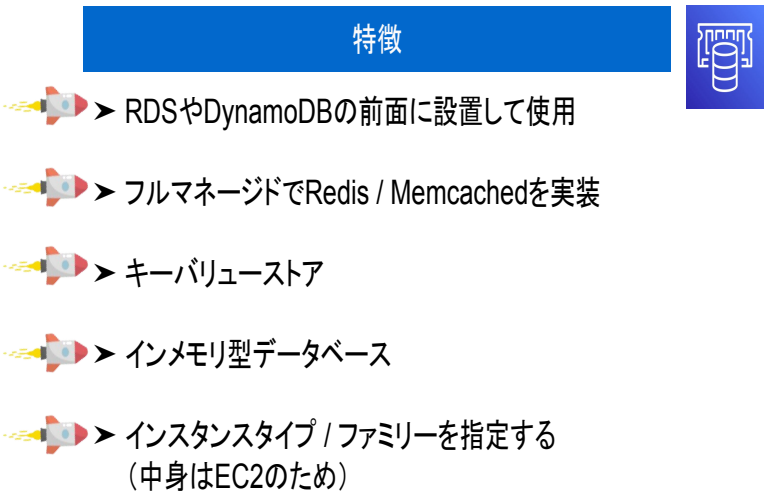
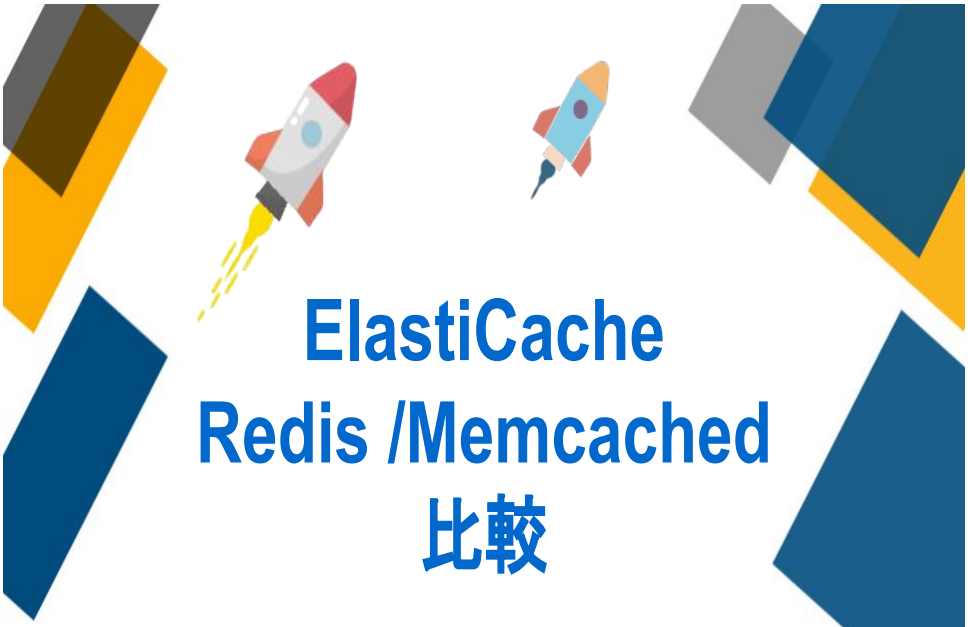
- インメモリ型のデータベースサービスです。

ElastiCacheの特徴



- RDSやDynamoDBの前面に設置して使うタイプということ
  - 要は、補助的な役割があり、RDSのようにメインのデータベースとしては使われません。
- フルマネージドでRedis、Memcachedをサポートしています
  - オープンソースのデータベース技術です。どのような特徴があるのか、後ほど詳しく比較していきます。
- キーバリューストアであるという点。データの保存についてです。
  - キーバリュー形式とは、JSON形式のように、ある一つのキーが決まることでもう一つの値が決まるようなデータ形式のことを言い、これをセットで保存するものをキーバリューストアと呼びます。ElastiCacheやDynamoDBは、キーバリューストアの AWSサービスです。
- インメモリ型データベースのため、性能が高い。非常に高速で低遅延で動作します。

