モバイルゲームの全世界オンライン対戦を実現する方法を考察する

クルーズ株式会社

田沢 知志

CROOZって何やってる会社？

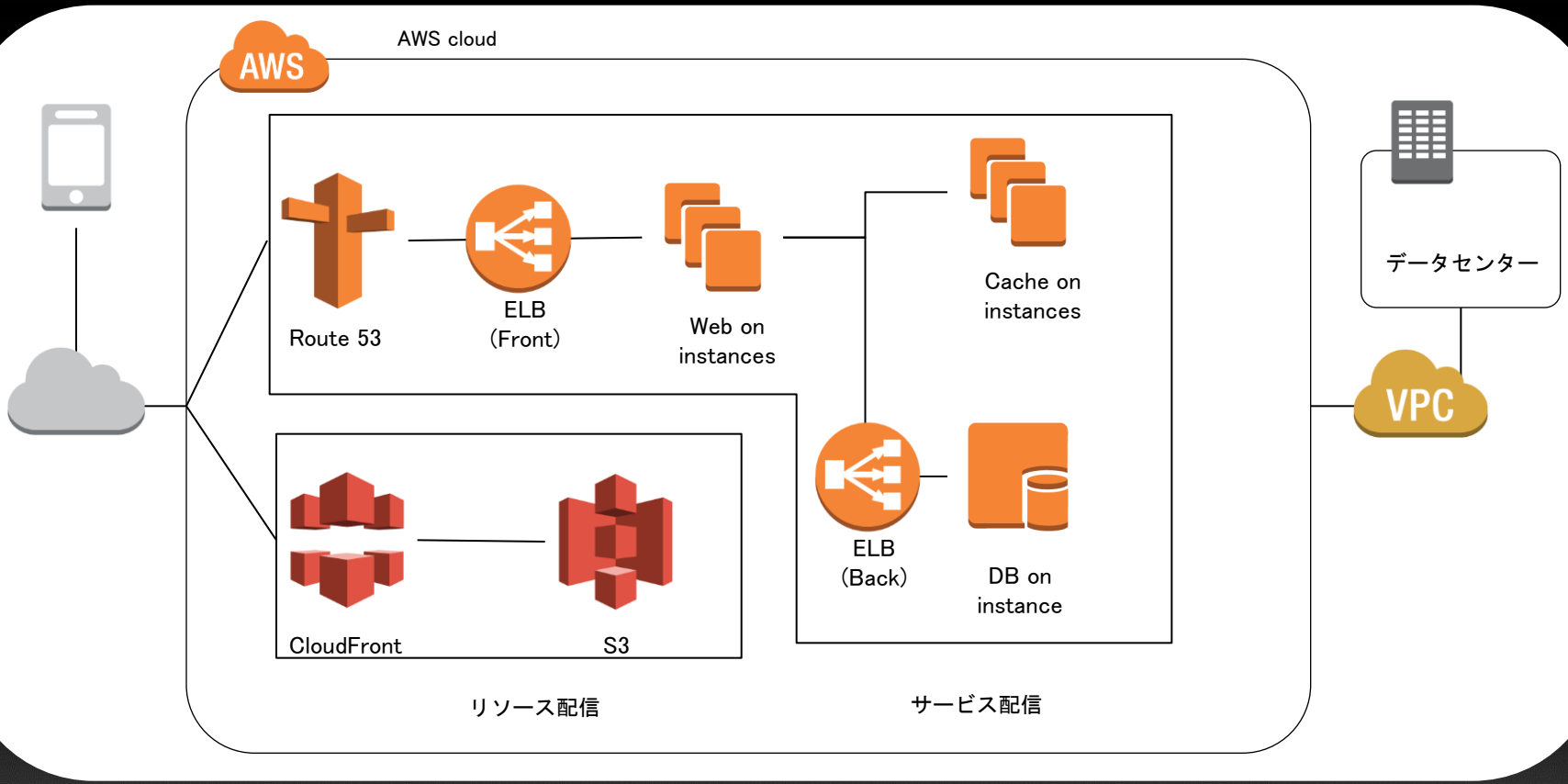
CROOZは、ソーシャルゲームやネット通販を中心に、世界中にインターネットサービスを提供するエンターテインメント企業です

