Amazon ElastiCache

AWS Black Belt Tech Webinar 2015
アマゾンデータサービスジャパン株式会社
ソリューションアーキテクト
成田　俊

2015/08/19

Agenda

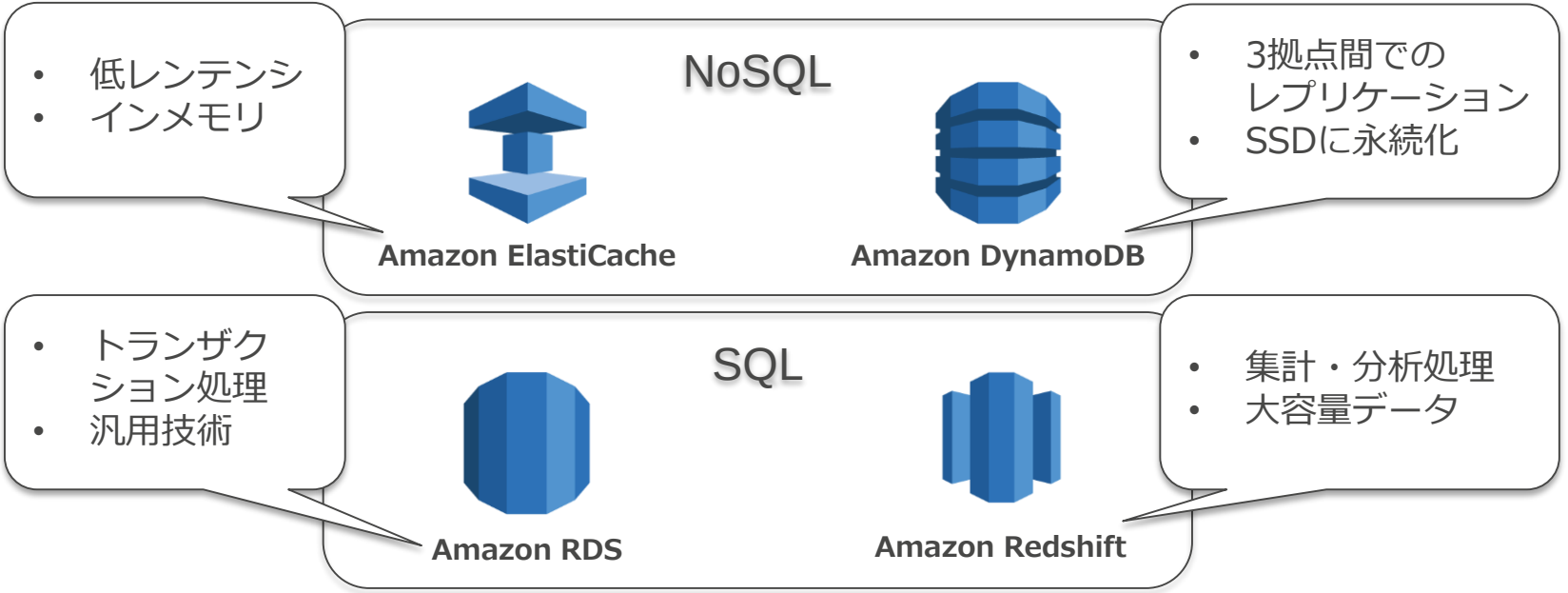
Agenda

- **1．Amazon ElastiCache とは？**
  - ElastiCache 概要
  - メモリキャッシング 概要
  - Memcached、Redis 概要
- **2．前回(2015年1月)のWebinar以降のアップデート**
  - 2015年 1月 – ElastiCache GovCloud(US)、Frankfult(EU)リージョン対応
  - 2015年 2月 – Redis Cost allocation tags対応
  - 2015年 3月 – Redis 2.8.19対応
  - 2015年 7月 – Redis 2.8.21対応、Memcached Auto Discovery PHP 5.6対応
- **3．ユースケース、システムアーキテクチャ、注意点**
- **4．価格、まとめ**

1．Amazon ElastiCache とは？

Amazon ElastiCacheの位置づけ

- データ・ストアの特性に応じた使い分け



Amazon ElastiCacheとは

メモリ内キャッシュをマネージドで提供するサービス
→キャッシュに読み込まれたオブジェクトを保存し、性能負荷を軽減する

- 構築
  - キャッシュクラスタを数クリックで起動
  - EC2、RDSと同様、初期費用無し、時間単位の従量課金
- 移行
  - 2種類のエンジン**(memcached, redis)**をサポート
  - 既存アプリケーションの変更不要
- 運用
  - 可用性を向上させる機能
  - モニタリング、自動障害検出、復旧、拡張、パッチ管理機能を提供
- セキュリティ
  - セキュリティグループ、VPC対応

Memcached/Redis on EC2 との差異

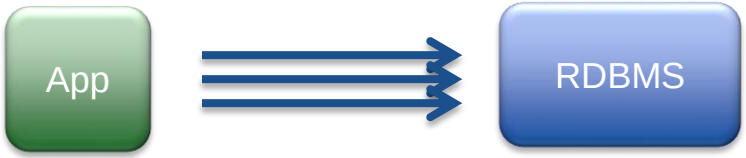
- 導入
  - ノード用インスタンスのOSセットアップ 不要
  - memecached/ Redisのインストール・セットアップ 不要
  - 複数キャッシュノードへのコンフィグファイル配布・同期不要
  - Management Console/CLI/APIから数分で起動・ノード追加
- 運用
  - 監視・高可用性の作り込み不要
  - ノードリカバリ、パッチ適用 自動
  - Firewallの用意、詳細設定不要
- AWS独自の用語・概念
  - セキュリティ関連
  - クラスタ関連

メモリキャッシングとは

- 目的
  - アプリを高速化する手法の一つ
  - 消えても良いデータを格納してDBアクセス・負荷を低減
  - メモリにキャッシュしたデータを再利用し 低遅延化・負荷低減
- 用途
  - クエリ結果を再利用 (DBサーバの負荷低減、高速化)
  - 揮発性の高いデータを格納 (セッション情報管理)
  - 複雑な計算結果・二次データを再利用　（APPサーバの負荷軽減)

Web+DBアプリとメモリキャッシュ

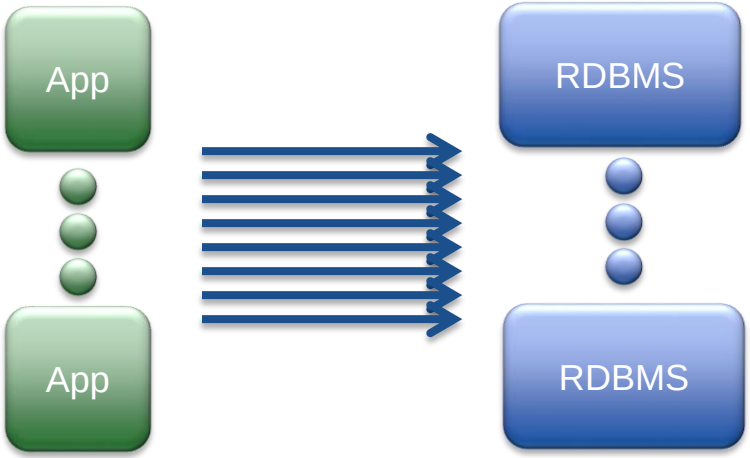
- 典型的な構成



1. クライアントからのリクエスト
2. Appサーバが、DBサーバに問い合わせ
3. DBサーバが結果を戻す
4. Appサーバがレスポンスをクライアントに返す

