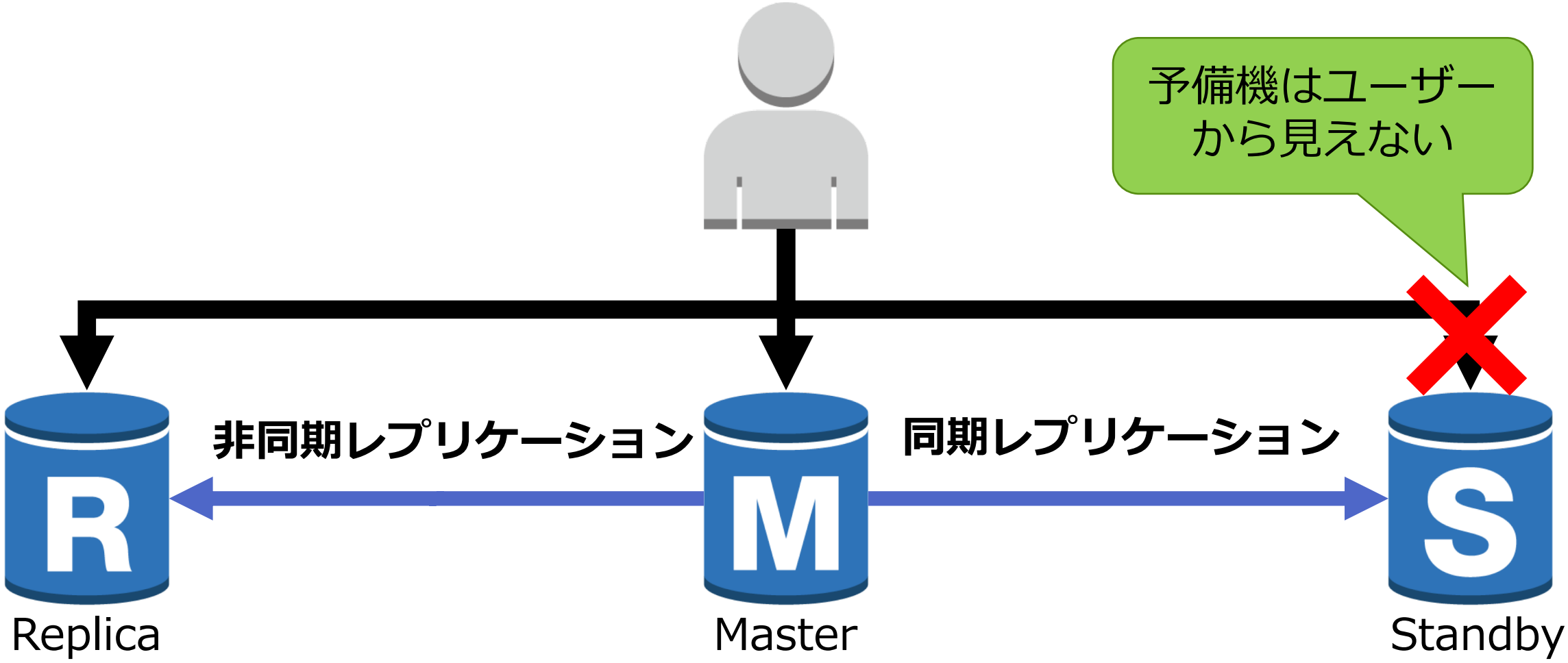
コスト最適化の柱

必要な時に必要な量のリソースを使用してコスト最適化

- Multi-AZの予備機もRead Replicaとして使用可能
- Read ReplicaをAuto Scaling可能
- Aurora Serverlessで待機コストを極小化
- ストレージも完全な6重化せずにコストを抑える
- etc