アジェンダ

- クラウド導入の一般的な考慮点(LAMP環境)
- ストレージI/Oの考慮点
- オンラインゲーム 設計のステップアップ
- 最後に

クラウド導入の一般的な考慮点（LAMP環境）

■インフラ構成概要



■OS/Middleware概要

- OS:CentOS6.4
- Web:apache2.2系/PHP5.4系
  - 社内独自フレームワークVENUS使用
- Cache:redis2.8系
- DB:Percona5.5系

■インスタンスタイプ選択の考慮点

- インスタンスタイプは3-6か月単位で向上
  - 旧/新インスタンスを比較すると、コストパフォーマンスは3割以上良い(印象)
- インスタンスタイプは定期的に変更
  - 数か月前はm2/hi1タイプをメインで使用
  - 現在メインで使用してるタイプは・・
    - Web系：m3.xlarge、m3.2xlarge
    - Cache/DB系：r3.xlarge、r3.2xlarge

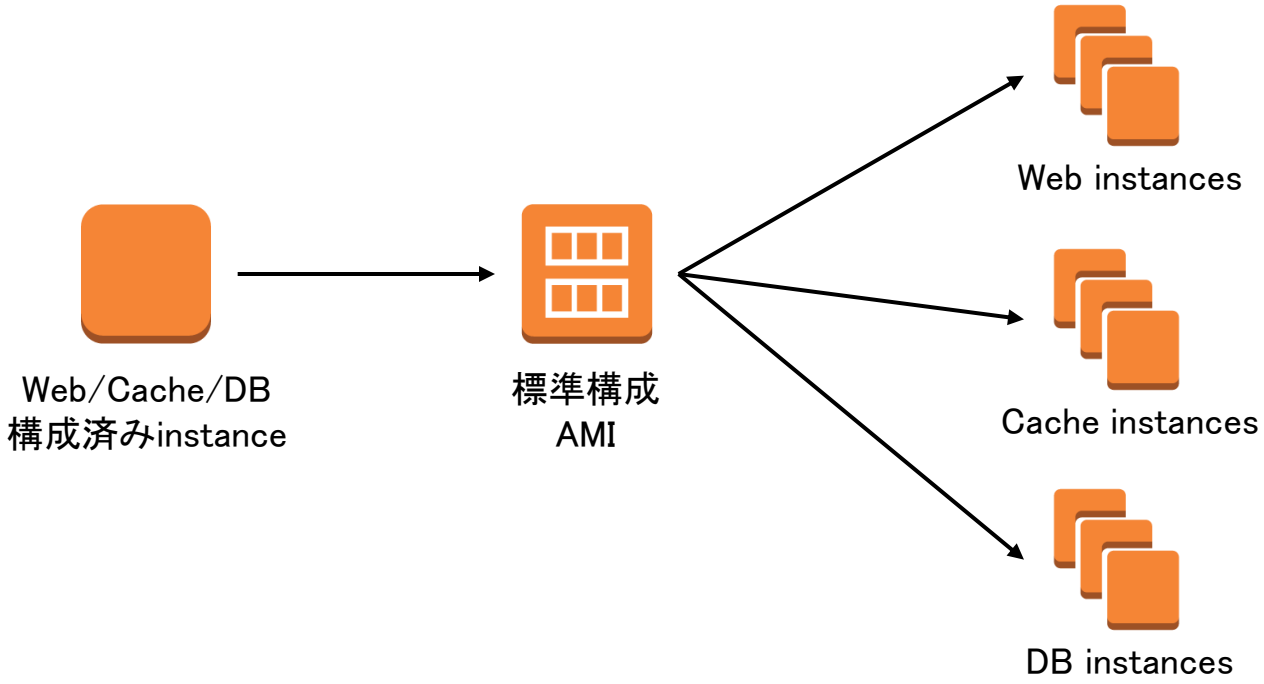
■ベンチマークの考慮点

- 複数リージョン、複数インスタンス毎に比較
- 重要指標
  - DB：ストレージ IOPS、queries/s
  - Cache：requests/s
  - Web：CPU Load Average、USER使用率