Web+DBアプリとメモリキャッシュ

- トラフィックが増えると

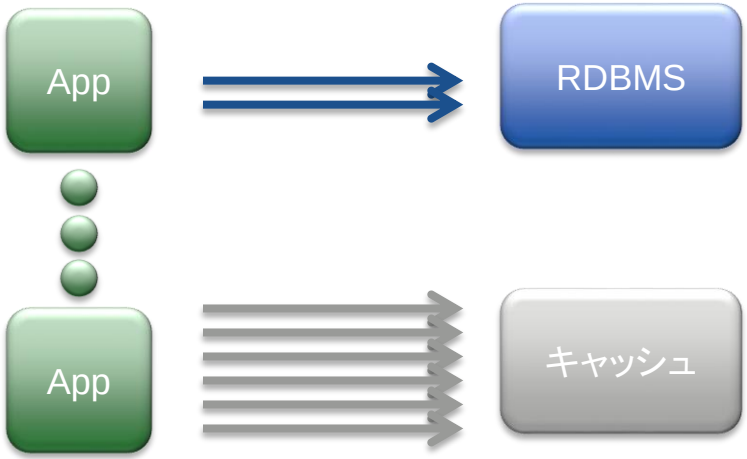


5. Appサーバ,DBサーバをスケール
6. 効果・効率・コスト的な面、DBをスケールさせる難易度は？

⇒　RDBをスケール“アウト”させるのは難しい。

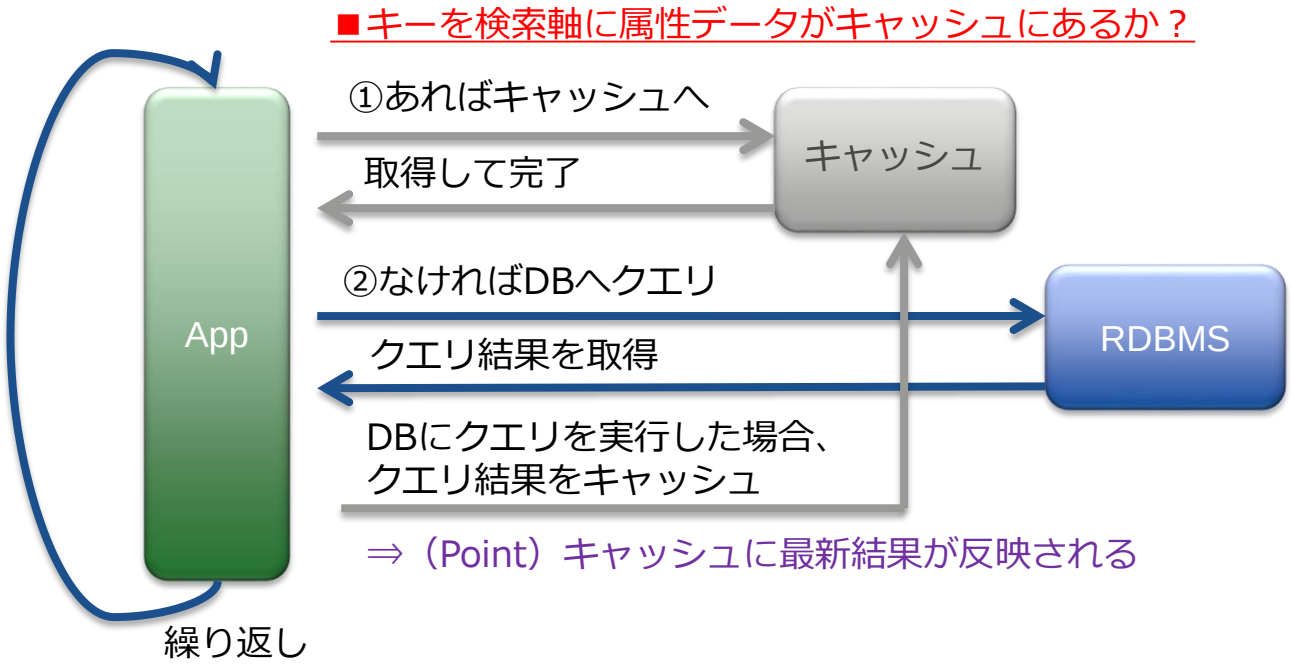
Web+DBアプリとメモリキャッシュ

- DB負荷を軽減するためにキャッシュにデータを載せる
  - アプリケーション側で、DBとキャッシュを使い分ける



Web+DBアプリとメモリキャッシュ

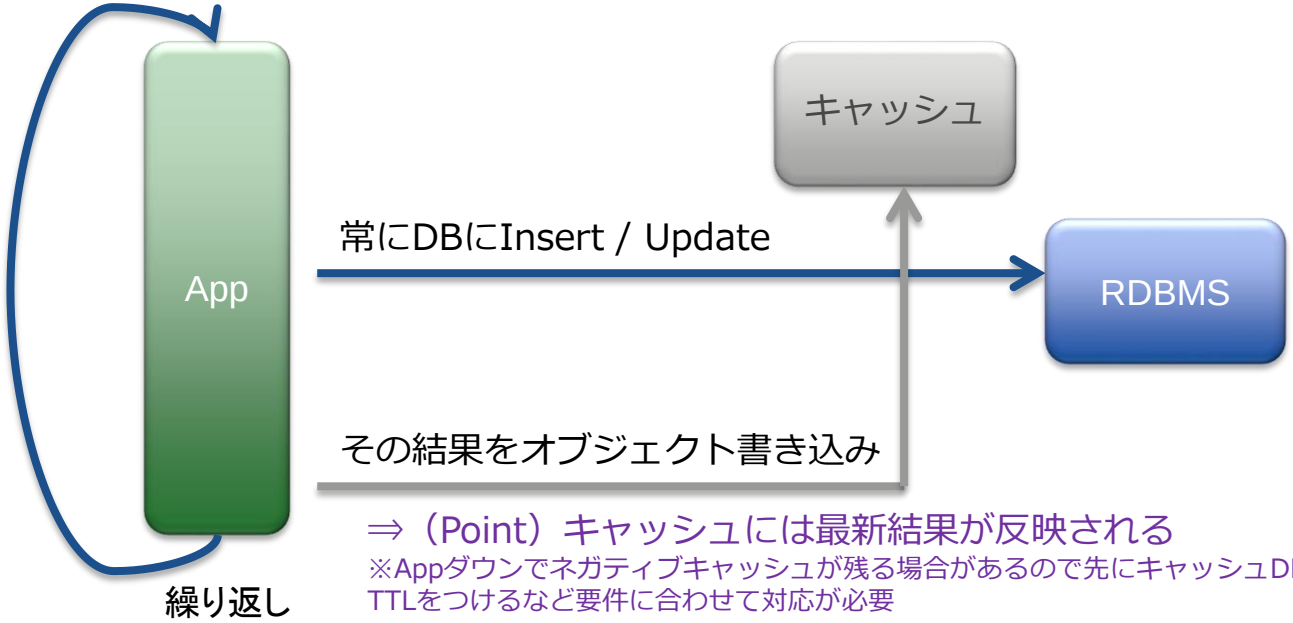
- データ参照時の操作



Web+DBアプリとメモリキャッシュ

- 更新時の操作

■キーを検索軸に属性データをキャッシュに配置する

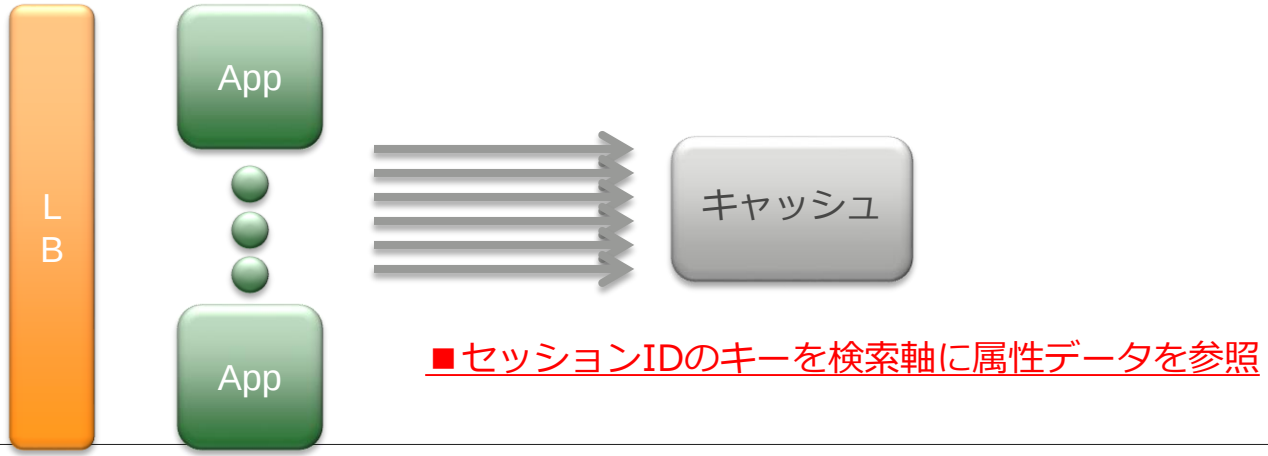


⇒（Point）キャッシュには最新結果が反映される

※Appダウンでネガティブキャッシュが残る場合があるので先にキャッシュDELETEする、TTLをつけるなど要件に合わせて対応が必要

Webアプリとメモリキャッシュ(セッション)

- 共有キャッシュとして使った構成
  - 複数のAppサーバで共有するセッション用メモリ空間を実現
  - 多くの言語やフレームワークが対応済み
  - セッションレプリケーションやロードバランサに依存しない構成が可能



■セッションIDのキーを検索軸に属性データを参照

Memcachedとは？

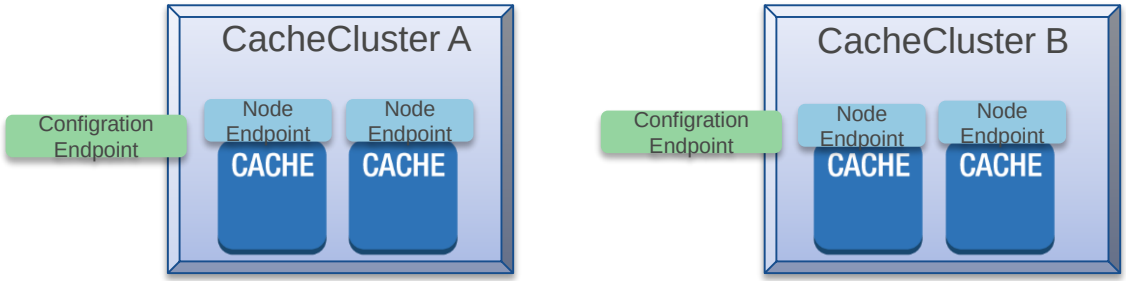
- インメモリ　key-value ストアキャッシュサーバ
  - 2003年にDanga Interactiveが開発(BSDライセンス)
  - ブログサービス「Live Journal」の負荷対策用に作られたもの
  - 多くのサイトで採用 (YouTube, Wikipedia, mixi, etc.)

- 特徴
  - KVSのデファクトスタンダードプロトコル
    - Key-valueのシンプルなデータ構造
    - Telnetでも操作可能
    - パフォーマンス向上を重視
  - 主要機能のみのシンプルな機能
    - アクセス制御などのセキュリティ機能無し
    - マスタノード、シャーディング、レプリケーションなどの機構無し
  - データ削除は明示的、期限、LRU（LeastRecentlyUsed）の3方式

Amazon ElastiCache for memcached

- 特徴
  - 対応バージョン 1.4.5、1.4.14 (2015.8.19現在)
  - memcached プロトコル準拠
  - Cache Clusterという論理グループに、Cache Nodeを複数台起動
  - Cluster Group 全体に
    - Configration Endpoint（各Node EndpointのCNAMEエントリ）
    - Cache Node単体にNode Endpoint

    の２種類のアクセス用のエンドポイントがある
  - バックアップ機能(Snapshot)は持たない



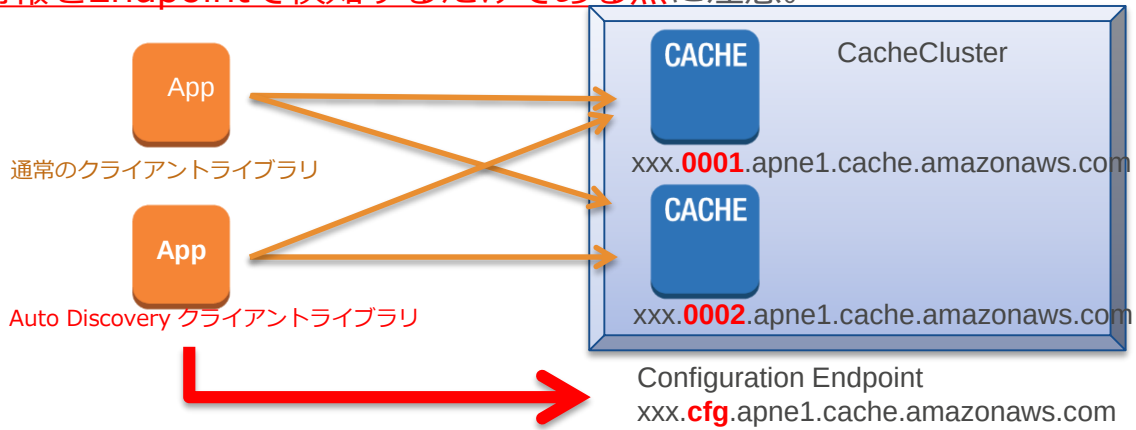
Auto Discovery for memcached

- 従来のクライアント側の設定
  - Cache Clusterの全エンドポイントを接続先として設定する。
- Auto Discoveryクライアント(Java, PHP,.NET)
  - Cache ClusterのConfiguration Endpointを接続先として設定すると、
    全エンドポイントを自動取得・設定し、接続する。
  - Configuration Endpointは、Cache Clusterの ロードバランサー(Proxy) ではなく、
    あくまでも自動検知情報をEndpointで検知するだけである点に注意。

