Fate/Grand Orderにおける 大規模なデータベース移行と負荷試験

2019年6月14日

今井 守生

DELiGHT WORKS Inc.

はじめに

DELiGHT WORKS Inc.

- **資料は公開予定です**
- **本セッションでは以下の内容をお話します**
  - AWS Database Migration Serviceを使った大規模なDB移行事例（垂直分割→水平分割）
  - 移行後に想定トラフィックを捌けることを検証するための負荷試験手法
- **弊社エンジニアが過去に2回、DB移行を主なトピックとした講演を行っていますのでこちらもご参照ください**
  - これで怖くない！？大規模環境で体験するDB負荷対策～垂直から水平の彼方へ～ /甲

    https://www.slideshare.net/hideakikabuto/db-108716671
  - 想定外な規模へと成長し続けるサービスを支えるサーバ開発・運用の軌跡

    ~ DBの縦横分割スケーリング ~ /髙屋

    https://speakerdeck.com/walkure/history-of-a-large-scale-web-api-service-development-and-operation-with-vertical-and-horizontal-database-partitioning

自己紹介

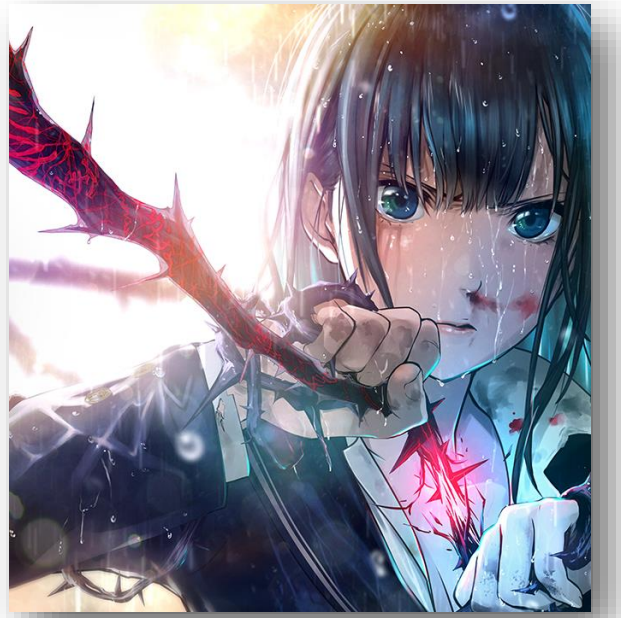
- 今井 守生（Morio IMAI）
- ディライトワークス株式会社に2014年2月入社
- Fate/Grand Orderの初期からサーバサイドの開発を担当

- 現在はいくつかのプロジェクトのサーバサイドを見ています

会社紹介

- 社名　　：ディライトワークス株式会社
- 設立　　：2014年1月22日
- 従業員数　：245名（単体・2018年11月時点）

Fate/Grand Order



新ゲームプロジェクト

アジェンダ

1. Fate/Grand Orderについて
2. DB移行の経緯
3. DB移行の方法
4. 負荷試験の準備
5. 負荷試験の実施
6. 移行リハーサルと段取り
7. 移行結果

アジェンダ

DELiGHT WORKS Inc.

1. Fate/Grand Orderについて
2. DB移行の経緯
3. DB移行の方法
4. 負荷試験の準備
5. 負荷試験の実施
6. 移行リハーサルと段取り
7. 移行結果

Fate/Grand Order（FGO）について

- 制作：TYPE-MOON / FGO PROJECT
  ※ノーツ、アニプレックス、ディライトワークス、
  3社の製作委員会

- 総監督：奈須きのこ（TYPE-MOON）
- 販売：アニプレックス
- 開発・運営：DELiGHTWORKS

Webサービスとしての概要（1/2）

- **APIサーバ**
  - Amazon EC2
- **DB**
  - Amazon Aurora
- **KVS**
  - Amazon ElastiCache（Redis, Memcached）

- **サーバフレームワーク**
  - ASP.NET MVC 4（C#）, Windows Server + IIS
- **クライアントアプリ**
  - Unity（C#）

Webサービスとしての概要（2/2）

DELiGHT WORKS Inc.

- **DBへのアクセスは、SELECTのみではなくUPDATE/INSERTもかなり多いです**

- **クライアントアプリもユーザの状態を持っています**
  - ログイン時にユーザデータをサーバからクライアントに連携し、
    それ以降は更新があったデータのみをクライアントに連携しています

アジェンダ

1. Fate/Grand Orderについて
2. **DB移行の経緯**
3. DB移行の方法
4. 負荷試験の準備
5. 負荷試験の実施
6. 移行リハーサルと段取り
7. 移行結果

稼働中サービスのアクセス障害例

- 1. ユーザ数の増加や広告による同時アクセス数の増加
- 2. 新機能・新規APIの追加
- 3. DBのデータ量の増加
  - DBのメモリにインデックスが乗らなくなると、Latencyが急激に劣化する