    - インメモリとは、データをすべてメインメモリーに読み込み、ストレージを使わない方式を指します。
    - 一般的にはデータベースはデータをディスクや SSD に保存しますが、そうはなく、データストレージ用のメモリを使用しています。つまり、超高速です。
- ElastiCacheは、実態というか、中身は EC2インスタンスを使用しています。
    - そのため、使用の際にはインスタンスタイプとファミリーを指定して使用します。



ElastiCache Redis /Memcached 比較

CloudTech

-

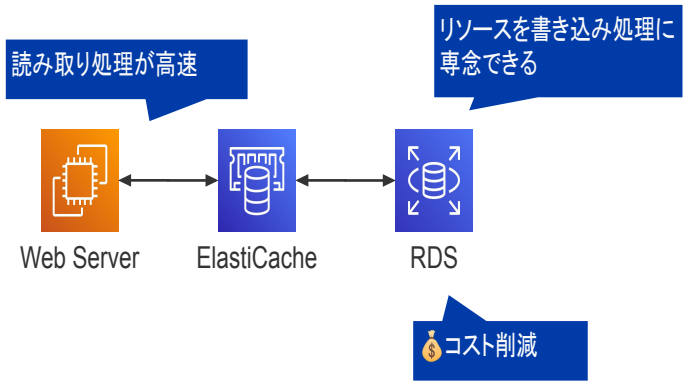
Redis / Memcached 比較


項目	Memcached	Redis
DBの負荷をオフロードするシンプルなキャッシュ	はい	はい
マルチスレッドパフォーマンス	はい	いいえ
高度なデータタイプ	いいえ	はい
データセットの並べ替え / ランキング	いいえ	はい
Pub / Sub キャパシティー	いいえ	はい
自動フェイルオーバー機能を備えたマルチAZ	いいえ	はい
永続性	いいえ	はい

CloudTech

オープンソースのデータベース技術である MemcachedとRedisの2つのタイプがあります。
それぞれの特徴を表した図を紹介します。試験でも問われます。
まずは、共通する点「DBの負荷をオフロードする」を見ていきますが、
そもそものデータベースの補助的な役割としてキャッシュを使うと、どんないいことがあるのか？を詳しくみていきます。

ElastiCache（キャッシング：DBの負荷をオフロード）



CloudTech

キャッシュが無い例として、EC2とRDSで構成されたシステムを考えてみます。
データが頻繁に更新されるようなシステムの場合、データベースに対する読み込みと書き込みの負荷は非常に高くなります。
データベースのスペックを拡張したとしても、レイテンシー（遅延）やコストの問題が発生します。

ここで、
EC2とRDSとの間に、インメモリデータベースを配置し、データを一旦ElastiCacheに格納することにより、
DBの負荷軽減とともにコストや読み取り遅延の問題を解決する考え方です。

メリットは複数あります。例えば

- RDSは読み込みトラフィックをキャッシュに任せることができるので、RDS自体は書き込みリクエストに専念できます。

    - （いわゆる負荷を肩代わりさせることをオフロードと言います）
- ECacheはインメモリデータベースなので、RDSよりも迅速に応答するため、処理が高速となること。
- コスト削減（一般的にRDSのスペックを巨大にするよりは費用対効果が高いと言われてます。）

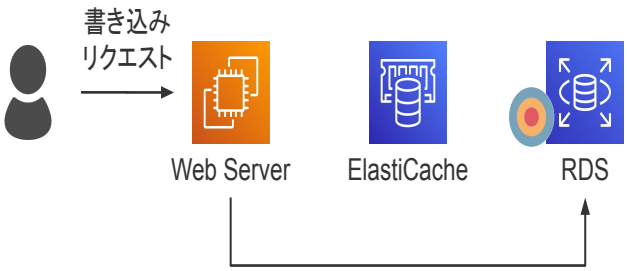
続いて、SAAの試験範囲外とはなりますが、データをどうキャッシュするのかについては2つの方式があります。
少し踏み込んでみましょう。