- 確認コマンド

・Memchached 1.4.14以上
>Config get cluster

・Memchached 1.4.14未満
>get AmazonElastiCache:cluster



http://docs.aws.amazon.com/AmazonElastiCache/latest/UserGuide/AutoDiscovery.html

amazon web services

Memcached アクセス用のClient Libraryの提供

- Auto Discovery 用には専用のライブラリがAWSから提供

通常アクセス用Client Library

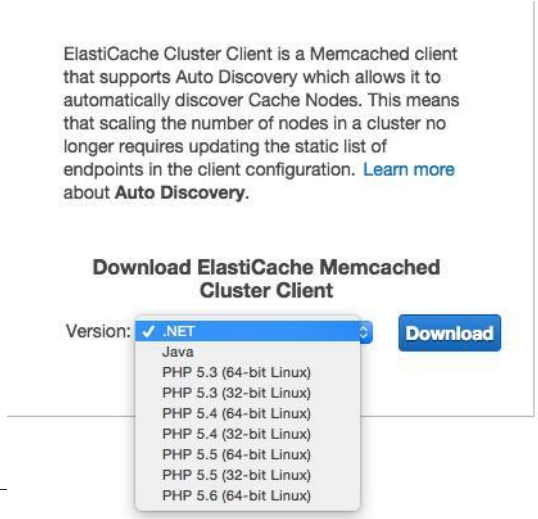
・Memcached サイトからダウンロードすることが可能

https://code.google.com/p/memcached/wiki/Clients

・PHP、Java、.NET、C、C++、Ruby、Python、Perl、他に対応

Auto Discovery用Client Libray

・AWS Management Consoleから取得可能
・PHP、Java、.NET に対応（New！PHP5.6）



CloudWatchによるmemcached の監視

- 監視項目
  - http://docs.aws.amazon.com/AmazonElastiCache/latest/UserGuide/CacheMetrics.Memcached.html
- 主に監視する項目
  - CPUUtilization (CPU使用率)
  - Evictions (キャッシュメモリ不足によるキャッシュアウト発生回数)
  - SwapUsage
  - メモリ使用量
  - CurrConnections
  - http://docs.aws.amazon.com/AmazonElastiCache/latest/UserGuide/CacheMetrics.WhichShouldIMonitor.html

CloudWatchによるmemcached の監視

- プロセスのメモリ使用量
  - memcachedはBytesUsedForCacheItemが現在の使用量を示す。
    こちらを用いて使用可能なメモリ量を計算していただく方法。

    例1）Memcached（cache.t2.micro）でBytesUsedForCacheItemの値が200MBであった場合

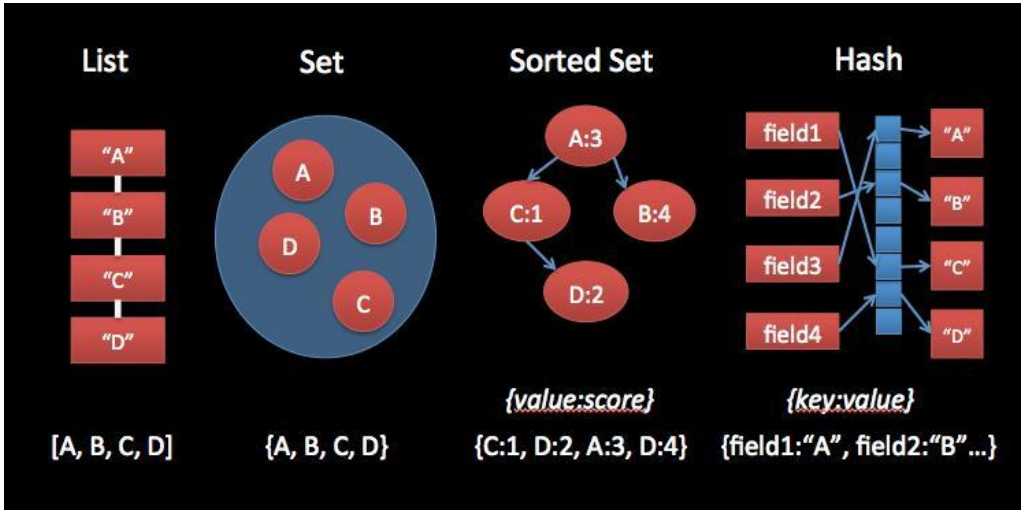
    max_cache_memory 555MB - 200MB = 333MBが残り使用可能なメモリ量

  - ノードタイプごとのメモリ最大値は下記ドキュメントに記載されておりますのでご確認ください。

    ・Memcached のノードタイプ固有のパラメータ(max_cache_memory)
    http://docs.aws.amazon.com/ja_jp/AmazonElastiCache/latest/UserGuide/CacheParameterGroups.Memcached.html#CacheParameterGroups.Memcached.NodeSpecific

Redisとは？

- In-memory Key-Value Store
- 高機能なデータ構造、データ操作
  - List, Set, Sorted Set, Hash
- 永続化機構
  - Snapshot, Append only File
- 冗長化機構
  - Replication
- Pub/Sub機能
- Lua scripting



http://redis.io/

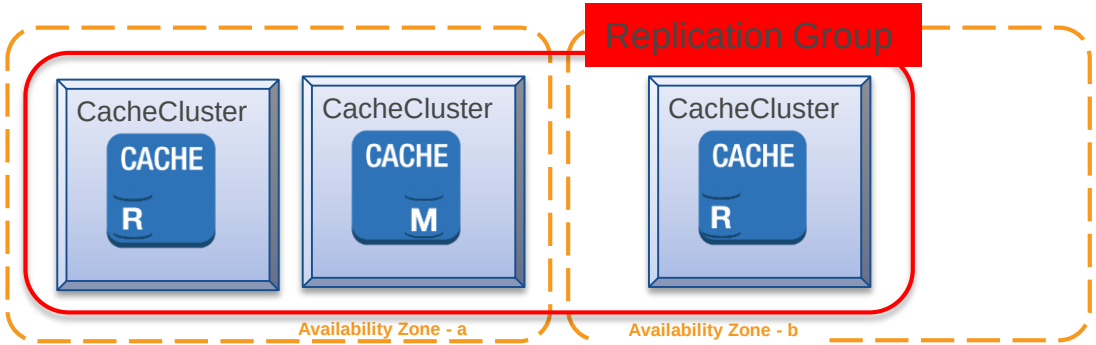
ElastiCache for Redis

- 特徴
  - 対応バージョン: 2.6.13、2.8.6、2.8.19、2.8.21 (2015.8.19現在)
  - 複数のCluster Group から構成されるReplication Group を構成し複数ノードで同期が取れる
  - S3上のスナップショット(RDB)プリロード機能でElastiCache 上へのデータ移行も容易
  - Multi-AZ配置での自動フェイルオーバーにも対応
  - Snapshotベースでのバックアップリスト機能にも対応
  - Redisの特徴をほぼサポート
    - Lua Scripting
    - Pub/Sub
    - Append Only File
    - HyperLogLog(2.8.19以降), ZRANGEBYLEX, ZLEXCOUNT, ZREMRANGEBYLEX.

- 対応しない機能
  - CONFIG, SLAVEOFなど一部コマンドのみ無効
  - パスワード (アクセス制御はセキュリティグループにて実施)

リードレプリカ (Replication)

- 以下の用途に利用可能
  - 耐障害性向上(ただし、非同期レプリケーション)
  - Read性能のスケーリング
- 構成
  - Replication Group内に、マスター 1 台、レプリカ 最大5台
  - Replica of Replica は未対応

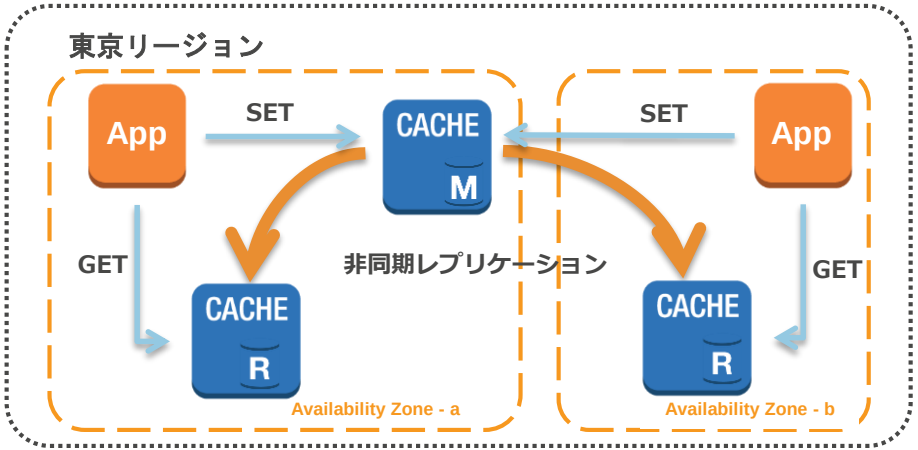


http://docs.aws.amazon.com/AmazonElastiCache/latest/UserGuide/Replication.html
http://docs.aws.amazon.com/AmazonElastiCache/latest/UserGuide/ManagingReplication.html

アベイラビリティゾーンをまたいだReplication構成

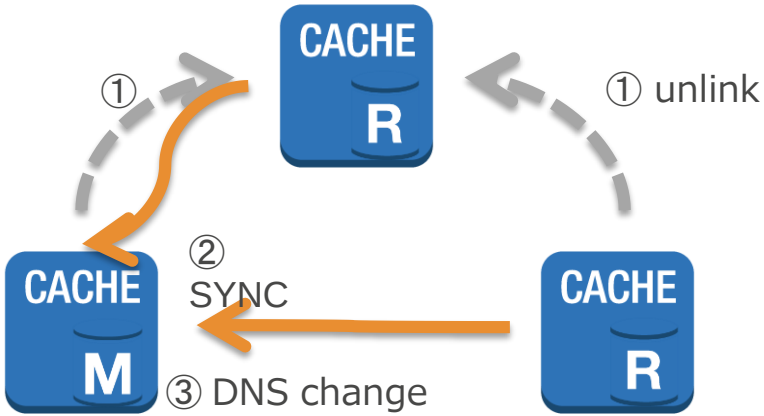
リードレプリカを複数のアベイラビリティゾーンにデプロイ可能

- 同一AZのリードレプリカを参照し高速なデータ取得が可能に
- プライマリノード側のAZ障害時のデータ保全が可能に



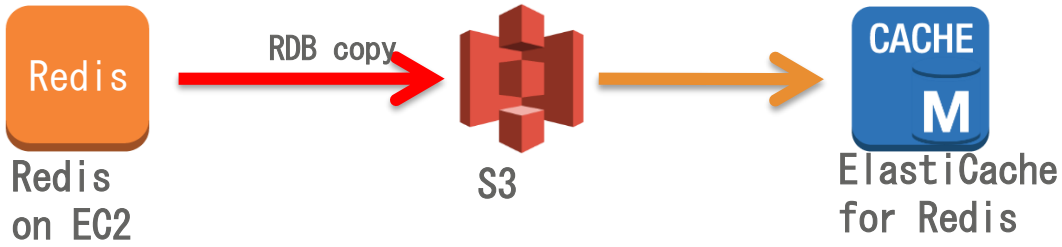
リードレプリカ昇格

リードレプリカをプライマリに昇格可能
　昇格は数分が必要
- ダウンが発生するため、クライント側でエラーハンドリングは必要
- アプリ修正不要（プライマリのendpointが変わらない）