今回は3番目のケースについてのお話です

サービスへの影響

DBのデータ量増加の影響は、以下のように進行していくと予想できます

- **Phase 1**
  - 注目度が高いゲーム内イベントの開始直後、終了直前にAPIサーバのLatencyが上がる
- **Phase 2**
  - アクセスが多い時間帯にAPIサーバのLatencyが上がる
- **Phase 3**
  - APIサーバのLatencyが高い状態が継続する

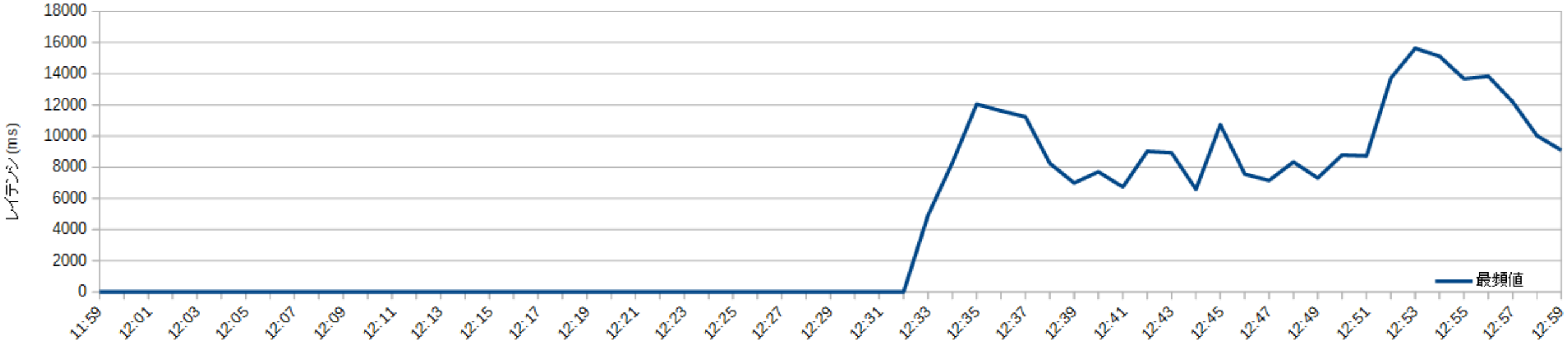
※進行ペースは、データ量が増加するペース次第

FGOでは（1/2）

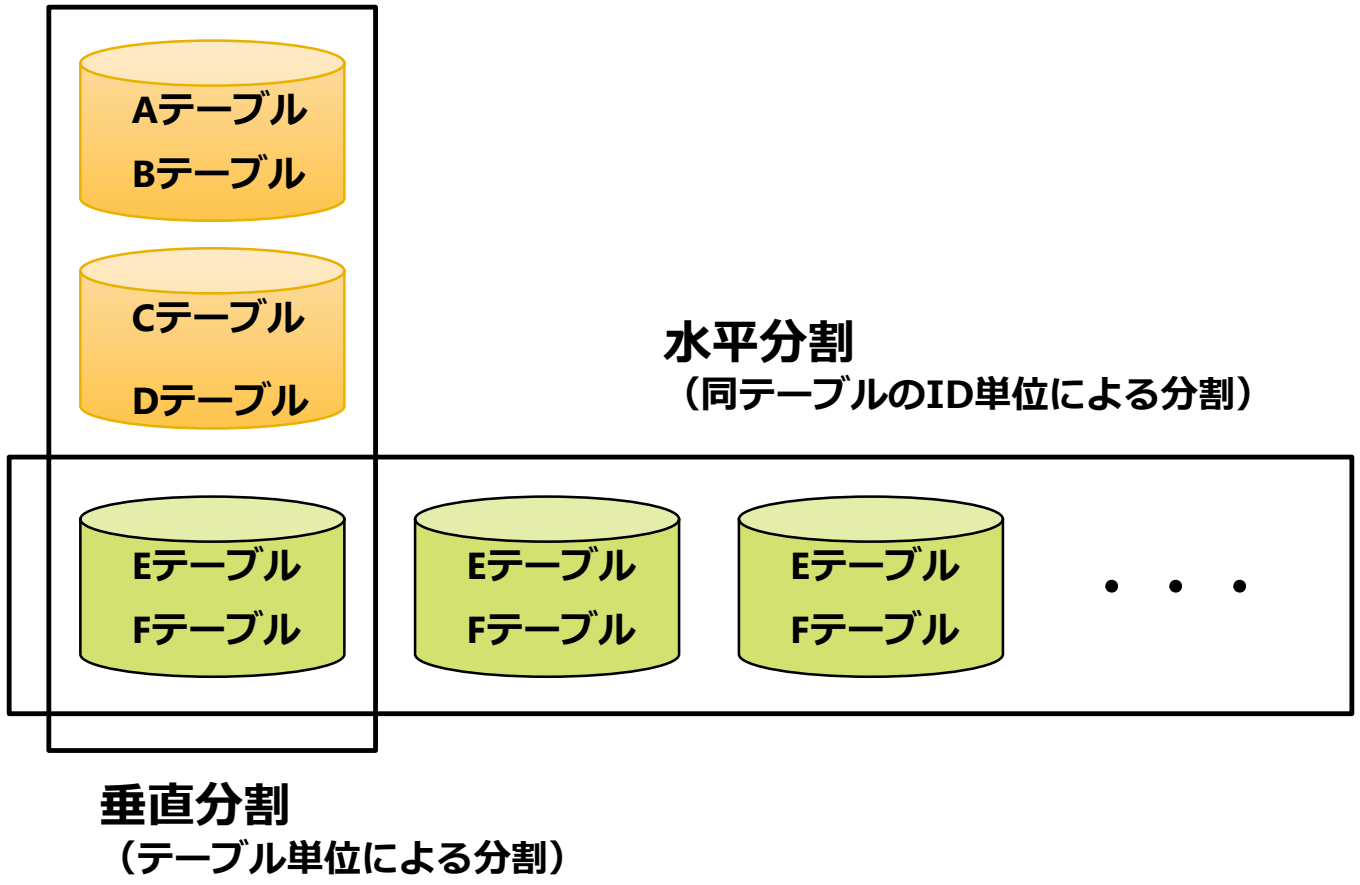
- **2015年7月のロンチ以降、想定外の速さでユーザデータ量が増え続けました**
- **垂直分割でその場を凌いでいたものの、垂直分割をやり尽くした状態で、2018年1月、前述のPhase 1まで進行しました**
  - この時点で最も大きいテーブルが、約30億レコードで450GBほど
- **特にガチャとボックスガチャの負荷が高く、以下のイベントで影響が出ていました**
  - 2018年正月のガチャ
  - 2018年1月のボックスガチャイベント