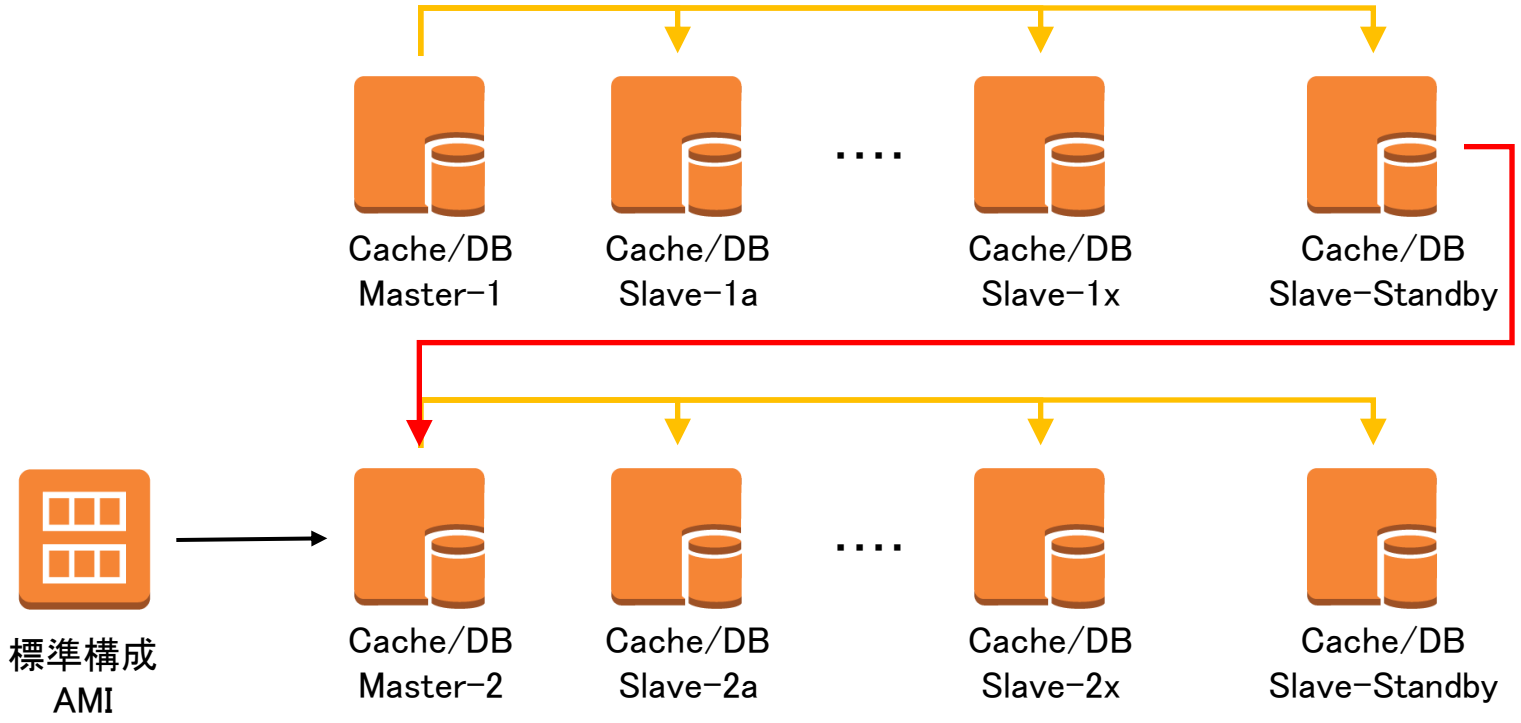
■スケーラビリティの考慮点

- Web/Cache/DB 基本的にhorizontal scaling
- Web
  - 構成済image(AMI)からインスタンス起動
- Cache/DB
  - sharding/partitioning
  - スタンバイ(バックアップ)インスタンスからデータをコピーして同期