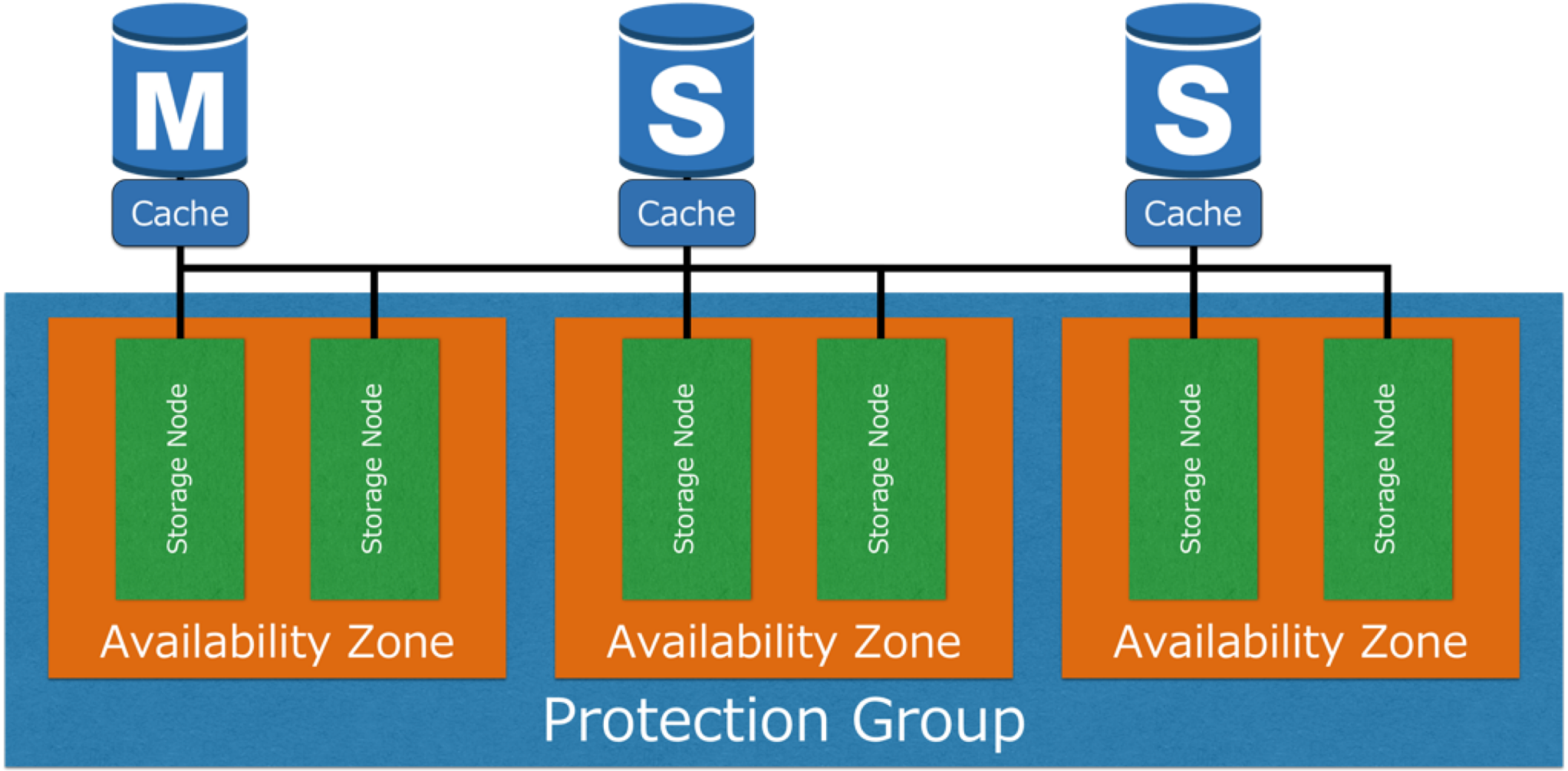
Multi-AZの予備機もRead Replicaとして使用可能

普通のRDSでは予備機は利用不可



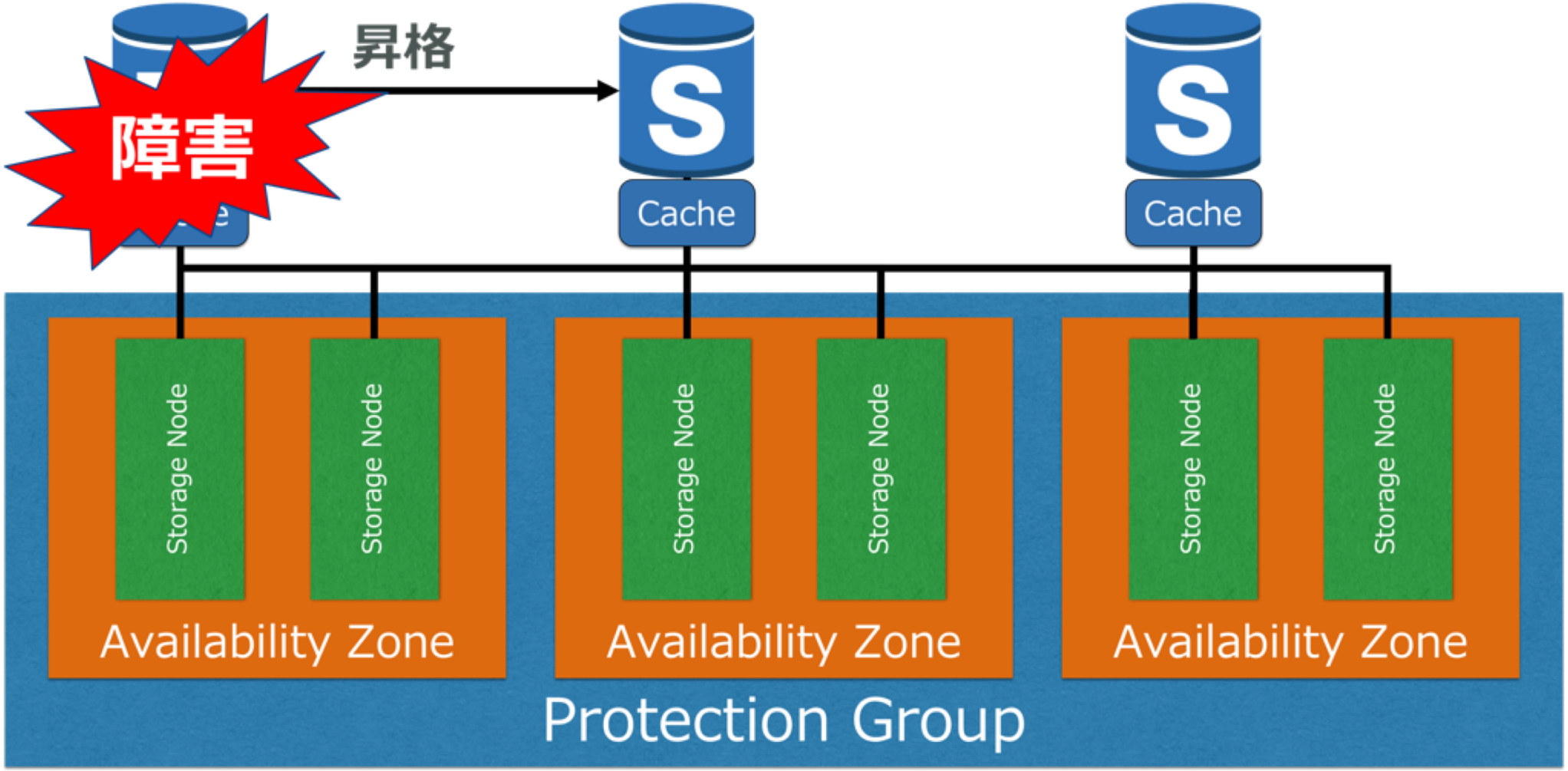
Multi-AZの予備機もRead Replicaとして使用可能

Auroraだとフェイルオーバー先はRead Replica

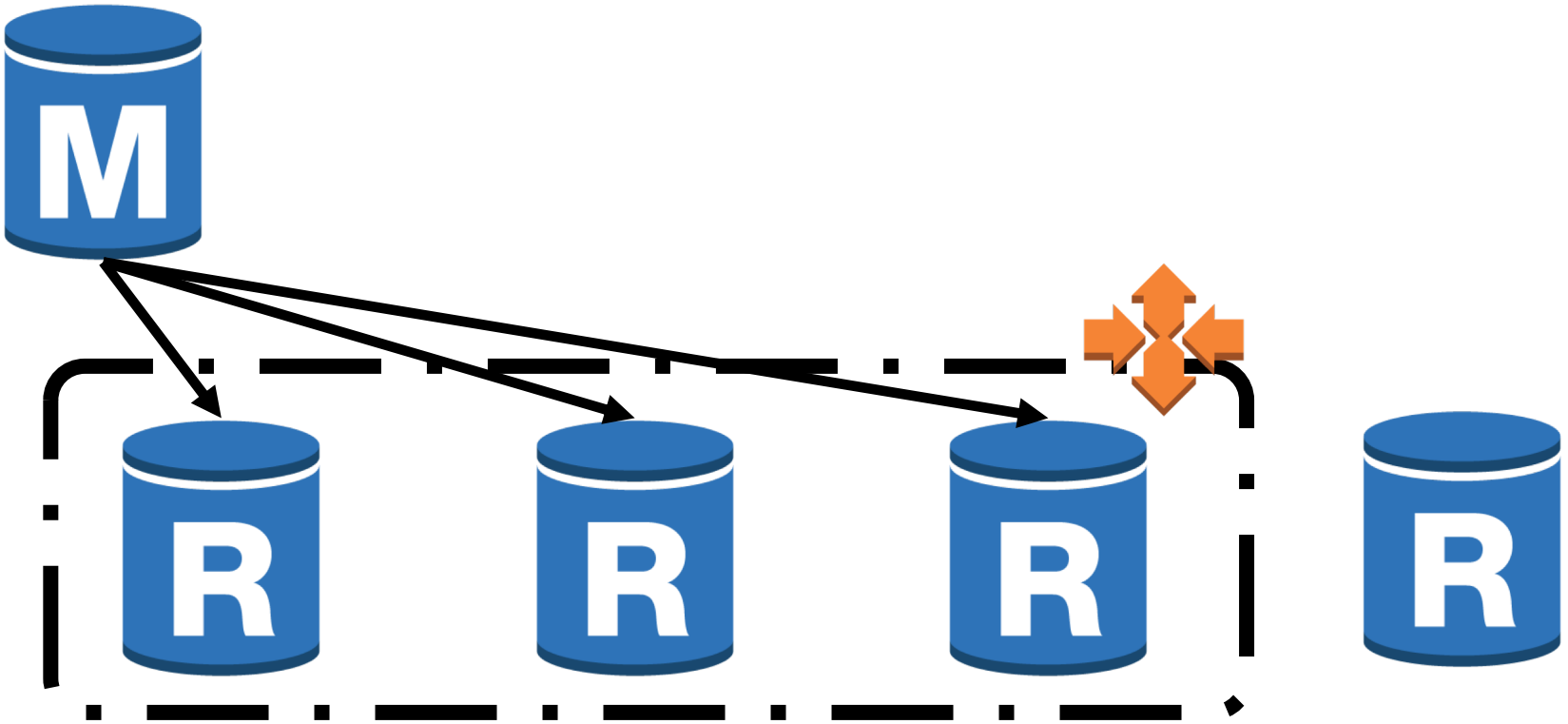


Multi-AZの予備機もRead Replicaとして使用可能

Auroraだとフェイルオーバー先はRead Replica