FGOでは（2/2）

2018年1月のボックスガチャイベント終了直前のAPIサーバのLatency

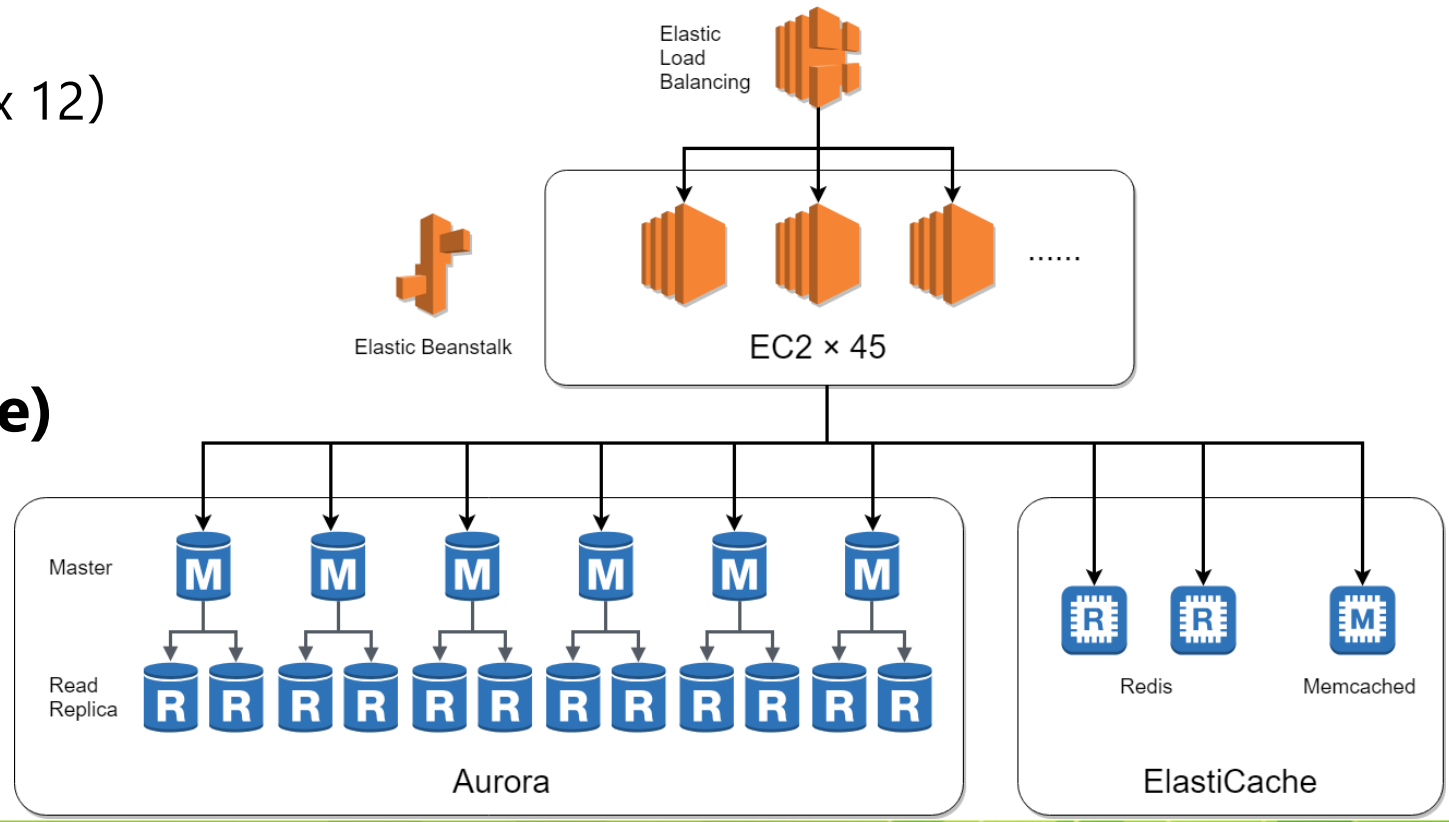


そもそもDB分割とは



サーバ構成（2018年1月時点）

- **IIS 8.5 (Amazon EC2 Windows Server 2012 R2)**
  - c4.8xlarge x 45
- **MySQL 5.6 (Amazon Aurora)**
  - db.r4.16xlarge x 18（master x 6, slave x 12）