キャッシュ戦略
レイジーロード / ライトスルー

CloudTech

-

ElastiCache（キャッシング：DBの負荷をオフロード）

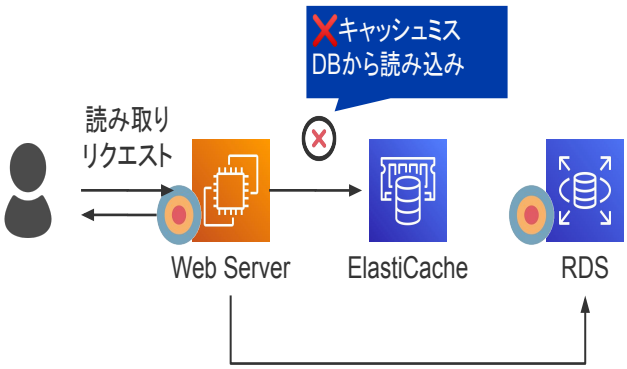


CloudTech

スタンダードなキャッシュの仕組みを図解していきます。

- まずユーザーから書き込みリクエストがきて
- WebServerのアプリケーションは RDSにデータを書き込みます。

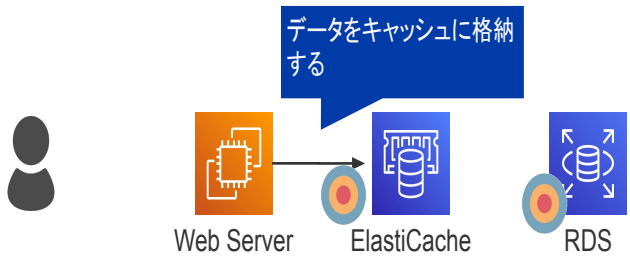
ElastiCache（キャッシング：DBの負荷をオフロード）



CloudTech

- その後、ユーザーからデータを参照する為、WebServerに読み取りリクエストがきたとします。
- WebServerはRDSではなく、まずはキャッシュであるElastiCacheにデータがあるかどうか問い合わせます。
    - しかし、キャッシュにデータを保持しないことが分かりました。（キャッシュミスと言います）
- キャッシュにデータがないので、WebServerはRDSに直接データを読み取りに行き、
    - 読み取ったデータをユーザーに応答します。

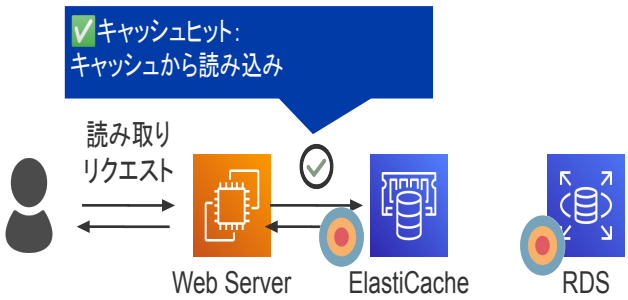
ElastiCache（キャッシング：DBの負荷をオフロード）



CloudTech

- その後、WebServerは読み取ったデータをキャッシュに格納します。（ちょっと遅れてキャッシュに格納する動き）
  - RDSに元のデータがありますし、ElastiCacheにはキャッシュとして同じデータが格納されています。

ElastiCache（キャッシング：DBの負荷をオフロード）



Lazy…のんびり
load…格納
Lazy Loading（レイジーローディング）

CloudTech

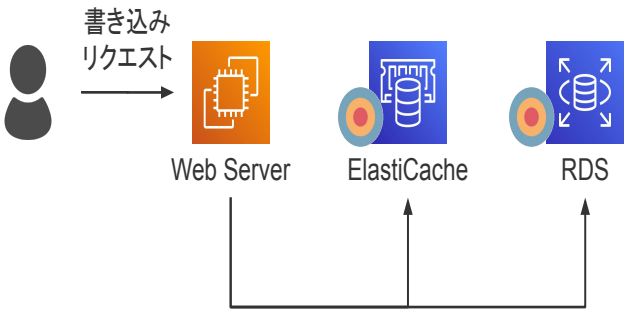
- その後、ユーザーからWebServerに再び同じデータの読み取りリクエストがあれば、
  - WebServerはまず、ElastiCacheにデータがあるかどうか問い合わせます
  - データがElastiCacheにキャッシュとして格納されていました。（これをキャッシュヒットと呼びます）
  - WebServerはこれを取り出し、ユーザーに応答します。