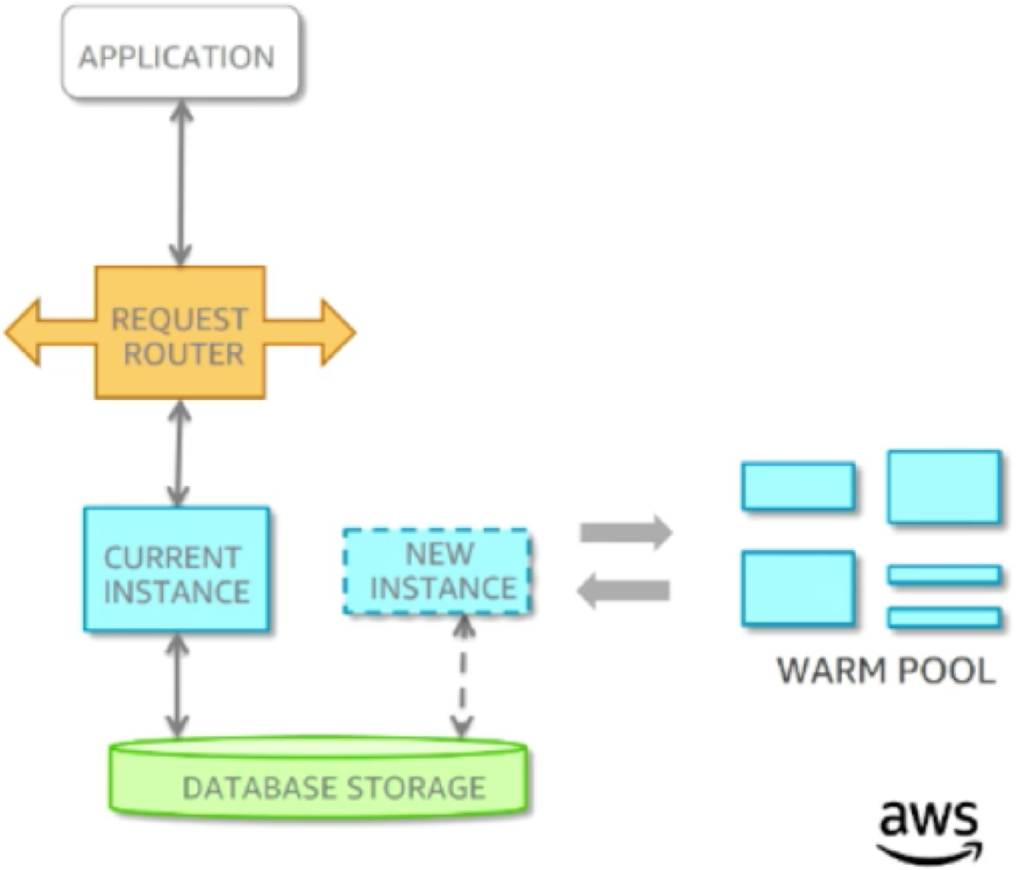
Read Replicaを負荷に応じて台数を増減可能



Aurora Serverlessで待機コストを極小化

re:Invent 2017で発表された新サービス

- 通常はストレージのみの料金
- リクエストがあると25秒程度でプロビジョニングして起動
- インスタンスの待機コストが掛からない



https://www.slideshare.net/AmazonWebServices/deep-dive-on-the-amazon-aurora-mysqlcompatible-edition-dat301-reinvent-2017

セキュリティの柱