RDBデータのプリロード

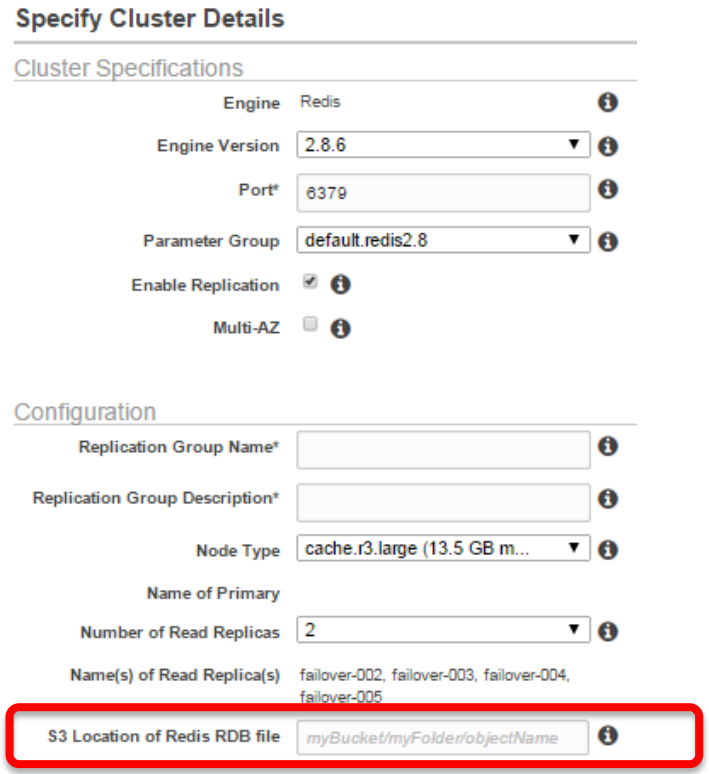
- 既存のRedisからのElastiCacheへのデータ移行に
  - 既存のRedisで取得したRDBファイルをS3に保存
  - キャッシュノード起動時に S3上のRDBファイルを読み込み
- 注意点
  - RDBファイルのバージョン互換性を確認
  - 保存したS3に対して、ElastiCacheが参照可能なパーミッションが必要
  - キャッシュノードタイプがサポートするメモリサイズを超えるRDBは読み込み不可。
    (起動時にエラーが発生)



http://docs.aws.amazon.com/AmazonElastiCache/latest/UserGuide/ManagingCacheClusters.html#ManagingCacheClusters.SeedingRedis

データのプリロード設定手順

S3からスナップショットをプリロード
- プライマリノード起動時にS3パスを指定
  (例: mybacket/path/data.rdb)



プリロードの応用: バックアップ＆リストア

- EC2上のRedisをslaveとして マスタに接続可能
- EC2上のRedis slave側でRDBファイルをS3へバックアップして
  おくことで、クラスタ起動時にS3からプリロード可能

http://redis.io/topics/persistence

http://docs.aws.amazon.com/AmazonElastiCache/latest/UserGuide/ManagingCacheClusters.html#ManagingCacheClusters.RedisSnapshots

Append-Only Files(AOF)について

- AOFとは
  - Redisの機能で受信した全コマンド(操作)をローカルストレージ上のAOFに追記
  - キャッシュノードreboot時にキャッシュデータの復元が可能
    - メンテナンス時のreboot時にデータを保持可能
- デフォルトでは off
  - パラメータグループで appendonly をyesに変更し有効化
  - cache.t1.microは未対応(appendonly=yesでも無視される)
- 注意点
  - ノード障害によるノード入れ替えが発生した場合はAOFが喪失する
  - データ保全のためには最低１台のリードレプリカ構成を推奨
  - 耐障害性という観点からMAZを利用した構成を推奨

http://docs.aws.amazon.com/AmazonElastiCache/latest/UserGuide/CacheParameterGroups.Redis.html#CacheParameterGroups.Redis.AOF

耐障害性: AOF とマルチ AZ の比較

Redis の AOF が有効になっている場合、サーバーによって受信されたすべての書き込みオペレーションはログに記録されます。このため、AOF は非常に大きくなり、対象のデータセットの .rdb ファイルよりも大きくなる場合があります。ElastiCache は、サイズに制限があるローカルストレージ（インスタンスストア）を利用するため、AOF を有効にすると、ディスク容量不足の問題が発生する可能性があります。

耐障害性のための Redis AOF の有効化

AOF ファイルが復旧シナリオで役に立つという理由で AOF を有効にする場合があります。ノードの再起動やサービスのクラッシュが発生したときに、Redis は AOF ファイルから更新を再生することによって、再起動やクラッシュによって消失したデータを復旧します。

> **Warning**
>
> AOF はすべての障害のシナリオに対応できるわけではありません。たとえば、基になる物理サーバーでハードウェア障害が発生したためノードでエラーが発生した場合、ElastiCache は別のサーバーで新しいノードをプロビジョニングします。この場合、AOF ファイルは使用できなくなり、データの復旧には使用できません。したがって、Redis はコールドキャッシュを使って再開されます。

耐障害性に対するより適切なアプローチとしての Redis マルチ AZ の有効化

データの消失に備えて AOF を有効にしている場合は、AOF の代わりに、マルチ AZ を有効にしたレプリケーショングループの使用を検討してください。Redis レプリケーショングループを使用している場合、レプリカに障害が発生すると、レプリカは自動的に置き換えられ、プライマリクラスターと同期されます。Redis レプリケーショングループでマルチ AZ が有効になっており、プライマリに障害が発生した場合、プライマリはリードレプリカにフェイルオーバーされます。この機能は、AOF ファイルからプライマリを再構築するよりも高速です。信頼性を高め、より迅速な復旧を可能にするため、異なるアベイラビリティーゾーンに 1 つ以上のリードレプリカを持つレプリケーショングループを作成し、AOF を使用する代わりにマルチ AZ を有効にすることをお勧めします。このシナリオで AOF は必要ないため、ElastiCache はマルチ AZ レプリケーショングループで AOF を無効にします。

詳細については、次のトピックを参照してください。

- AOF（Append-Only File）
- レプリケーショングループとリードレプリカ (Redis)
- Redis レプリケーショングループを使用したマルチ AZ

CloudWatchによるRedis の監視

- 主に監視する項目
  - CPUUtilization (CPU使用率)
  - Evictions (キャッシュメモリ不足によるキャッシュアウト発生回数)
  - CurrConnections
  - Replica Lag (レプリケーション遅延)
  - メモリ使用量
- CPU 使用率を監視する際の注意点
  - Redisは シングルスレッドなので、1コアで動作
  - cache.m1.xlarge(4コア)だと、 25% (100% / 4) が最大値となる
  - Alert設定時には、上記点に注意する
  - http://docs.aws.amazon.com/AmazonElastiCache/latest/UserGuide/CacheMetrics.WhichShouldIMonitor.html

CloudWatchによるRedis の監視

- プロセスのメモリ使用量
  - RedisはBytesUsedForCacheが現在の使用量を示す。
    こちらを用いて使用可能なメモリ量を計算していただく方法。

    Redis（cache.t2.micro）でBytesUsedForCacheの値が200MBであった場合

    maxmemory 約560MB - 200MB = 360MBが残り使用可能なメモリ量

  - ノードタイプごとのメモリ最大値は下記ドキュメントに記載されておりますのでご確認ください。

    ・Redis のノードタイプ固有のパラメータ(maxmemory)
    http://docs.aws.amazon.com/ja_jp/AmazonElastiCache/latest/UserGuide/CacheParameterGroups.Redis.html#CacheParameterGroups.Redis.NodeSpecific

CloudWatchによるRedis の監視

- swapUsage
  - Redis の BGSAVE の挙動により、メモリの多くを使い、かつ書き込みが多いユースケースではswapが発生して問題になる可能性があります。メモリに関するパラメータで回避できるため、詳しくはドキュメントのベストプラクティスをご覧ください
  - http://docs.aws.amazon.com/ja_jp/AmazonElastiCache/latest/UserGuide/BestPractices.html

Amazon ElastiCache の実装のベストプラクティス

このトピックでは、Amazon ElastiCache を実装するためのベストプラクティスを示します。

Topics

- Redis スナップショットを作成するための十分なメモリがあることの確認
- 耐障害性: AOF とマルチ AZ の比較
- 効率的な負荷分散のための ElastiCache クライアントの設定

Redis スナップショットを作成するための十分なメモリがあることの確認

Redis ElastiCache を使用する場合、Redis は多くの場合 BGSAVE コマンドを呼び出します。

- バックアップ/復元のためのスナップショットを作成するとき。
- プライマリとレプリケーショングループ内のレプリカを同期させるとき。
- Redis の AOF（Append-Only File）機能を有効にするとき。
- レプリカをマスターに昇格するとき（プライマリ/レプリカの同期が実行される）。

Redis が BGSAVE を実行するときは、常に、このプロセスのオーバーヘッドに対応するのに十分なメモリが利用できる必要があります。十分なメモリを利用できない場合、このプロセスは失敗します。このため、Redis クラスターの作成時には、十分なメモリがあるノードインスタンスタイプを選択することが重要です。

BGSAVE プロセスとメモリの使用

BGSAVE が呼び出されると、Redis は常にそのプロセスを生成（フォーク）します（Redis はシングルスレッドであることを思い出してください）。1 つのフォークがデータをディスクの Redis .rdb スナップショットファイルに永続化します。もう 1 つのフォークは、すべての読み取りと書き込みのオペレーションを処理します。スナップショットがポイントインタイムスナップショットであることを保証するために、すべての書き込みオペレーションが、データ領域とは別の使用可能なメモリ領域に書き込まれます。

データをディスクに永続化しながら、すべての書き込みオペレーションを記録するのに十分なメモリが使用できる限り、メモリ不足の問題は発生しません。次のいずれかに該当する場合は、メ

CloudWatchによるRedis の監視

- セッション数の確認

http://docs.aws.amazon.com/AmazonElastiCache/latest/UserGuide/CacheMetrics.html

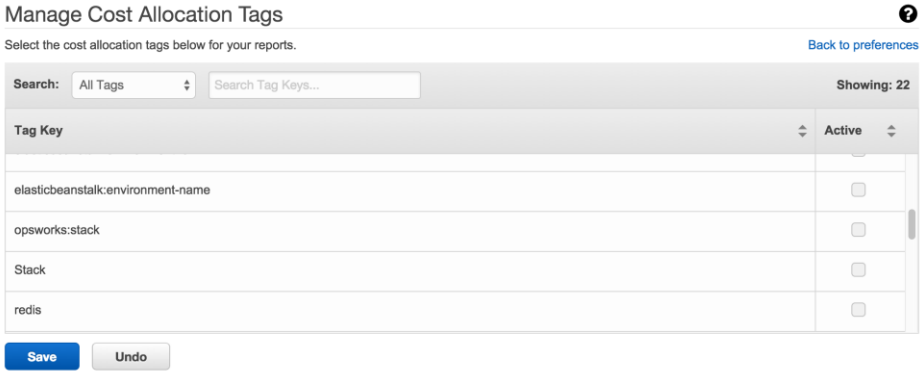
2．前回Webinar(2015年1月)からのアップデート

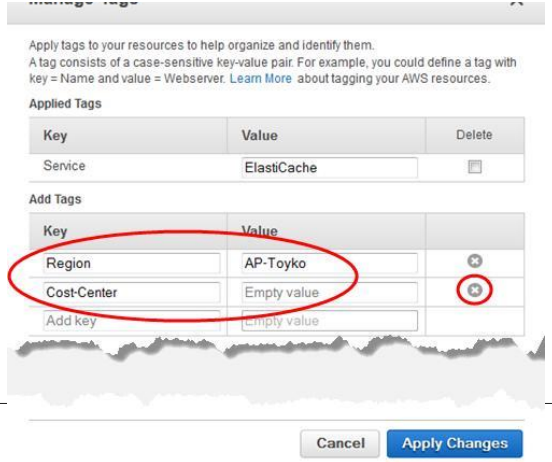
アップデート一覧

- 2015年 1月 – ElastiCache GovCloud(US)、Frankfult(EU)リージョン対応
- 2015年 2月 – Redis Cost allocation tags対応
- 2015年 3月 – Redis 2.8.19対応
- 2015年 7月 – Redis 2.8.21対応、Memcached Auto Discovery PHP 5.6対応

Redis Cost allocation tags

コスト割り当てタグの管理で
Redisが対応





PHP 5.6対応

- Auto Discovery Client for PHP 5.6

Announcement: Auto Discovery Client for PHP 5.6 and Open Source

Posted By: DanZ@AWS

Created in: Forum: Amazon ElastiCache

Posted on: Jul 30, 2015 9:58 PM

Amazon ElastiCache now supports Cluster Client for PHP 5.6. This can be downloaded from the AWS Management Console under the "ElastiCache Cluster Client" tab. For details on installing the client, please see here. For more information about Node Auto Discovery please see here.