■スケーラビリティの考慮点-インスタンス



■スケーラビリティの考慮点-データ同期



■キャパシティの考慮点

- Web：性能限界のポイント(弊社事例)
  - ボトルネックの要因は・・？
  - プログラムが酷くない限りCPU負荷はない
  - Cache/DB のレスポンス遅延
  - ローカルポート不足(デフォルト30000弱)
    - ip_local_port_range で 約50000まで拡張
    - tcp_max_tw_buckets で time_wait 数を調整
- Cache/DBは後半で・・

■コストの考慮点

- 1インスタンスあたりのMaxDAUを想定
  - 弊社参考例
  - Web(m3.2xlarge) 60,000DAU
  - DB(r3.2xlarge IOPS4K) 120,000DAU
  - Cache(r3.large) 180,000DAU
- 想定MaxDAUから必要インスタンス数を算出
  - 月額売上の？%をクラウドコスト目標に

■リソース(バイナリデータ)配信の考慮点

- リソース配信はCloudFrontを使用
  - 各リージョン毎にエッジロケーション
  - オリジンはS3に配置
  - Reports & Analytics 機能もあり
  - 数100TB/月の配信で利用
  - リザーブドプラン契約により3-4割安に

ストレージI/Oの考慮点

一般的なIOPS

SAS 15krpm	175 - 300 IOPS
Amazon EBS 	1,000 - 4,000 IOPS
SSD	10,000 - 15,000 IOPS
FusionIO ioDrive2	150,000 - 200,000 IOPS

クラウドストレージの特性を考慮すると・・

■DBのスケーラビリティ

- IOの弱点を考慮して・・
  - innodb_buffer_pool_sizeに乗るDBサイズ
    - オンメモリであれば、数Kiops程度
    - queries/sの方が限界に達する
  - 臨機応変にpartitioning / sharding
  - 参照はできるだけCacheへ

■DBのベンチマーク(弊社検証参考)

- Percona Server 5.5系
- sysbench
  - 10,000,000recods/ReadWrite/1thread/60秒

	read	write	transactions
r3.2xlarge (EBS iops2000)	192,612	55,032	13,758
hi.4xlarge (SSD Ins Vol)	144,032	41,152	10,288
SSD (ほぼ同スペック物理)	312,494	89,284	22,321

■Cacheのスケーラビリティ

- Cacheの注意点(弊社事例/r3.large)
  - IOPSが問題になることはほとんどない
  - redisベンチは 約500,000requests/s
    - redis-benchmark -r 1000000 -n 2000000 -q -P 16
  - メモリをフル活用するために＆保存時のレスポンス遅延をなくすために、ストレージ保存(BGSAVE)させない
    - 保存用スタンバイインスタンスを用意