データストアなので当然セキュリティにも力を入れている

- データの通信時と保存時の暗号化
- ネットワークの分離
- etc

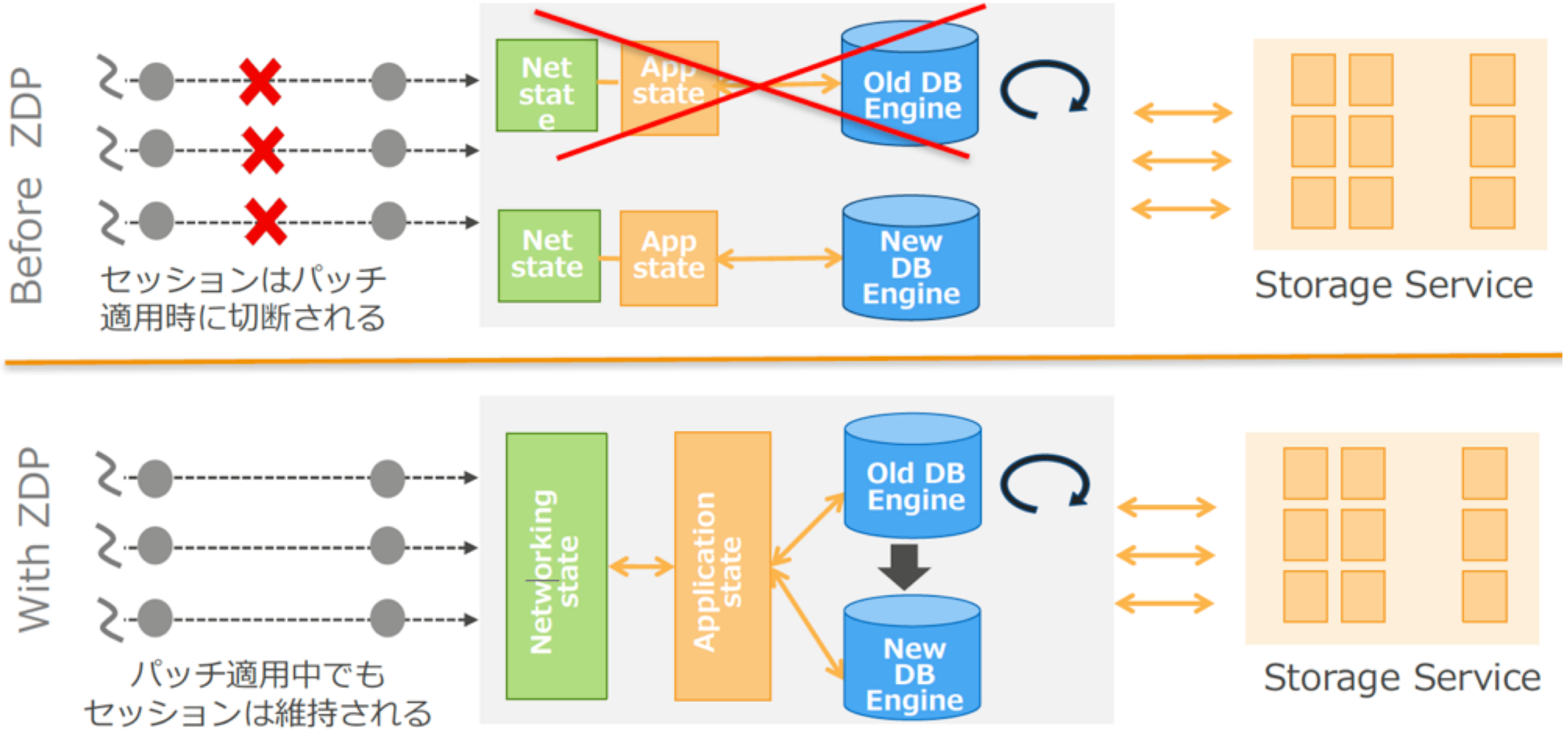
運用上の優秀性の柱

運用上の優秀性

クラウド用に再設計したRDBMSであるため様々な機能が豊富にあります。

- Zero Downtime Patch
- 高速フェイルオーバー
- backtrack
- Database clone
- Performance Insights
- IAM認証でログイン可能
- パフォーマンス低下が少ない監査ログ
- 高速DDL
- Lambdaを実行可能
- etc