We have also open sourced the code of the Amazon ElastiCache Cluster Client for PHP. It is available on our GitHub repository here. For instructions on compiling the Cluster Client please see here.

Redis 2.8.19、2.8.21対応によるコマンド追加

- Redis 2.8.19対応により、2.8.6以降で追加された機能が利用可能に。
  - HyperLogLog:PFCOUNT、PFADD、PFMERGEの追加
  - ZRANGEBYLEX、ZLEXCOUNT、ZREMRANGEBYLEXの追加
  - ROLE、BITPOS、COMMANDの追加

HyperLogLog

HyperLogLogはセット内のユニークな要素数を効率的で高速に近似することが出来るコンパクト(各Key毎に12KB)なRedisのデータ・タイプです。(カーディナリティと呼ばれているものです)。

推定値(0.81パーセントの標準誤差で)を取得することが可能。

Example PFADD

redis> PFADD hll a b c d e f g
(integer) 1
※hllというkeyが変更されたので1が戻り値

Example PFCOUNT

redis> PFCOUNT hll
(integer) 7
redis>
※a,b,c,d,e,f,gという7種類の値が存在するので戻り値が7

Example PFMERGE

redis> PFADD hll1 foo bar zap a
(integer) 1
redis> PFADD hll2 a b c foo
(integer) 1
redis> PFMERGE hll3 hll1 hll2
OK
redis> PFCOUNT hll3
(integer) 6
※hll1とhll2をマージしたhll3というkeyを作成し合算した値でPFCOUNTを実行。計6種類あるので戻り値が6

ZRANGEBYLEX、ZLEXCOUNT、ZREMRANGEBYLEX

Sorted Set型で

- ZRANGEBYLEX
  （範囲指定探索）
- ZLEXCOUNT
  （範囲指定カウント）
- ZREMRANGEBYLEX
  （範囲指定削除）

が使用可能。

Example ZRANGEBYLEX

```
redis> ZADD myzset 0 a 0 b 0 c 0 d 0 e 0 f 0 g
(integer) 7
redis> ZRANGEBYLEX myzset - [c
1) "a"
2) "b"
3) "c"
redis> ZRANGEBYLEX myzset - (c
1) "a"
2) "b"
redis> ZRANGEBYLEX myzset [aaa (g
1) "b"
2) "c"
3) "d"
4) "e"
5) "f"
redis>
```

Example ZLEXCOUNT

```
redis> ZADD myzset 0 a 0 b 0 c 0 d 0 e
(integer) 5
redis> ZADD myzset 0 f 0 g
(integer) 2
redis> ZLEXCOUNT myzset - +
(integer) 7
redis> ZLEXCOUNT myzset [b [f
(integer) 5
redis>
```

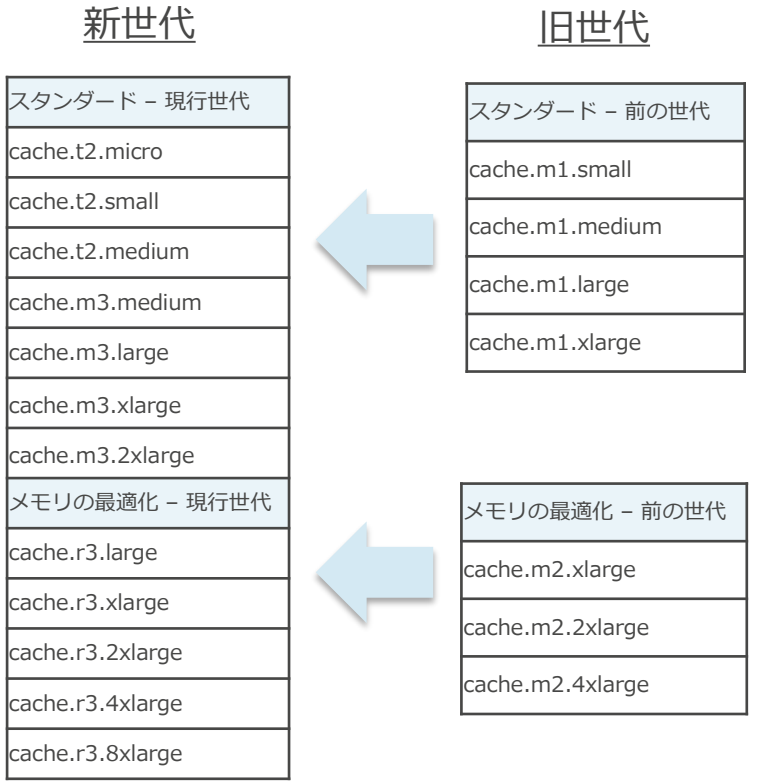
Example ZREMRANGEBYLEX

```
redis> ZADD myzset 0 aaaa 0 b 0 c 0 d 0 e
(integer) 5
redis> ZADD myzset 0 foo 0 zap 0 zip 0 ALPHA 0 alpha
(integer) 5
redis> ZRANGE myzset 0 -1
1) "ALPHA"
 2) "aaaa"
 3) "alpha"
 4) "b"
 5) "c"
 6) "d"
 7) "e"
 8) "foo"
 9) "zap"
10) "zip"
redis> ZREMRANGEBYLEX myzset [alpha [omega
(integer) 6
redis> ZRANGE myzset 0 -1
1) "ALPHA"
2) "aaaa"
3) "zap"
4) "zip"
redis>
```

バージョン追加とインスタンスタイプ

- バージョンの追加
  - Redis 2.8.19、2.8.21がサポート。

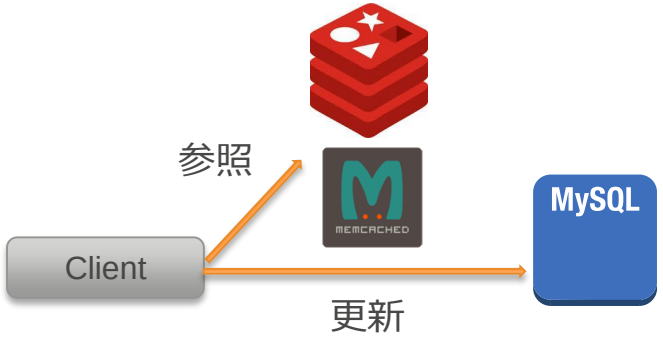
- インスタンスタイプ
  - EC2、RDS同様に新世代への移行を推奨
  - T1、T2ではAOF、マルチAZ、バックアップ・復元が利用不可。

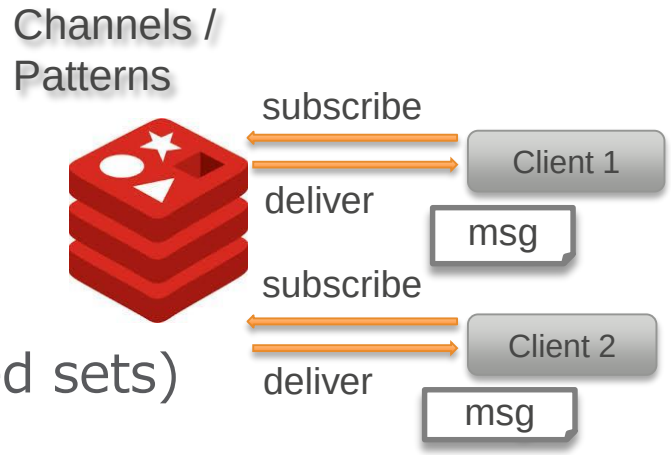


3．ユースケース、注意点、システム構成例

ユースケース

- memcached , redis
  - データキャッシュ
  - セッションストア
- redis
  - 分散カウンター (atomic counter)
  - リアルタイム・メトリック (bitmaps)
  - キュー (lists)
  - メッセージング (pub/sub)
  - リーダーボード,スコアランキング (sorted sets)