- **Redis 2.8 (Amazon ElastiCache)**
  - cache.r3.4xlarge x 2
- **Memcached (Amazon ElastiCache)**
  - cache.r3.2xlarge



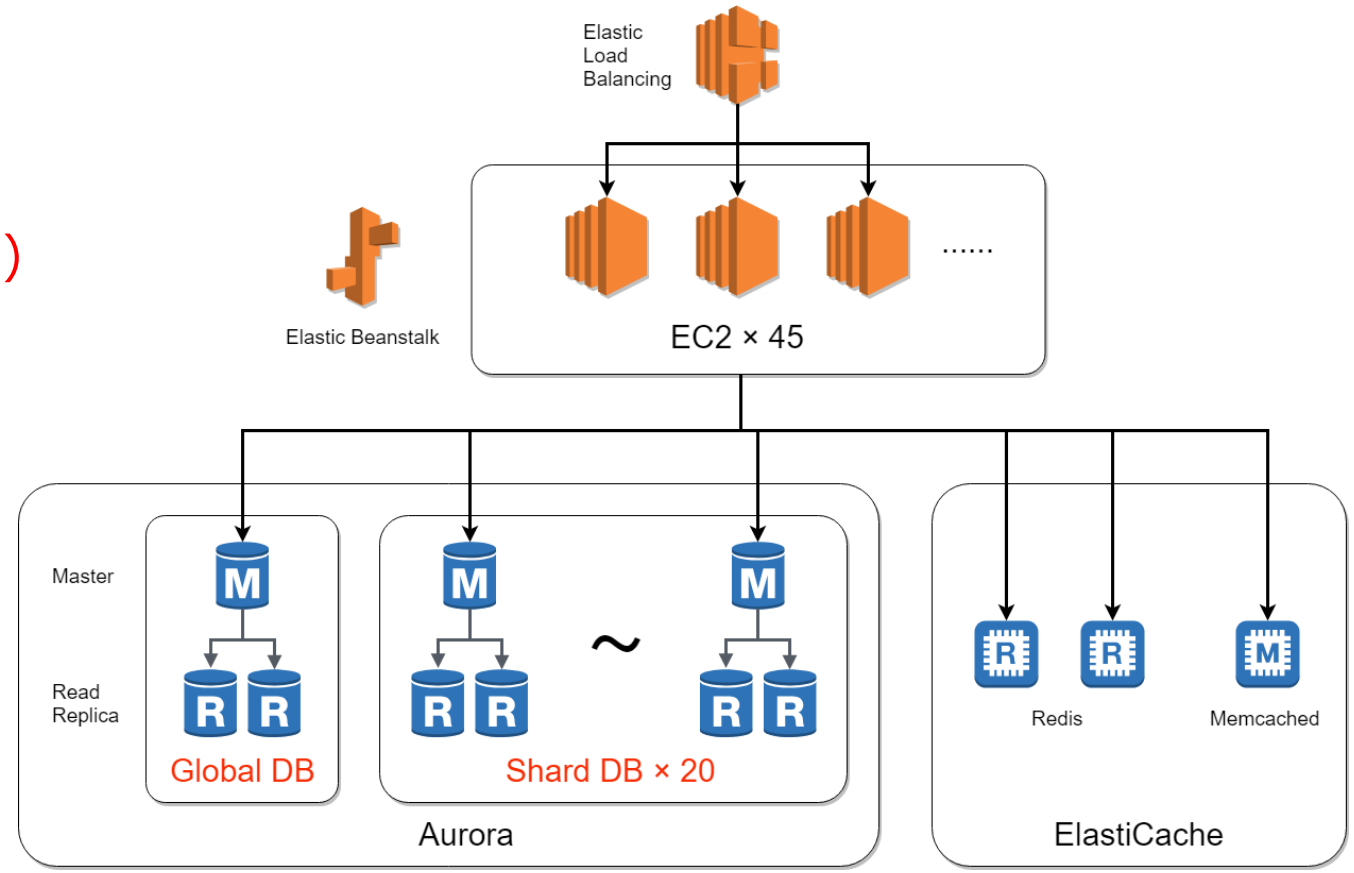
垂直分割構成の問題点

- 最大のインスタンスタイプ以上にスケールできない
- 1アクセスにつき、最大で、DBの台数分のDBアクセスが発生する
- 部分的にCOMMITが失敗した場合の復旧が辛い