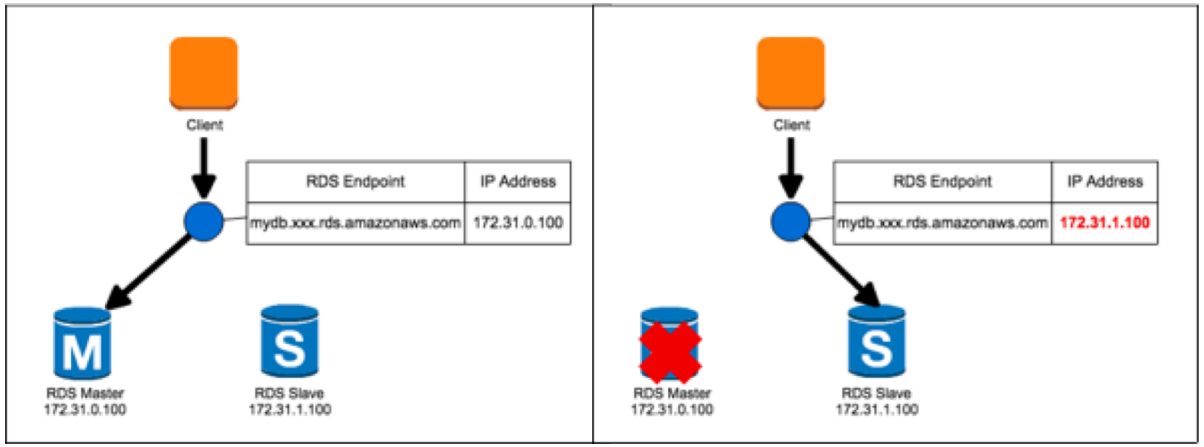
コネクションを切断せずにパッチ適用可能(例外あり)



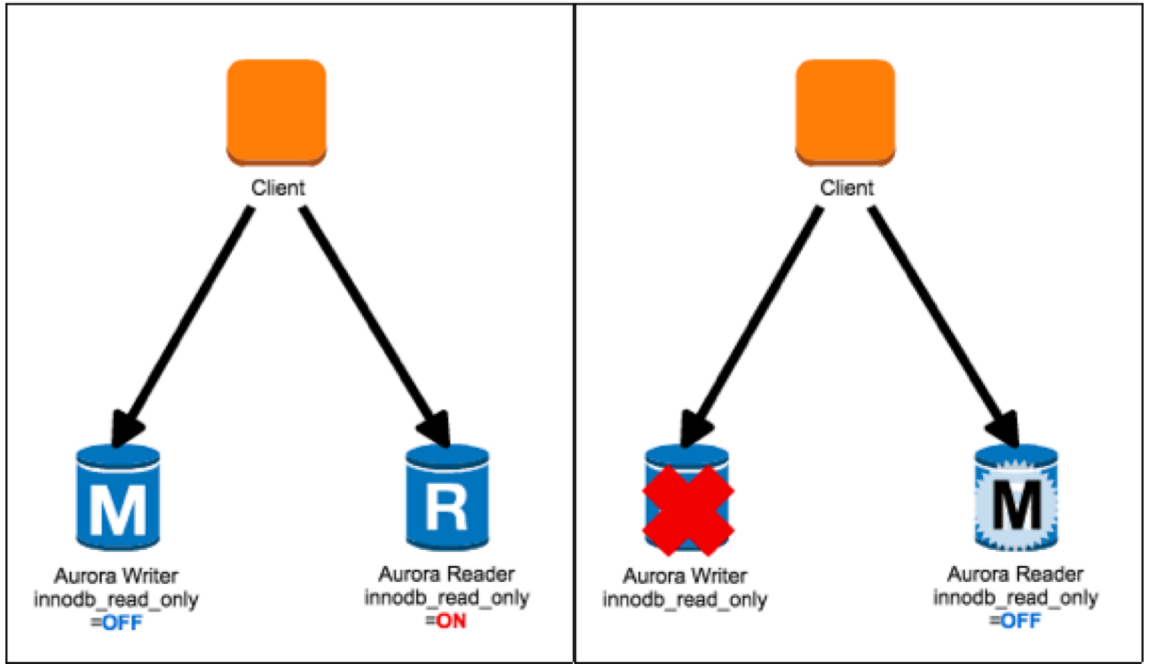
https://d1.awsstatic.com/events/jp/2017/summit/slide/D3T1-6.pdf