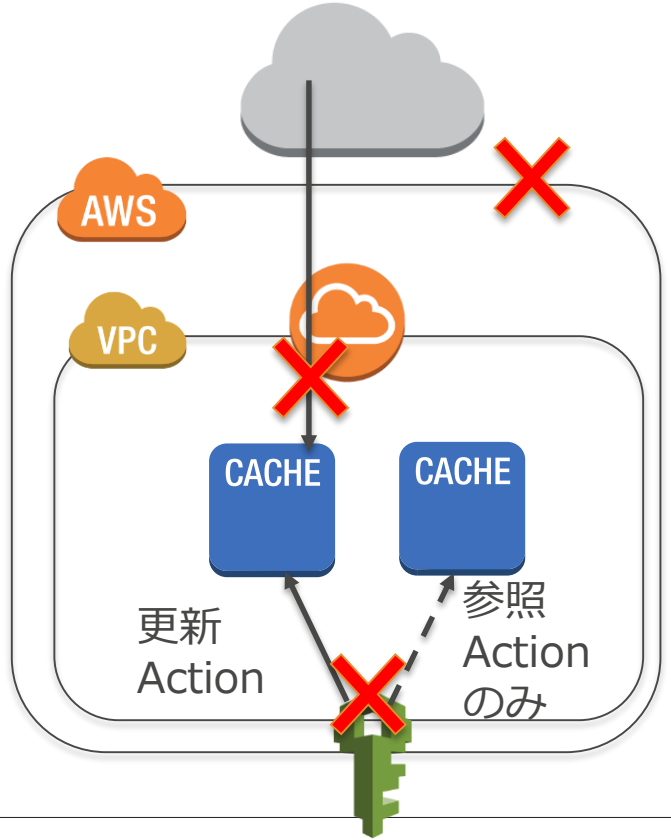


2015年8月時点でのElastiCache利用上の注意点

2015年8月時点での注意点

- VPC内に起動した場合、VPC外からアクセスできない。
  - Endpoint がPublic Facing対応していないため。

- IAMでキャッシュノードのリソース制御できない
  - ARN(Amazon Resource Name)が無いため。

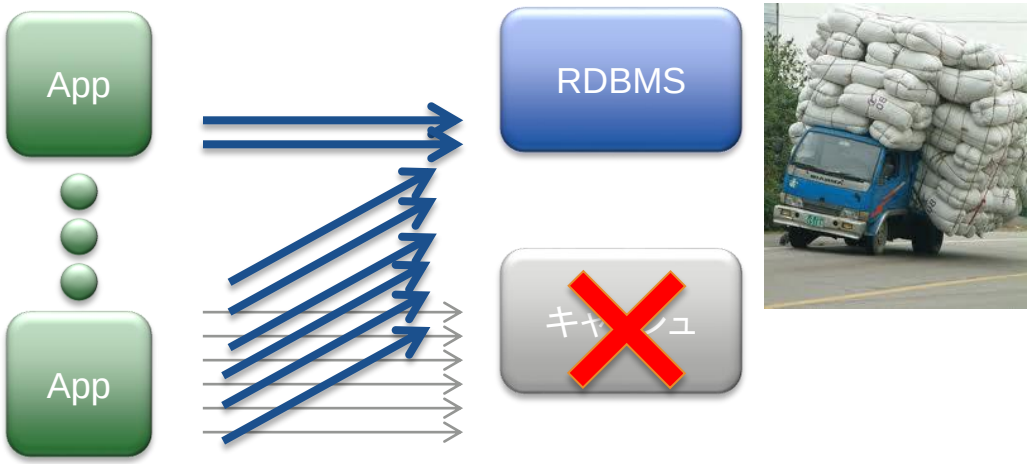


メモリキャッシュを利用する際の注意点

- システムの性質を考える
  - 一般的なWebシステムは、参照：更新≒9:1
  - 更新クエリの割合が大きいシステムでは効果薄
  - キャッシュミス時のペナルティ対策（定期更新等）
  - キャッシュ喪失時の対策も織り込む
- キャッシュするデータの性質を考えてキャッシュする
  - 有効期限の短いデータは不向き
  - 参照頻度の低いデータは、メモリ効率が悪い
- トランザクション・コヒーレンシを考える
  - 一貫性が必要なデータは、慎重に設計・実装する
  - DBより古いキャッシュを使わない工夫

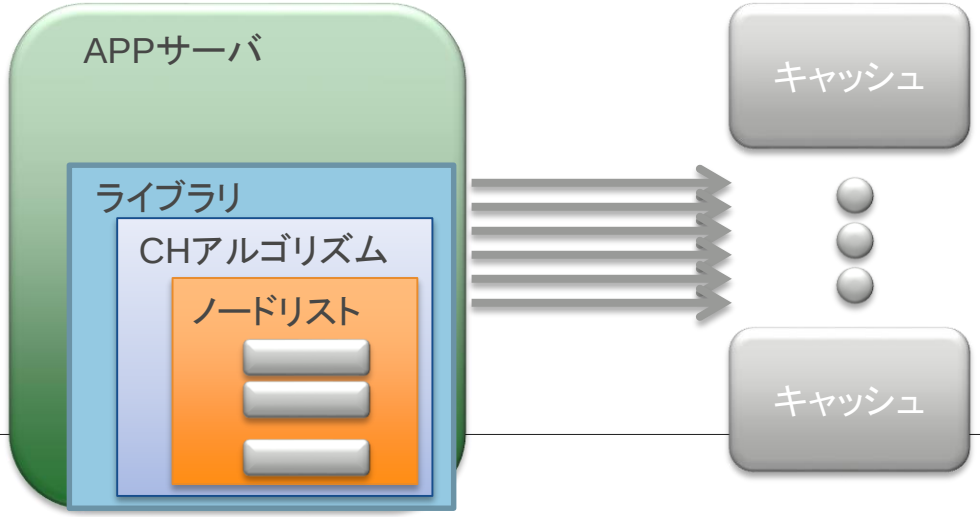
キャッシュのSPOF（単一障害点）について

- 単一キャッシュノード構成の課題
  - 障害時のデータ喪失による影響は少ない（はず）
    - ただしDBが過負荷になり、システムスローダウンやダウンも
    - キャッシュ容量増強のためにはスケールアップ(容易ではない)



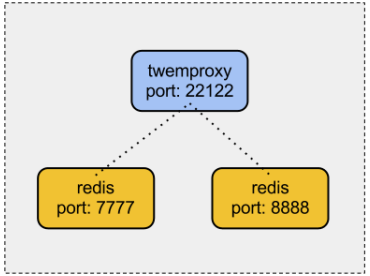
キャッシュのクラスタ化(Consistent Hashing)

- Consistent Hashingの特徴
  - Appサーバ側でConsistent Hashingアルゴリズムで振り分ける
    - ノード障害時のキャッシュ喪失が限定的
    - ノード追加で総キャッシュ容量を増やしやすい
    - **ノード数変更時のリバランスコスト**が限定的



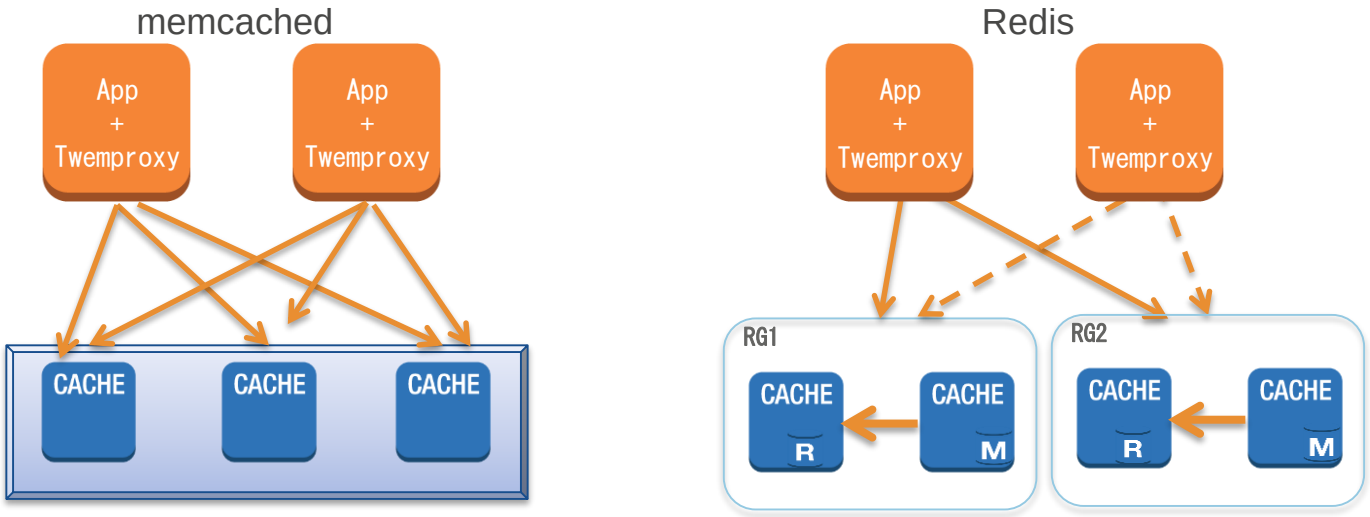
キャッシュのクラスタリングの方法

- クライアントライブラリでの実装
  - PHPなどで使うlibmemcachedは標準でdistribution アルゴリズム実装済み
- サードパーティソフトでの実装：Twemproxy
  - Twitter社が開発したMemcached/Redis用のproxy
  - 複数サーバーのシャーディング (consistent hashing)
  - キャッシュサーバーの接続管理
  - 一部未サポートのコマンドもあり
- 参考
  - Consistent hashing - Wikipedia, the free encyclopedia
    - http://en.wikipedia.org/wiki/Consistent_hashing
  - memcachedを知り尽くす：第4回　memcachedの分散アルゴリズム | gihyo.jp … 技術評論社
    - http://gihyo.jp/dev/feature/01/memcached/0004?page=3
  - Partitioning: how to split data among multiple Redis instances.
    - http://redis.io/topics/partitioning



Twemproxyの構成例

TwemproxyがSPOFとならないように、各アプリサーバ上Twemproxyを起動し、
アプリケーションはローカルホストのtwemproxyにアクセスする



同一リージョン内のAZを跨いだクラスタ対応

- AZを跨いでクラスタを構成する事が可能

Memcached

Configure Advanced Settings

Network & Security

Cache Subnet Group: net (vpc-88714ae1)

Note: Nodes in a Cross Availability Zones cluster will only be placed in zones that are part of the selected Cache Subnet Group.

Availability Zone(s): Specify Zones

Selected Number of Nodes 2

Zone	Number of Nodes
ap-northeast-1a	
ap-northeast-1c	

Nodeを複数選択し、Availability Zone(s)で「Specify Zones」を選択し、各Zoneに配置するノード数を指定する。

Redis

Configure Advanced Settings

Network & Security

Cache Subnet Group: net (vpc-88714ae1)

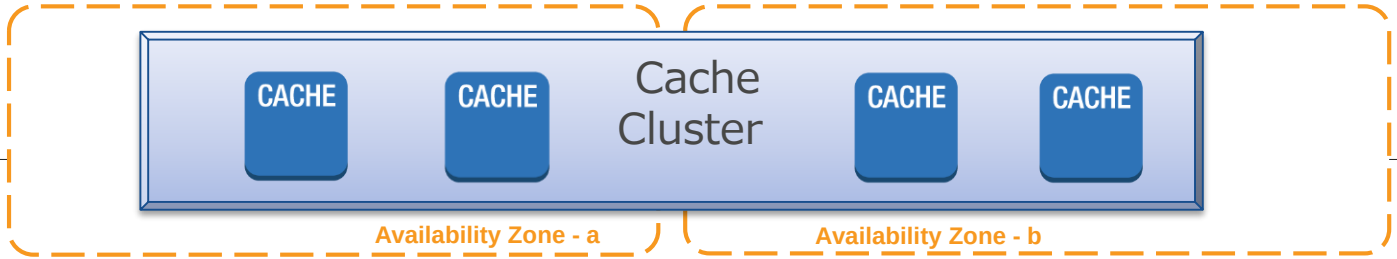
Availability Zone(s)