- これ以降も同様の動きです。
  - 正確にはキャッシュされたデータには TTL（time to live）というものがあり、この有効期限が切れるまではキャッシュデータを取り扱います。

- まとめると、まずWebServerはキャッシュにあるかないかを問い合わせ、あればキャッシュヒットして成功、
  - なければRDSからデータをとりだし、ちょっと遅れて

- キャッシュに格納する動きです。
  - これをlazy loadingと呼びます。
    - 英単語のlazyは「のんびり」
    - loadは「格納」という意味、マシンガンにリロードしたり、ゲームのNow loadingのアレですね。
      - lazy loadingと呼ばれます。直訳すると、のんびり格納ですね。

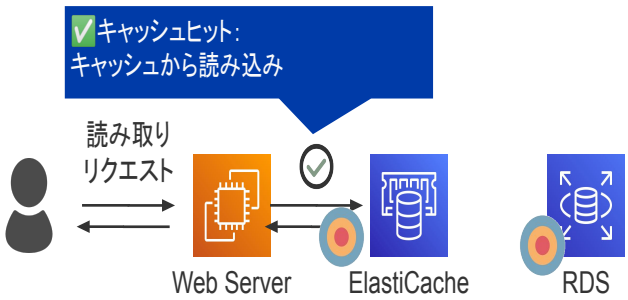
ElastiCache（ライトスルー）



CloudTech

- では、ライトスルー戦略を見ていきましょう。
- まず、WebServerは、ユーザーから書き込みリクエストを受け取ったら、データをRDSに書き込む。ここまでは先ほどのLazy Loadingと同様ですね。
  - それと同時に、ElastiCacheにも書き込みます。
- ライトスルー、同時書き込みのイメージですね。
  - なお、ライトスルーという用語は、AWSだけでなく、CPUとキャッシュメモリの文脈でも使われる一般的な用語なのでイメージをつけておきましょう。

ElastiCache（ライトスルー）



- **WebServerの負荷が高い**
- **キャッシュのサイズが小さいとうまく動作しない**

CloudTech

- その後、ユーザーからWebServerに再び同じデータの読み取りリクエストがあれば、
  - WebServerはまず、ElastiCacheにデータがあるかどうか問い合わせます
  - データがElastiCacheにキャッシュとして格納されていました。（これをキャッシュヒットと呼びます）
  - WebServerはこれを取り出し、ユーザーに応答します。

ライトスルーはメリットもある一方、問題点としては 2つですね。

- WebServerの負荷が高いこと。両方書き込まないといけませんので。
- 次に、キャッシュのサイズが小さいとうまく動作しないということ。

これら、どのようにキャッシュデータを取り扱うかは、紹介した通

り、2つの戦略、Lazy Loading戦略とライトスルー戦略があります。参考にしてみてください。

Amazon ElastiCache (Memcached)

CloudTech

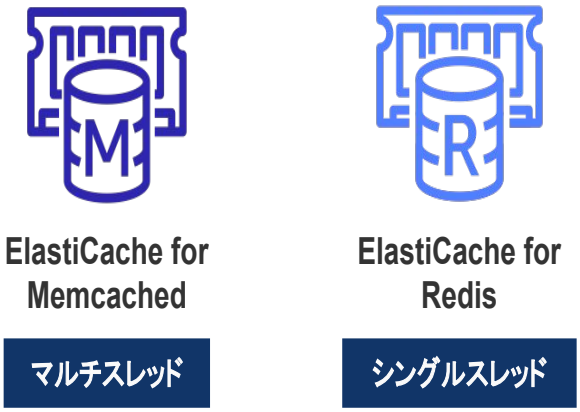
Redis / Memcached 比較

項目	Memcached	Redis
DBの負荷をオフロードするシンプルなキャッシュ	はい	はい
マルチスレッドパフォーマンス	はい	いいえ
高度なデータタイプ	いいえ	はい
データセットの並べ替え / ランキング	いいえ	はい
Pub / Sub キャパシティー	いいえ	はい
自動フェイルオーバー機能を備えたマルチAZ	いいえ	はい
永続性	いいえ	はい