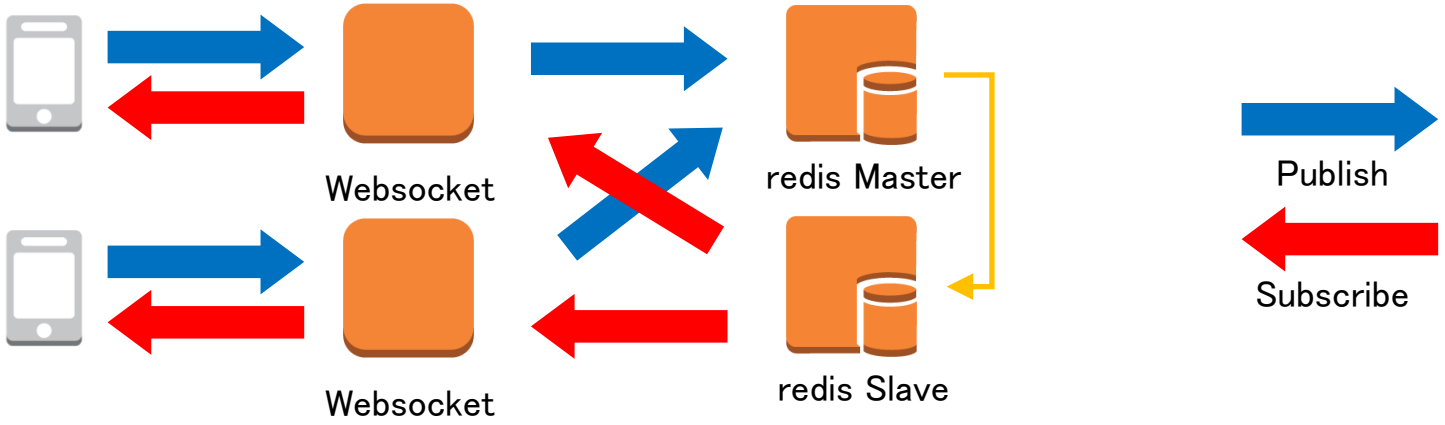
オンラインゲーム
設計のステップアップ

■全世界オンライン対戦の考慮点

- 通信レイテンシーを短縮するには？
  - 通信プロトコルの選択は？
  - リージョンの配置は？
  - リージョン間データの同期は？

■オンラインゲームのプロトコルは？

- 現時点ではWebsocket
  - Java (GlassFish)を選択
  - redis の Pub/Subによりスケールアウト



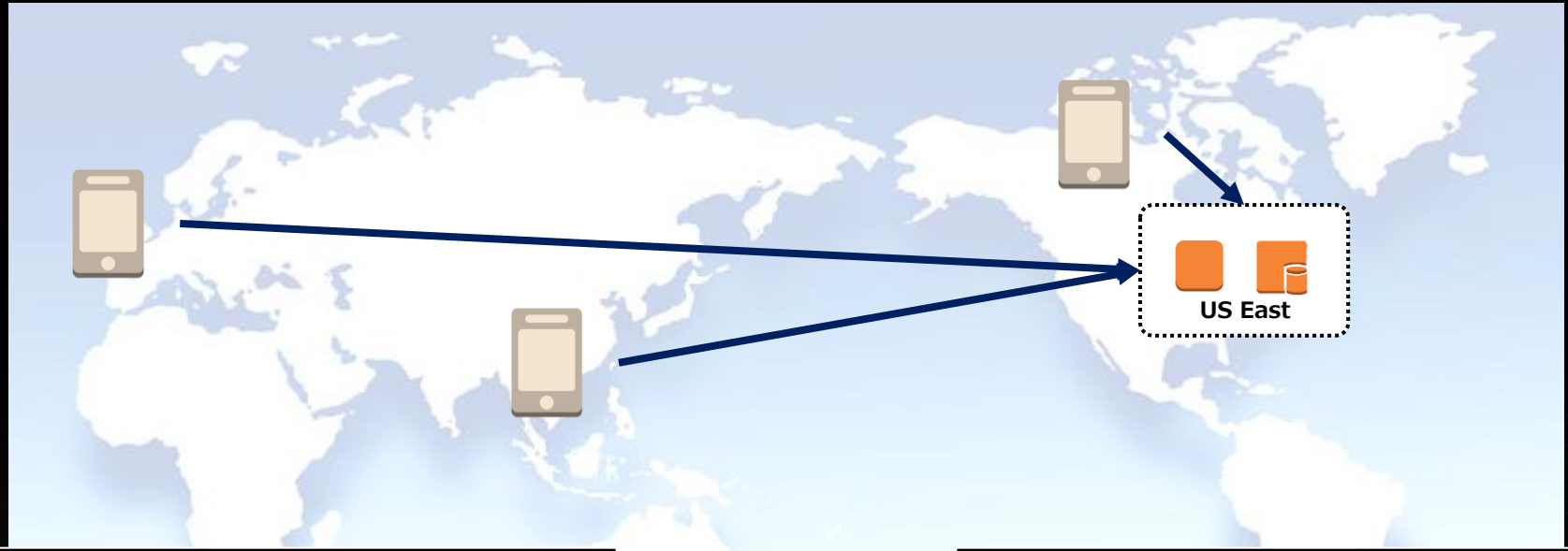
■オンラインゲームのプロトコルは？

- 今後は HTTP/2 も検証予定
  - 6/17現在 draft13
  - Server Push 機能を利用
  - 参考
    - http://tools.ietf.org/html/draft-ietf-httpbis-http2-13
    - https://github.com/http2/http2-spec/wiki/Implementations

■ Websocketのベンチマーク(弊社検証参考)

- インスタンスタイプ
  - Websocket/redis ともに m3.large
- ベンチ結果
  - Websocketサーバー1台あたり、3,000同時接続
  - レスポンスタイムはMax2.5秒