これらの問題を解決するために全面的な水平分割への移行を決めました

サーバ構成（水平分割移行後の想定）

- **IIS 8.5 (Amazon EC2 Windows Server 2012 R2)**
  - c4.8xlarge x 45
- **MySQL 5.6 (Amazon Aurora)**
  - db.r4.16xlarge x 3（master x 1, slave x 2）
  - db.r4.4xlarge x 60（master x 20, slave x 40）
- **Redis 2.8 (Amazon ElastiCache)**
  - cache.r3.4xlarge x 2
- **Memcached (Amazon ElastiCache)**
  - cache.r3.2xlarge



アジェンダ

1. Fate/Grand Orderについて
2. DB移行の経緯
3. **DB移行の方法**
4. 負荷試験の準備
5. 負荷試験の実施
6. 移行リハーサルと段取り
7. 移行結果

垂直分割から水平分割への移行の要点

- **ユーザIDをキー（ユーザID % 20）として分割する**
  - 1ユーザの更新が1つのトランザクションに乗るため
- **AWS Database Migration Serviceでデータを移行する**
- **移行前の環境へ切り戻せるように、数日間は「ダブルライト」を行う**
  - 参照クエリは移行後DBを参照し、更新クエリは両方のDBを更新する方法
- **切り戻し用の環境も別で用意する**

AWS Database Migration Service（AWS DMS）とは

最小限のダウンタイムでDBのデータ移行ができるサービス

- DMSを動作させるレプリケーションサーバが必要
  - 移行元DB（ソースデータストア）、移行先DB（ターゲットデータストア）とは別のサーバ
- 以下のフェーズで構成される
  - 既存のデータの全ロード（FullLoad）
  - キャッシュされた変更の適用
  - 継続的なレプリケーション（CDC）