CloudTech

ここまで、DBの負荷をオフロードする特徴について見てきました。
では、memcachedの特徴について、マルチスレッドパフォーマンスについて。

ElastiCache for Memcached



CloudTech

Memcachedはマルチスレッドで動作します。
マルチスレッドとは、並列処理のことですね。
CPUを並列で使用できるというイメージです。
Memcached はマルチスレッドであるため、複数の CPUコアを並列使用できます。

一方の、Redisはマルチスレッドではありません。
Redisはシングルスレッドで動作するので、CPUのコア数を上げてもMemcachedほどはパフォーマンスの向上を期待することはできません。
その分、複雑なことができます。ではここからは Redisの特徴について見ていきましょう。

Amazon ElastiCache (Redis)

CloudTech

Redis / Memcached 比較

項目	Memcached	Redis
DBの負荷をオフロードするシンプルなキャッシュ	はい	はい
マルチスレッドパフォーマンス	はい	いいえ
高度なデータタイプ	いいえ	はい
データセットの並べ替え / ランキング	いいえ	はい
Pub / Sub キャパシティー	いいえ	はい
自動フェイルオーバー機能を備えたマルチAZ	いいえ	はい
永続性	いいえ	はい

CloudTech

ElastiCache for Redis

➤ 高度なデータタイプ
Stringに加えてさまざまなデータタイプをサポート

➤ データセットの並べ替え / ランキング
Leader board（Redis）

➤ Pub / Sub メッセージングをサポート
チャットアプリケーション

CloudTech

高度なデータタイプ
Redis は、String型 に加えて、List、Set、Hash、など、複雑なデータ形式もサポートしています。
先ほどのMemcachedはシンプルなデータ構造のみ使用可能ですが、より複雑な場合のデータ構造を使う場合には Redisを選択すると良いでしょう。

Redisの柔軟なデータ構造を使用して、例えばよく挙げられるユースケースが、LeaderBoardです。
LeaderBoardとはゲームのハイスコアのトップ 10 など各プレイヤーのスコアを表示したボードのことです。野球などのスコア表はスコアボードと呼ばれますよね。
リーダーボードは、大勢のゲームプレーヤーが同時にプレーしており、スコアが常に変化し、計算が複雑になります。そのようなユースケースにうってつけのサービスです。

Pub / Sub キャパシティー
公式ドキュメントでは、Redis は、チャットアプリケーションに利用される例が紹介されています。
チャットアプリケーションを構築するには、クライアントがチャットルームの他の参加者に再配信されるメッセージ通信チャネルが必要となり，一般に publish-subscribe パターン (PubSub) を使用して実装されます。

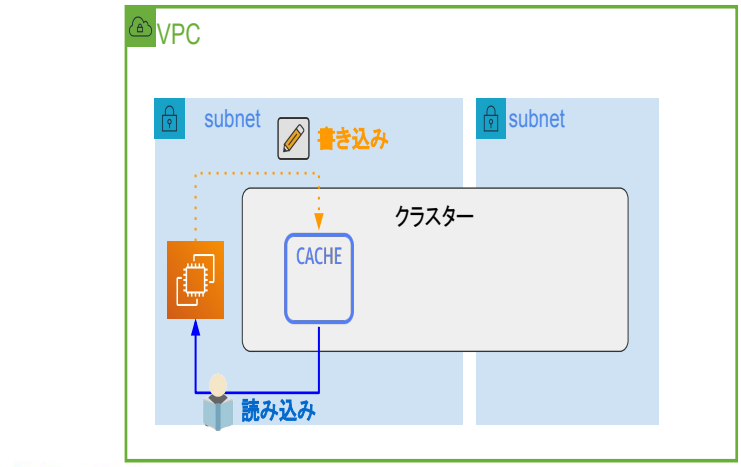
Redis / Memcached 比較

項目	Memcached	Redis
DBの負荷をオフロードするシンプルなキャッシュ	はい	はい
マルチスレッドパフォーマンス	はい	いいえ
高度なデータタイプ	いいえ	はい
データセットの並べ替え / ランキング	いいえ	はい
Pub / Sub キャパシティー	いいえ	はい
自動フェイルオーバー機能を備えたマルチAZ	いいえ	はい
永続性	いいえ	はい