Primary: a-001 ap-northeast-1a

Read Replica(s): a-002 ap-northeast-1c

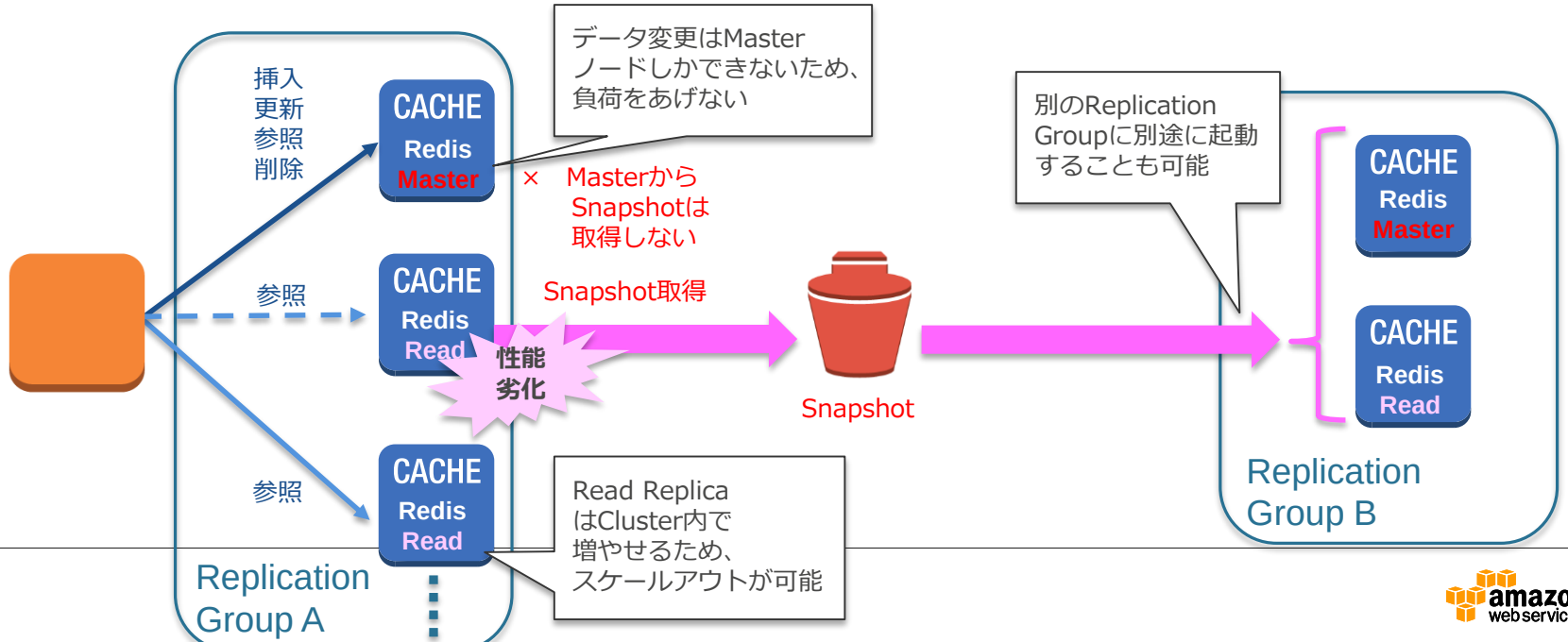
a-003 ap-northeast-1a

Read Replicaを選択し、デフォルトでAZを指定できるため、PrimaryとRead Replicaを任意のAZに配置する。



ElastiCache Redis バックアップリストア機能

- ElastiCache Redis が Snapshot機能によるバックアップリストア機能に対応
  （キャッシュサービスは障害時にデータが消失するため、要件次第でバックアップが必要）
- Snapshot取得時はRedisで内部的にBGSAVEを行っているため、
  性能懸念がある場合はRead Replicaから取得を推奨