Auroraでは高速なフェイルオーバーがサポートされており数秒で切り替えが可能

通常のフェイルオーバーはDNSでMasterとStandbyを切り替える

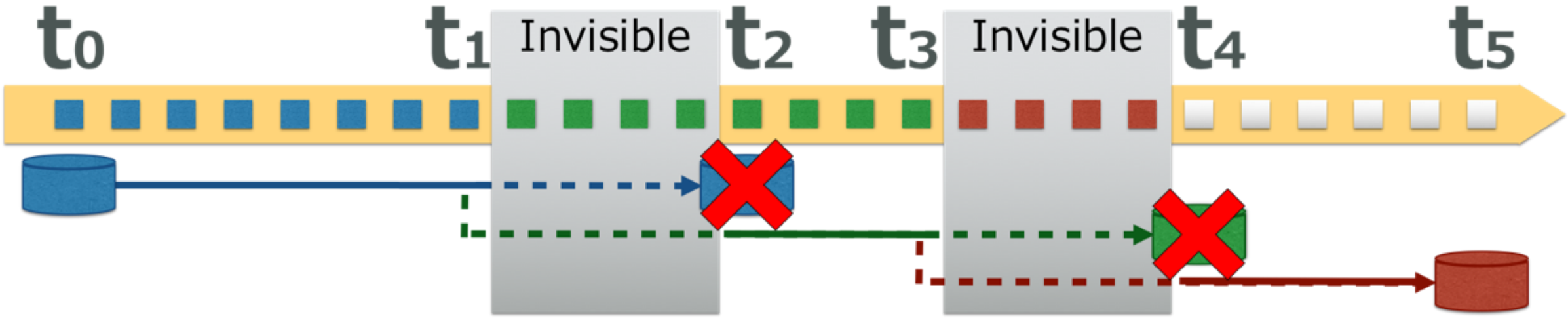


高速フェイルオーバーは接続先DBのフラグをクライアントが判断する



DBデータを任意の時点に高速に巻き戻す事ができます

- 1分程度で指定した任意の時刻へデータを戻せる
- DBのオペレーションミスの復旧(drop tableとか)に活用
- テストの再実行でデータを復元



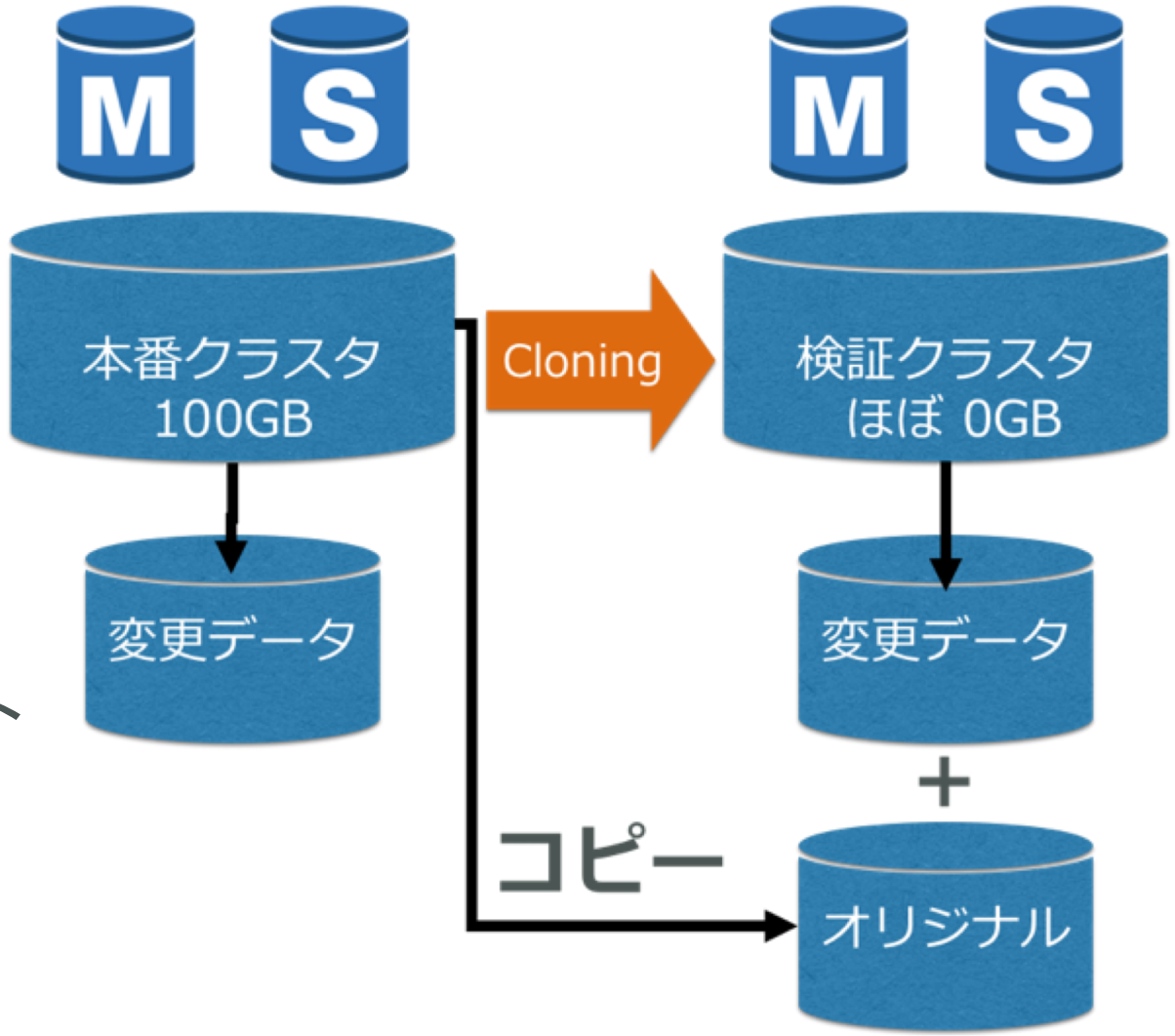
Log-Structured Storageの特徴が出た機能

即座にDBデータを複製

コピーせずに同じデータを
参照するのでほぼ即座に完了

クローン先で変更（追加/更新
/削除）した時にコピー

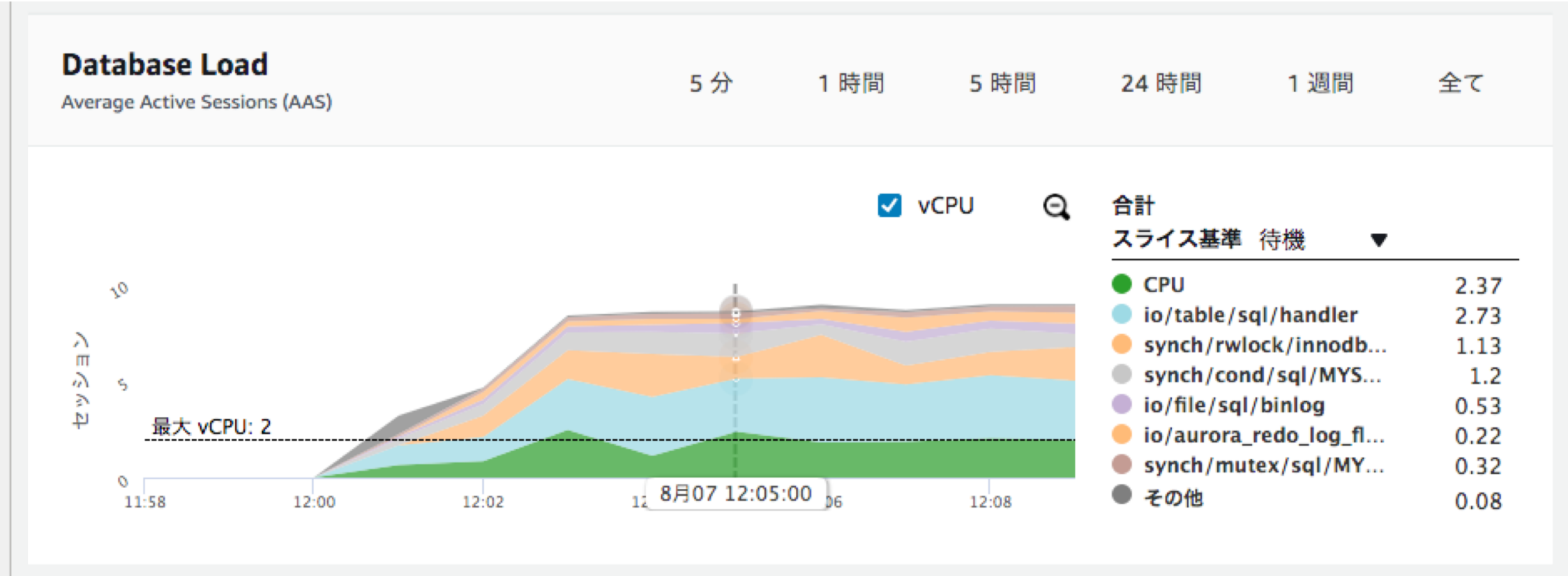
データ変更が無いとノーコスト



パフォーマンスの分析とトラブルシューティングが可能

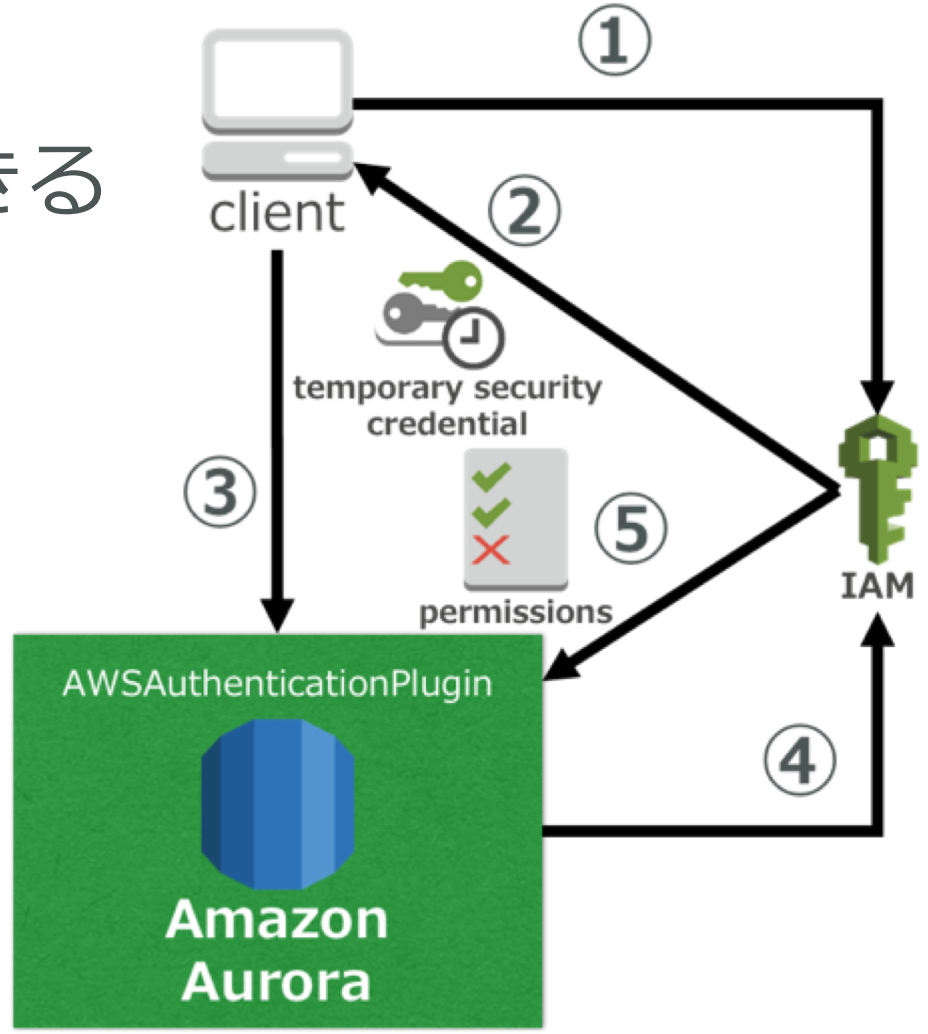
DBの待ち時間がかかっているのか可視化

クエリ単位でも状況確認可能



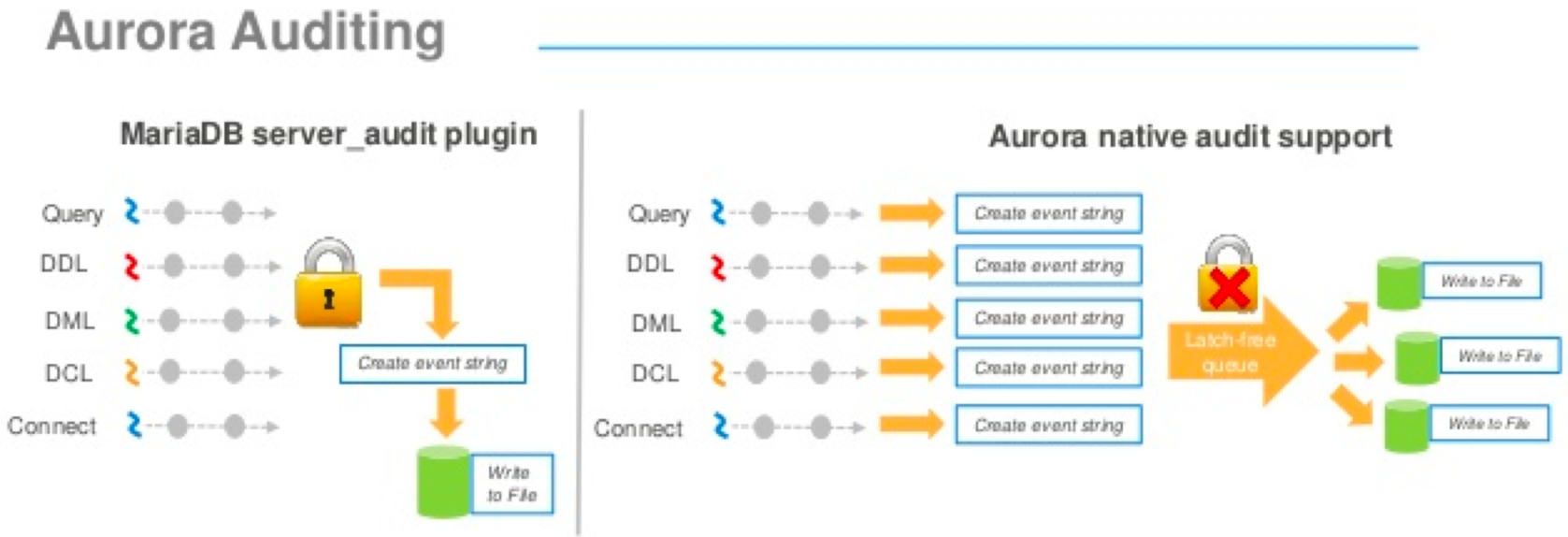