CloudTech

ElastiCache（Redisデプロイモデルとレプリケーション）

シングルノード（単一のノード）



自動フェイルオーバー機能を備えたマルチ AZ　についてですが、その前提知識として、Redisには3つのデプロイモデルがあるのでそちらを説明します。
まずは、シングルノードクラスターです。

これは常にクラスターが
1つのノードで構成されるモデルです。ノードとはキャッシュ処理をする実態である、EC2インスタンスのことだと思ってください。
ご覧のように、CACHEサーバーというアイコンでノードを表現しています。

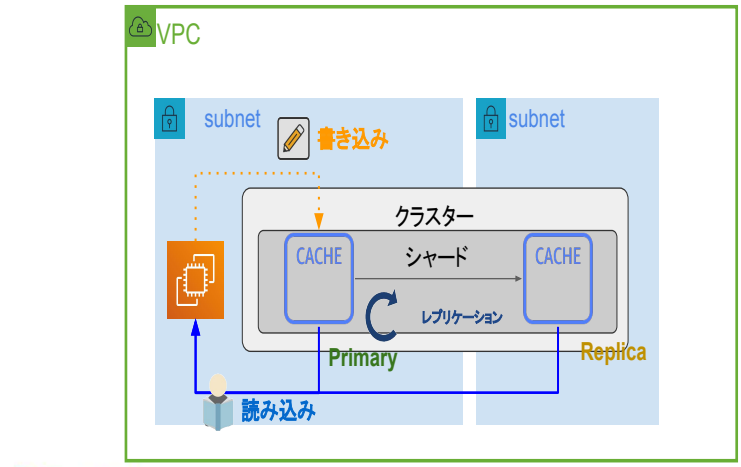
このノードに対し、クライアント（この例では EC2インスタンスのアイコンを用いていますが）が
読み込みや書き込みを行います。
シングルノードクラスターでは、後続で説明しますが、ノードを束ねたシャードの利用はできません。
ノードは常に単一のサブネットで実行されているため、マルチ AZ

構成はとれませんが、データ自体のバックアップやリストアはサポートされています。
負荷が高まった際の、スケールに関しては、このひとつのノードタイプのスペックを変更する垂直スケーリングに対応しています。

シングルノードは、システムにとっての単一障害点となるので、クリティカルな使用用途には避けましょう。

ElastiCache（Redisデプロイモデルとレプリケーション）

クラスターモードが無効（単一シャード）



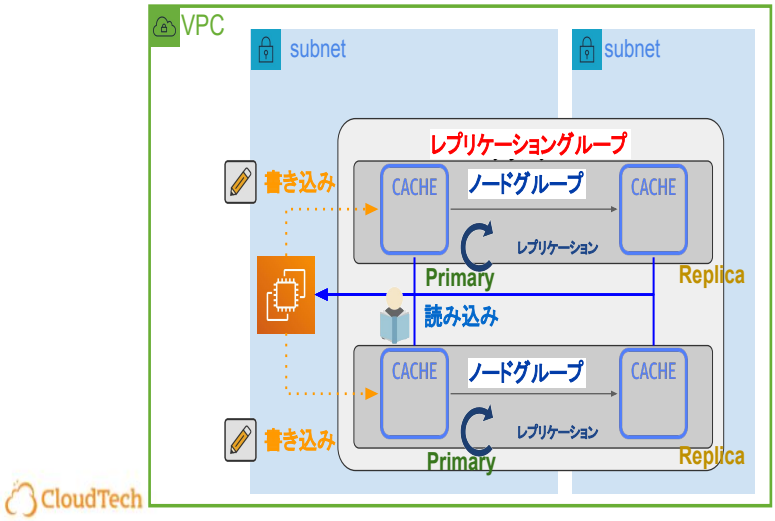
2つめのでプロモデルが、「クラスターモード無効」というデプロイモデルです。ちょっと分かりにくいので、単一のシャードと補足させてもらいました。

- シャードというものが登場してきましたね。
  - シャードとは、1つのプライマリのノードと、最大 5つのレプリカノードで構成される処理単位の名称です。
    - この例ではプライマリーノード 1つ、レプリカノード1つで記載しています。最大 5つまで増やせます。
    - プライマリからレプリカへは、データがレプリケーション（同期）されています。
  - クライアントであるEC2は、プライマリに対しての書き込みを行い、
    - プライマリとレプリカに対しての読み込みを行います