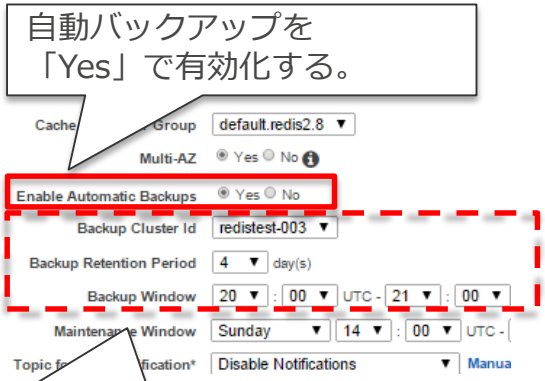


ElastiCache Redis バックアップリストア手順

- Snapshot取得は、自動でも手動でも可能。

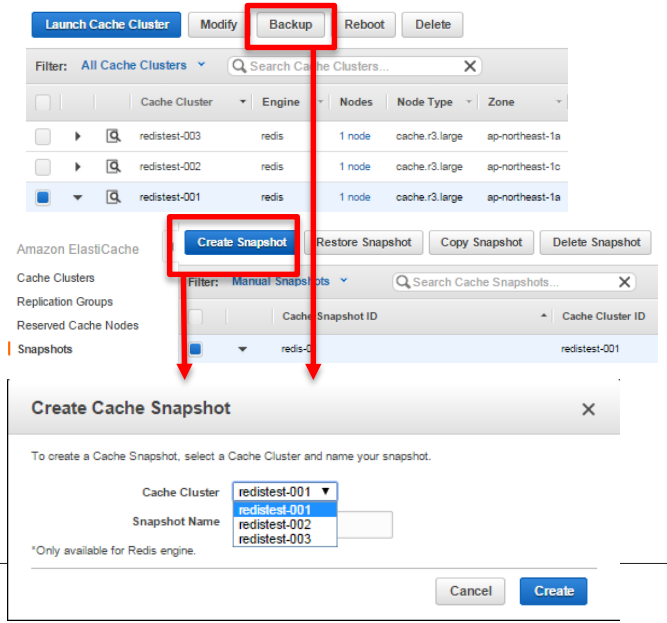
自動バックアップ手順

Clusters全体に対して、「Create」若しくは「Modify」で設定。



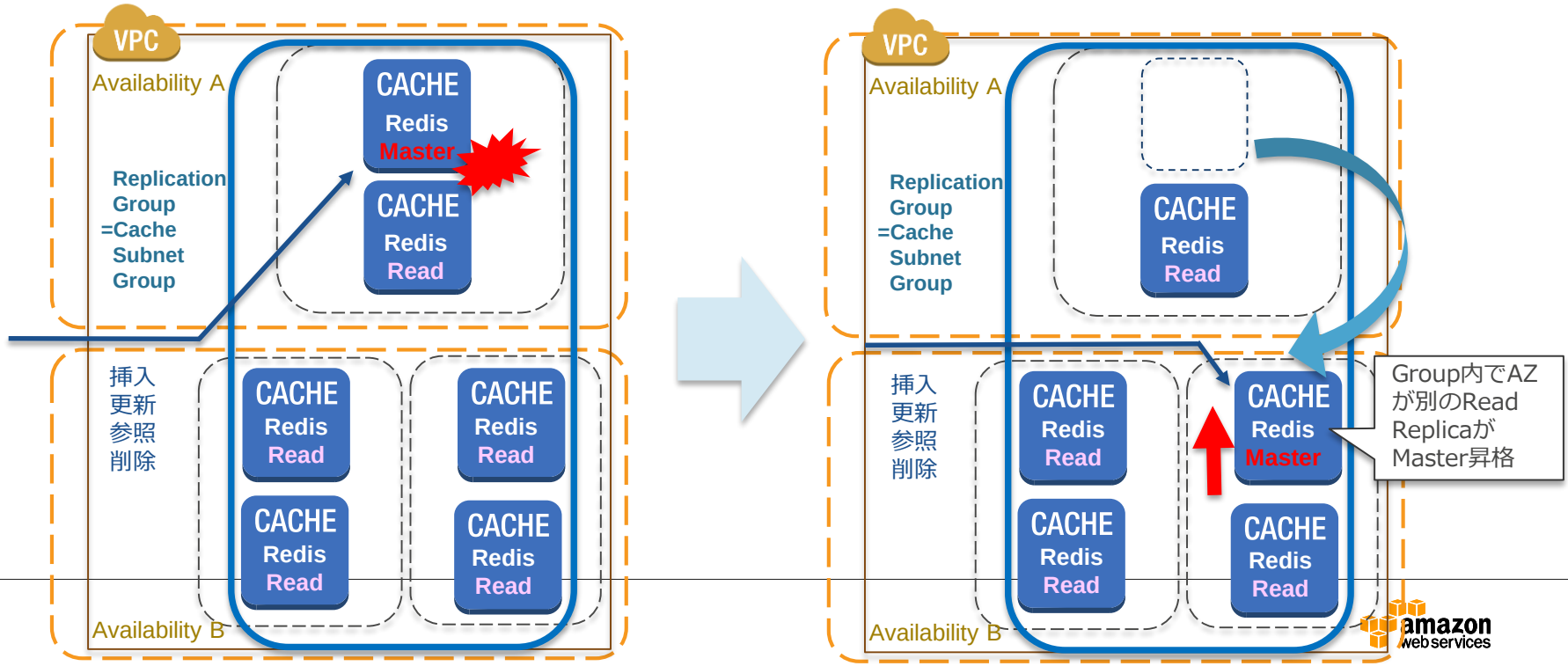
手動バックアップ手順

「Cache Clusters」か「Snapshot」のどちらかから実施

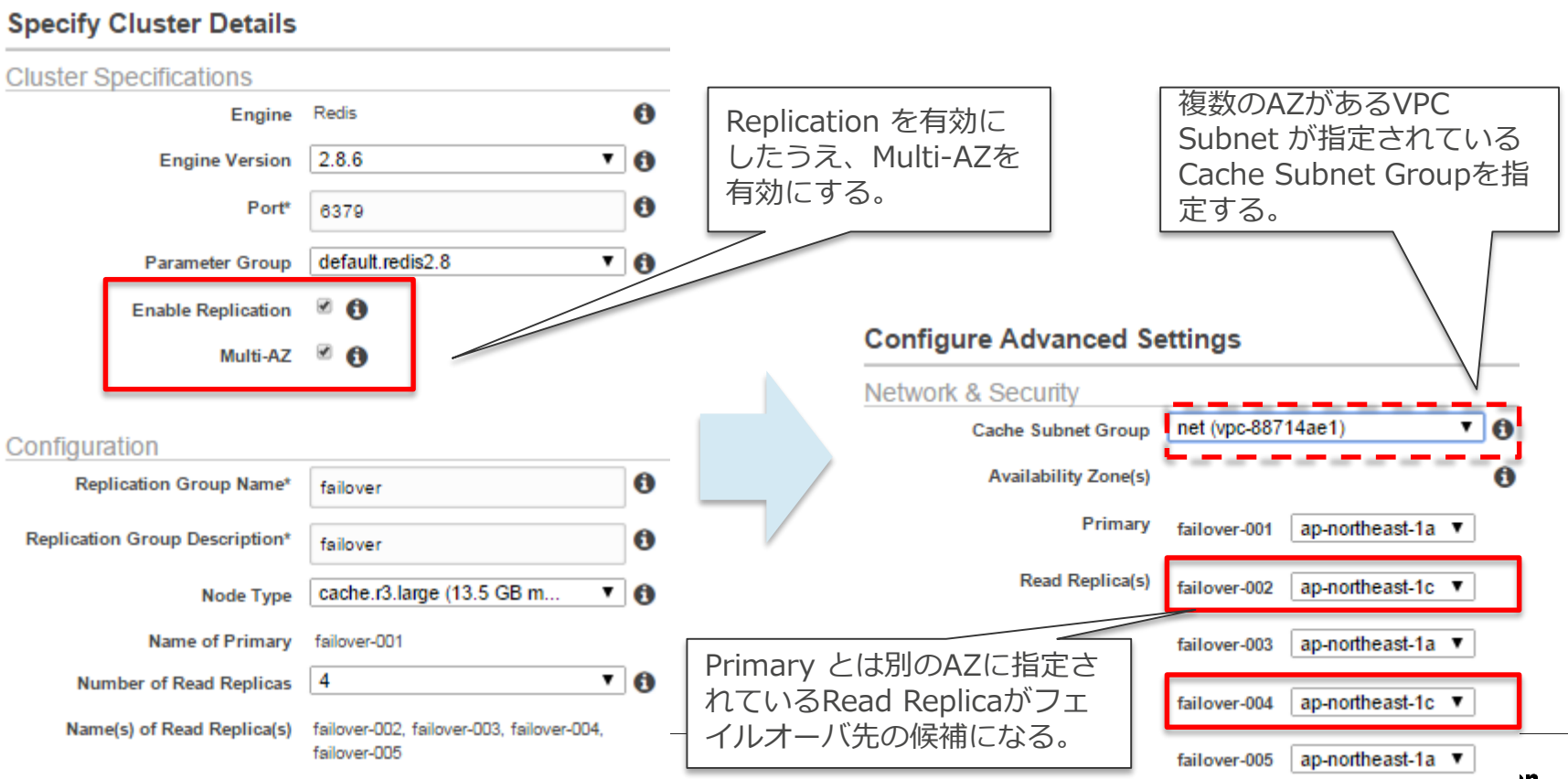


ElastiCache Redis Multi-AZ自動フェイルオーバー機能

- ElastiCache Redis のMaster Nodaが障害時に自動フェイルオーバする機能
- フェイルオーバは同じReplication Group内の別AZにRead ReplicaがMasterに昇格し実現



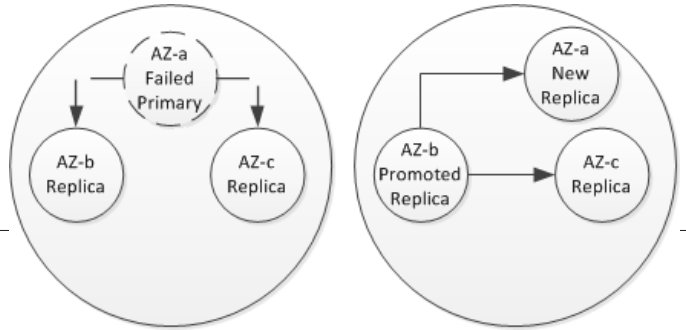
ElastiCache Redis Multi-AZ自動フェイルオーバー設定



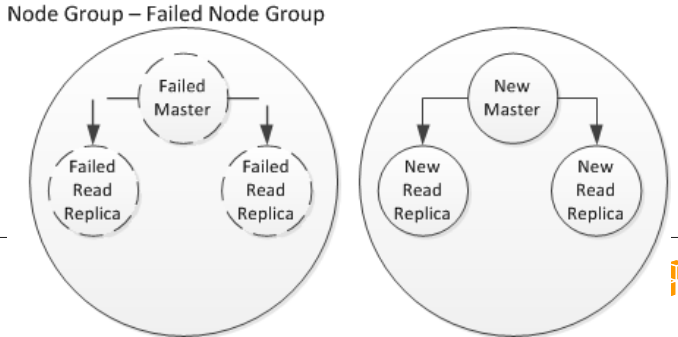
自動ファイルオーバ無効時の障害時の挙動

	リードレプリカあり	リードレプリカなし
障害	プライマリ障害	ノード全面障害
挙動	1. キャッシュイベントをSNSで通知を受取る 2. 再起動を試みる 3. 再起動失敗した場合、フェイルオーバーが必要 4. FO後プライマリDNS自動切替	1. キャッシュイベントをSNSで通知を受取る 2. 新しいノードを立ち上げ 3. DNS切替
影響	• フェイルオーバー完了まで書込不可 • それまでの一定の時間（秒単位）RR読込不可	キャッシュデータロスト
対策	• SNS通知を設定する • クライント/アプリでエラーハンドリング（retry, DBから取得など）	• SNS通知を設定する • キャッシュ喪失時の対策も織り込む

プライマリ障害



ノード全面障害



amazon web services

4. 価格、まとめ

価格

- オンデマンド キャッシュノード
  - 初期費用無し、時間単位の従量課金モデル
  - MemcachedとRedisでどちらも料金は変わらず
- リザーブド キャッシュノード
  - 予約金を支払うことで時間当たり価格を割引(最大70%節減)
  - EC2と違いアベイラビリティゾーンの指定が不要
- バックアップストレージ
  - Redis向け機能
  - 各クラスタに対して1つのSnapshotは無料
  - 2つ以上のSnapshotから毎月 0.085 USD/GBが課金
- AZ間データ転送量
  - ElastiCache間の通信は課金対象外
  - EC2とElastiCache間でAZを超える場合 0.01 USD/GB が課金

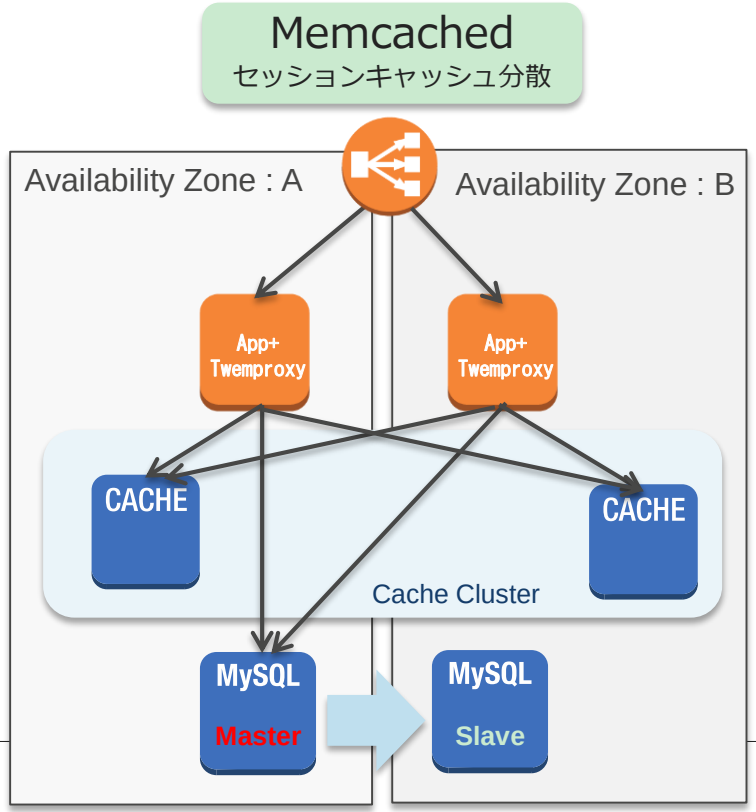
時間あたりの料金（東京リージョン）
※2015年8月18日現在

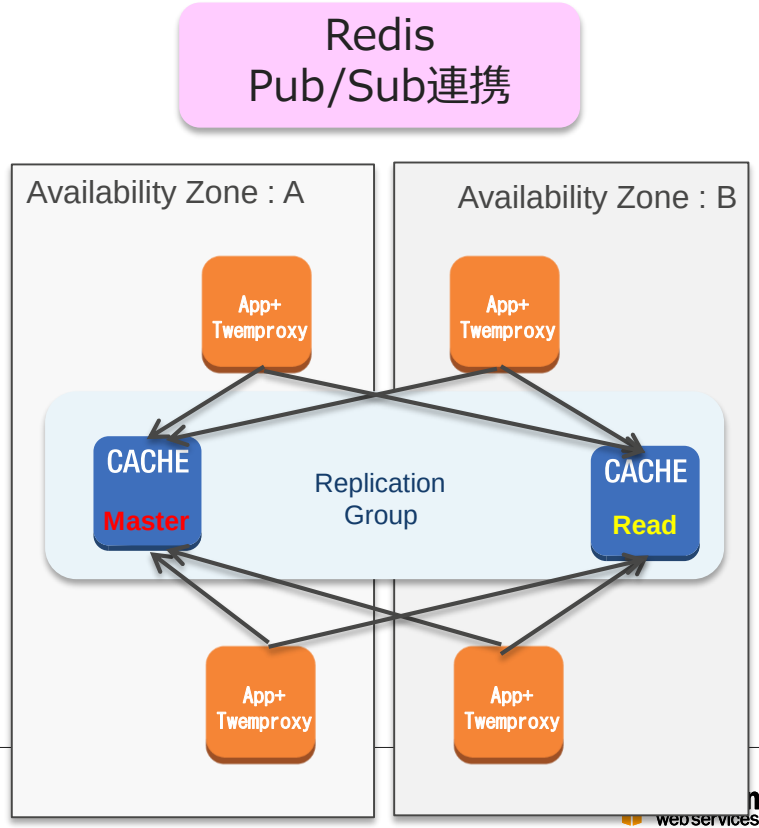
Standard Cache Nodes - Current Generation	
cache.t2.micro	$0.026
cache.t2.small	$0.052
cache.t2.medium	$0.104
cache.m3.medium	$0.120
cache.m3.large	$0.240
cache.m3.xlarge	$0.485
cache.m3.2xlarge	$0.965
Memory Optimized Cache Nodes - Current Generation	
cache.r3.large	$0.273
cache.r3.xlarge	$0.546
cache.r3.2xlarge	$1.092
cache.r3.4xlarge	$2.184
cache.r3.8xlarge	$4.368

http://aws.amazon.com/jp/elasticache/pricing/
http://aws.amazon.com/jp/elasticache/reserved-cache-nodes/

Amazon ElastiCache のまとめ

- 構築・運用が非常に容易で、Multi-AZ構成の高可用性機能にも対応





参考資料

- Amazon ElastiCache Document
  http://aws.amazon.com/jp/documentation/elasticache/
- Amazon ElastiCache FAQ
  http://aws.amazon.com/jp/elasticache/faqs/
- Amazon ElastiCache Pricing
  http://aws.amazon.com/jp/elasticache/pricing/
- Memcached
  http://memcached.org/
- Redis
  http://redis.io/

Q&A

Webinar資料の配置場所

- AWS クラウドサービス活用資料集
  - http://aws.amazon.com/jp/aws-jp-introduction/

プロダクト別：				
Amazon S3		AWSマイスターシリーズ Re:Generate Amazon Simple Storage Service (S3)	Slideshare	PDF
Amazon Glacier		AWSマイスターシリーズ Reloaded Amazon Glacier Amazon Glacierのご紹介 機能編	Slideshare（Reloaded） Slideshare（機能編）	PDF（Reloaded） PDF（機能編）
Amazon Route 53		AWSマイスターシリーズ Re:Generate	Slideshare	PDF

公式Twitter/Facebook
AWSの最新情報をお届けします

@awscloud_jp

もしくは
http://on.fb.me/1vR8yWm

最新技術情報、イベント情報、お役立ち情報、お得なキャンペーン情報などを
日々更新しています！

ご参加ありがとうございました。