AWS DMSの留意点

Aurora リードレプリカをソースにできない

- 本番のmaster DBをソースとするのは怖いため、
  本番のmaster DBからMySQL レプリケーションした環境をソースに指定

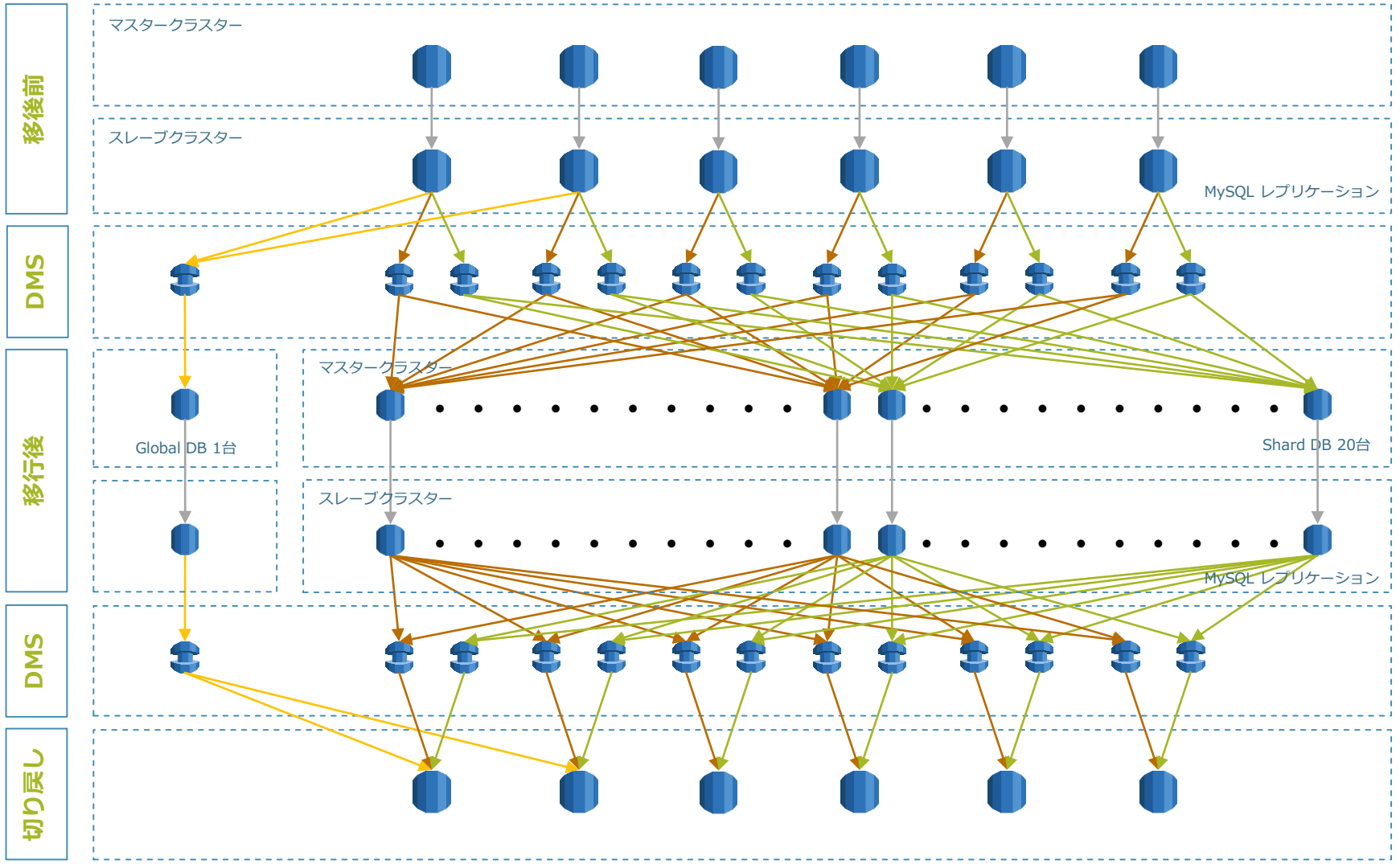
ソースフィルタ機能は関数をサポートしていない

- 「ユーザID % 20」の結果をもとにターゲットデータストアを選択することができない
- ターゲットとなるShard IDをレコード単位であらかじめ持たせておく必要がある
  - FastDDLで専用カラムを追加し、本番DB稼働中に10日間ほどかけて値を設定
  - パーティションがあるテーブルではFastDDLできないため、
    （パーティションがあるのは履歴テーブルだったので）データをすべて退避してからカラム追加

↓追加

user_id	……	horizontal_id
10010	……	10

移行時のDB構成



スケジュール

完了目標は、2018年7月末のFGO3周年キャンペーンまで

- 多くのアクセスが見込まれるキャンペーンのため

移行準備期間は2018年3月～7月

移行時の負荷試験

DELiGHT WORKS Inc.

- それまではリリーススケジュールに追われ、
本格的な負荷試験を実施できない状況が続いていました

- 今回の移行は**複雑で大規模**、かつ**失敗した時のリスクが絶大**のため、
相応のスケジュールと人員と予算で徹底的に負荷試験を実施する方針としました

アジェンダ

DELiGHT WORKS Inc.