AuroraにIAM権限でログイン

- AWS側で統一的に認証管理ができる
- メンテナンス用途に良さそう
- スループットが低い (20接続/秒)
- アプリケーション接続用としては低負荷の場合のみ



パフォーマンス低下が少ない監査ログ

通常の監査ログプラグインでは65%ダウンですが、Auroraでは15%ダウンのスループットで済みます。



Aurora can sustain over 500K events/sec

	MySQL 5.7	Aurora	
Audit Off	95K	615K	6.47x
Audit On	33K	525K	15.9x

Sysbench Select-only Workload on 8xlarge Instance

https://image.slidesharecdn.com/dat303-161214152920/95/aws-reinvent-2016-deep-dive-on-amazon-aurora-dat303-51-638.jpg

特定パターンでのカラム追加で圧倒的に高速に実行可能

テーブル末尾にnullableなカラムを追加する場合に
1万倍以上！高速になります

On r3.large

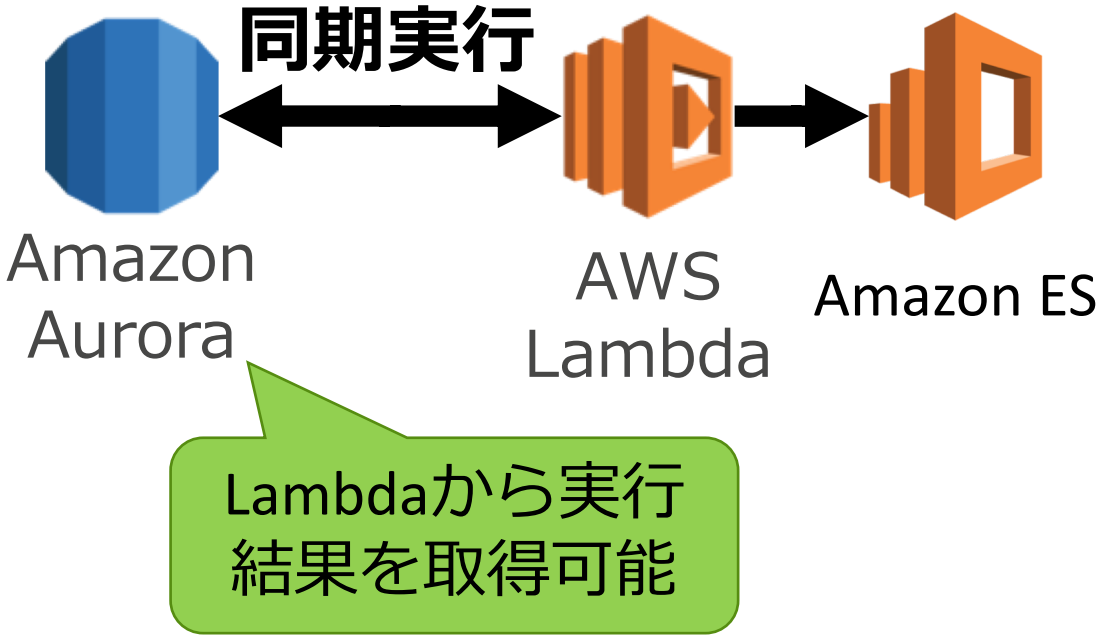
	Aurora	MySQL 5.6	MySQL 5.7
10GB table	0.27 sec	3,960 sec	1,600 sec
50GB table	0.25 sec	23,400 sec	5,040 sec
100GB table	0.26 sec	53,460 sec	9,720 sec

On r3.8xlarge

	Aurora	MySQL 5.6	MySQL 5.7
10GB table	0.06 sec	900 sec	1,080 sec
50GB table	0.08 sec	4,680 sec	5,040 sec
100GB table	0.15 sec	14,400 sec	9,720 sec

AuroraからLambdaを呼べる

- 非同期では戻り値無し
- 同期実行では戻り値を取れるので外部サービスの情報を取得できます
- トリガーとも組み合わせ可能



まとめ

まとめ

- Auroraが凄いところは基本的にストレージに起因したものです。
- 話しきれないくらいにAuroraには凄い特徴が多種多様にあります。各種機能を理解すると痒いところに手が届いたり、RDSの利用を諦めていたケースでもマネージドサービスを利用できると思います。
- 個人的な意見ですが、Auroraはパフォーマンスの良さ以上に運用上楽になる機能が豊富にあることがメリットだと感じています。


classmethod