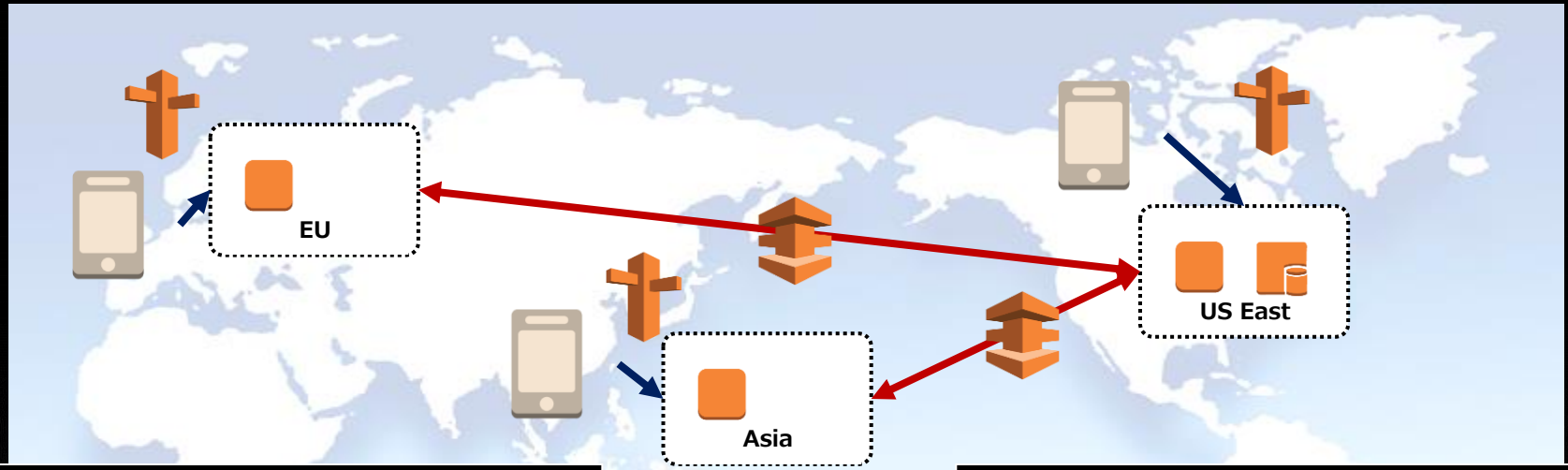
■ 1st Step USリージョンのみ



全世界からUSリージョンの
Application(Websocket)
サーバーへアクセス
※Latency 1.5s-3s

USリージョンのみに
・Application(Websocket)
・Cache/DB
を配置

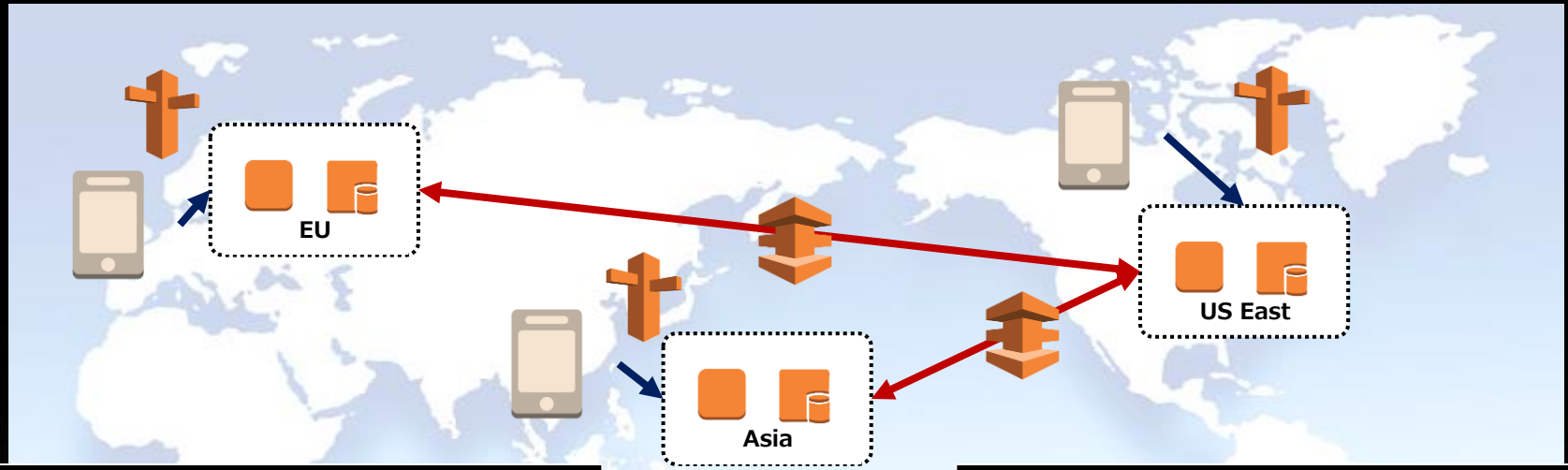
■2nd Step 各リージョンにedgeサーバー



Route53により最短の
edgeサーバーへアクセス
edge⇔Appはリージョン間
Direct Connectで接続
※Latency 500ms-1.5s

USリージョンに
・Application(Websocket)
・Cache/DB
各世界リージョンに
・edgeサーバー
を配置

■ 3rd Step 各リージョンにDBノード



Route53により最短の
edgeサーバーへアクセス
edge⇔Appはリージョン間
Direct Connectで接続
※Latency 500ms-1s

各世界リージョンに
・Application(Websocket)
・Cache/DB
・edgeサーバー
を配置
Cache/DBも同期

最後に

■クラウドの魅力はまだまだたくさん！

- 性能、機能は日々進化
  - 新インスタンス/ストレージ性能/etc…
  - PaaS機能の充実
    - 今後はBaaS系機能も取り入れたい
- 新しいクラウド デザイン パターンの可能性
- ヒット予測の難しいモバイルアプリ配信に対して柔軟な拡張が可能

■今後AWSに期待したいことは・・・

- 無停止でのインスタンスタイプ変更
- RDS関連
  - PerconaやMariaDBサポート
  - クラスタ化
  - ELBサポート
- 親アカウントからダイレクトに子アカウント操作ができる
- リージョン間 Direct Connect無償化