1. Fate/Grand Orderについて
2. DB移行の経緯
3. DB移行の方法
4. **負荷試験の準備**
5. 負荷試験の実施
6. 移行リハーサルと段取り
7. 移行結果

負荷試験ツールで検証したいこと

本番のピーク時の負荷を、水平分割環境で捌けることの検証

本番のピーク時以上の負荷を、水平分割環境で捌けることの検証

移行リハーサルでの使用

- 本番同様の負荷がかかった状態での移行リハーサルを実施することが目的

異常系の検証

- 移行中、および移行後にDBがダウンした場合の挙動を確認、修正することが目的

負荷試験に求めること

1. 負荷試験ツールに求めること

- 小規模から大規模まで手軽に負荷をスケールさせたい
  - 移行リハーサルから性能限界試験まで使用したいため

2. 負荷試験シナリオに求めること

- 手軽に、何度も、長時間の負荷をかけたい
- 本来のユーザの挙動をできる限り再現したい

3. 負荷試験環境に求めること

- 本番と全く同じ構成の環境を使用したい
- 生のユーザデータを使用したい

負荷試験に求めること

1. 負荷試験ツールに求めること

- 小規模から大規模まで手軽に負荷をスケールさせたい
  - 移行リハーサルから性能限界試験まで使用したいため

2. 負荷試験シナリオに求めること

- 手軽に、何度も、長時間の負荷をかけたい
- 本来のユーザの挙動をできる限り再現したい

3. 負荷試験環境に求めること

- 本番と全く同じ構成の環境を使用したい
- 生のユーザデータを使用したい

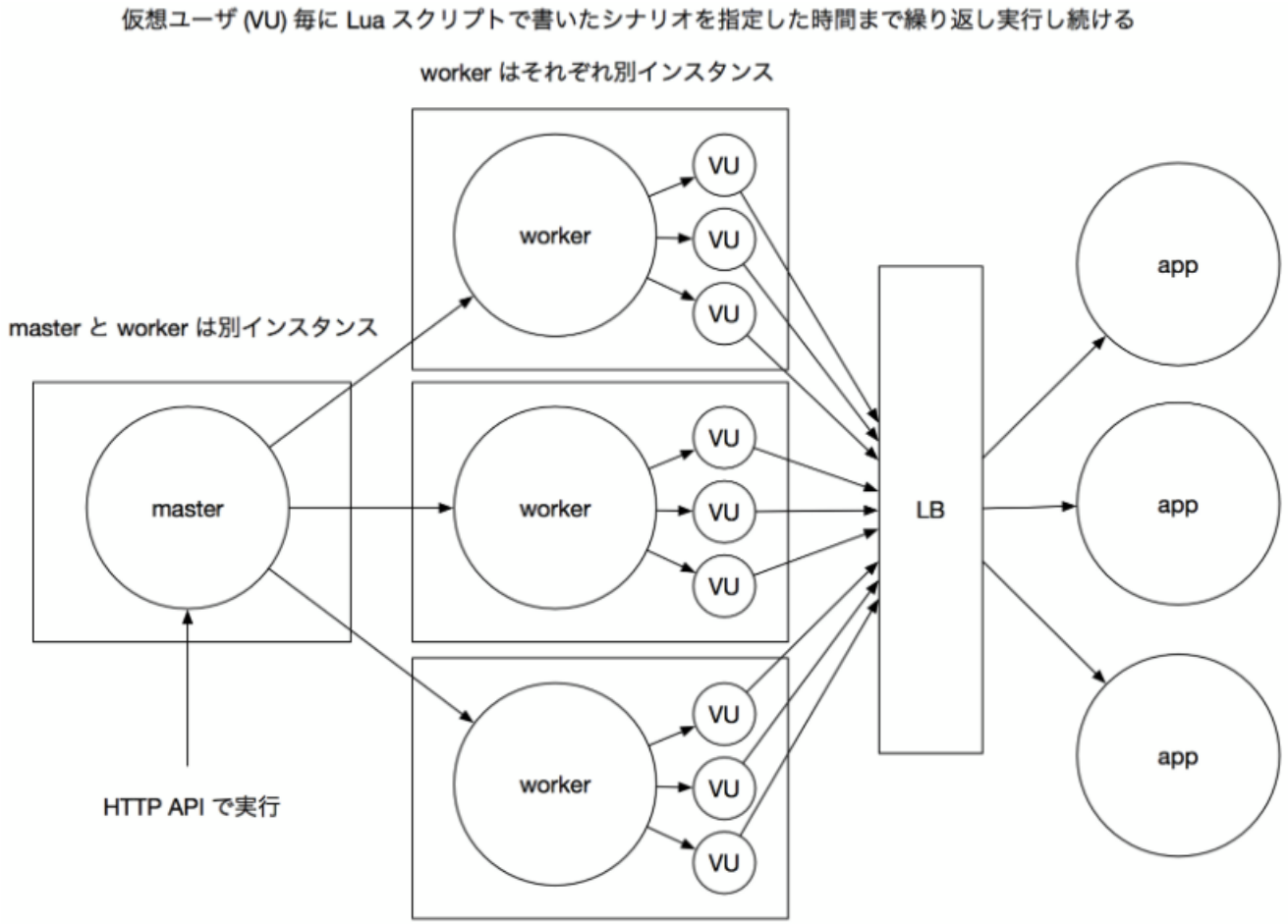
負荷試験ツールの構成

構成

- master（司令塔）　：1台
- minion（VU管理）　：　N台

minionにmasterのprivate ipを設定した状態でAMIを作成

- minionをスケールさせると自動的にmasterにぶら下がる



https://gist.github.com/voluntas/5c9e0f778e36c1e934e83611a94ffdfa
時雨堂 シナリオ負荷試験ツール Oribe 開発ログ

minionの調整

最初は思うように負荷をかけられず、minion側を調整していきました

- **portが枯渇しないように増やす**
  - net.ipv4.ip_local_port_range=10001 65000
- **CPU、メモリ不足を解消するためにインスタンスサイズを上げる**
  - 最終的にc5.4xlargeを使用

負荷試験に求めること

DELiGHT WORKS Inc.

- 1. 負荷試験ツールに求めること
  - 小規模から大規模まで手軽に負荷をスケールさせたい
    - 移行リハーサルから性能限界試験まで使用したいため
- **2. 負荷試験シナリオに求めること**
  - 手軽に、何度も、長時間の負荷をかけたい
  - 本来のユーザの挙動をできる限り再現したい
- 3. 負荷試験環境に求めること
  - 本番と全く同じ構成の環境を使用したい
  - 生のユーザデータを使用したい

シナリオ作成手順

1. 本番の分析

- 1-1. 本番で負荷が高かった時間帯のアクセスログを取得
- 1-2. アクセス回数をAPIごとにカウント

2. シナリオ作成

- 2-1. 1つのプレイサイクルとなるAPIをまとめて、「APIグループ」を作成
  - 例えば「ログイン」や「クエスト」など
  - 実行回数が極端に少ないAPIは除外
  - 実行回数が多いAPIは、繰り返し実行が難しい場合でもなんとかして組み込む
- 2-2. 「APIごとの実行比率」が手順1-2で集計した比率と概ね一致するように、
  「APIグループ内でのAPIの実行回数」と「APIグループごとの実行比率」を調整

シナリオ作成時の留意点

可能な限り、所有物（レコード）が増えたまま「APIグループ」を終わらせない

- 例）プレゼントを貰ったら受け取るところまで行う
- 長時間実行してレコード数が急激に増えていくと最初と最後で検証結果が変わったり、APIサーバが何かしらのエラーを返却したりしてしまう
- 無限に実行可能であることを「APIグループ」単位で保証する

スタック式のアイテムが増えたままシナリオが終わるのは気にしない

- 例）user_item（user_id, item_id, num）のnumが増加してもDBの性能への影響はない

シナリオ実装中の出来事の一例

DELiGHT WORKS Inc.