このプライマリとレプリカを異なる AZに配置して、マルチ AZ配置とすることができます。
AZ障害時にはレプリカがプライマリーに昇格させることもできます。
ちょうどRDSのリードレプリカがプライマリーに昇格するような感じです。
スケールに関しては、ノードタイプのスペックを変更する垂直スケーリングに対応しています。

ElastiCache（Redisデプロイモデルとレプリケーション）

クラスターモードが有効（複数シャード）



最後、3つめが「クラスターモードが有効」のデプロイモデルです。複数シャードと補足しました。

- 複数のシャードで処理を分担して、読み取り /書き込みの性能を上げることができます
  - クライアントであるEC2は、それぞれのシャードに所属するプライマリに書き込みを行います。
  - 均等に振り分けた場合、各プライマリは 50%ずつの処理負荷になります。
    - さらにそれぞれのプライマリ、及びレプリカから読み込みを行います。
- このデプロイモデルではデフォルトでマルチ AZ が有効になります。
- シャード単位で増やしてスケーリングができます。
  - 先ほどの2つのデプロイモデルではノードのスペックを上げる垂直スケーリングでしたが、
  - こちらは水平スケーリングですね。

その他補足事項です。ElastiCacheのGUI画面ではクラスタ／シャードという用語を使っていますが、
SDK、AWS CLI、CloudFormationでは、レプリケーショングループ / ノードグループという異なる用語が使われています。若干混乱してしまいますのでご注意ください。

最後、永続性について軽く触れておきます。
デフォルトでは、ElastiCache の Redis ノード内のデータはメモリにのみ存在し、永続的ではありません。
ノードが再起動されるか、基になる物理サーバーでハードウェア障害が発生した場合、キャッシュ内のデータは失われますが、
データの耐久性が必要な場合、Redis の AOF 機能を有効にすることができます。

キャッシュの仕組みは、そもそもデータに TTLを付与しておいて、データを永続させないというコンセプトもあります。
後で必要になるような、バックアップが必要になるようなデータはそもそもキャッシュに保存はしない方が良いというのが大前提にありますね。

Amazon ElastiCache for Redis でクラスターモードを使用する | Amazon Web Services ブログ

Redis / Memcached 比較


項目	Memcached	Redis
DBの負荷をオフロードするシンプルなキャッシュ	はい	はい
マルチスレッドパフォーマンス	はい	いいえ
高度なデータタイプ	いいえ	はい
データセットの並べ替え / ランキング	いいえ	はい
Pub / Sub キャパシティー	いいえ	はい
自動フェイルオーバー機能を備えたマルチAZ	いいえ	はい
永続性	いいえ	はい

CloudTech

まとめると、redisの方が機能的には充実しています。

Amazon ElastiCache（ユースケース）

CloudTech

-

ElastiCache（ユースケースまとめ）

ユースケース
➤ データベースのキャッシュ
➤ Webのセッション管理
➤ Leader board（Redis）
➤ メディアストリーミング

CloudTech

- 使用用途としては、アプリケーションのセッション情報や、データベースクエリ結果のキャッシュするなどのユースケースで使用されます。
  - データベースのキャッシュとして
    - アプリケーションとデータベースパフォーマンスを高速化するキャッシング
  - Webのセッション管理
    - APサーバとセッションの依存関係が無くなるため以下のようなメリットが得られます。
    - EC2障害発生時に他のEC2にセッション情報を引き継いで処理が継続できる
    - 特定のAPサーバを意識する必要がなくなるのでELBの振り分けの自由度が上がる
    - 特定のAPサーバを意識する必要がなくなるのでスケールインが何時でも可能
      - ALBのスティッキーセッション機能でも可能だが、片方のEC2に偏るなどのリスク

            - もある。ここは何が正解かではなく、費用対効果のバランスを考えるべきです。
    - Leader board
        - ゲームのランキング表のような使用用途に使えます。
    - メディアストリーミング
        - 数百万人のユーザーの認証情報を保管する場所など