過去のイベント用クエストを実行するとエラーになる

- マスタデータを書き換えて対応

AP（行動力）が足りないとクエスト開始時にエラーになる

- しかしクエストのAP消費を0にするとその分のupdateが減ってしまう
- AP回復のAPIも適度に実行したい
  - 負荷試験ツールにユーザの状態をキャッシュして、APが足りなくなったらAP回復を実行するように実装
  - 他のクエストも実行できるように、全マスタデータを負荷試験開始時にダウンロードしてキャッシュ

パーティ編成APIは実際に所持しているキャラクターを指定しないとエラーになる

- 所持しているキャラクター情報も負荷試験ツールにキャッシュ

負荷試験に求めること

1. 負荷試験ツールに求めること

- 小規模から大規模まで手軽に負荷をスケールさせたい
  - 移行リハーサルから性能限界試験まで使用したいため

2. 負荷試験シナリオに求めること

- 手軽に、何度も、長時間の負荷をかけたい
- 本来のユーザの挙動をできる限り再現したい

3. 負荷試験環境に求めること

- 本番と全く同じ構成の環境を使用したい
- 本番と同じユーザデータを使用したい

負荷試験環境構築

DELiGHT WORKS Inc.

- **本番と全く同じ構成の負荷試験環境を構築し、本番DBのスナップショットを復元**
- **すべてのDBとRedisをAWS CLIで一括スケールアップ、ダウンするスクリプトを用意**
  - 負荷試験環境を維持するコストが高いため
- **本番DBの最新の定期スナップショットを負荷試験環境に復元するスクリプトも用意**
  - 試験中にユーザデータがおかしくなった場合のため
  - 定期的にユーザデータを最新化するため

試験用ユーザデータの準備

シナリオを無限に実行できるようにするための準備

- AP回復アイテムをMAXまで増やすなど

ユーザデータの条件を以下とした

- 特定クエストをクリア
  - データ量が多いユーザ、少ないユーザに偏ると本番の挙動に近くならないため、「ある程度進めたところにあるクエスト」を条件とした
- ログインが直近
  - 休眠ユーザは直近の新機能のデータが存在していない恐れがあるため

こうして負荷試験の準備が完了

AWSで負荷試験を行う場合の注意点

- **負荷をかける側、かけられる側のEC2、Auroraのインスタンス数、クラスター数の緩和申請が必要となるケースが多いです**
  - 緩和には数日を要するため、早めに確認・申請を
- **負荷をかけるサーバを配置するAWSアカウントは、他の用途で使用するAWSアカウントと分けることが推奨されています**
  - 負荷をかける側で想定外のトラフィックが発生した場合（大規模な負荷試験の実施にあたり、負荷試験の申請を行い、その申請値以上になった場合など）、AWS基盤側でトラフィックを絞ることがあるため、その際に稼働中の他のサーバに影響を与えないためです

    （※）発表時の内容から加筆・修正しています
- **1Gbpsまたは1Gppsを超える負荷試験は事前に申請する必要があります**
  - 詳しくは「Amazon EC2 Testing Policy」をご参照ください

    https://aws.amazon.com/jp/ec2/testing/

アジェンダ

DELiGHT WORKS Inc.

1. Fate/Grand Orderについて
2. DB移行の経緯
3. DB移行の方法
4. 負荷試験の準備
5. **負荷試験の実施**
6. 移行リハーサルと段取り
7. 移行結果

負荷試験の検証内容

- 1. 本番のピーク時の負荷を、水平分割環境で捌けることの検証
- 2. 本番のピーク時以上の負荷を、水平分割環境で捌けることの検証

負荷試験実施（垂直分割環境）

まずは垂直分割環境で、2018年1月のピーク時と同じ規模の負荷をかけました

- 調整の結果、minion（c5.4xlarge）× 30台で同等となった

確認観点

- **DB、APIサーバのLatency上昇が再現すること**
- **DBと各APIのThroughputが、ピーク時の本番同等であること**
  - 誤差1割以下が目標

これにより「負荷試験ツールとシナリオの正当性」を検証

負荷試験実施（水平分割環境）

- 次は水平分割環境で、先ほど同様の負荷をかけました

- 結果、DBのボトルネックが解消し、

APIサーバのLatencyは下がり、Throughputも改善しました

確認観点（1/3）

APIサーバ

- ログ
  - エラーログが大量に流れていないか
  - アクセスログが止まっていないか（負荷試験ツール側に問題が発生していないか）
  - APIの実行比率が本番同等か
- CPU
  - 50%程度に留まっているか
  - 特定のインスタンスのみCPUが高くなってないか
  - 特定のインスタンスがダウンしていないか
- Latency

項目	目標値
Average Latency	100ms

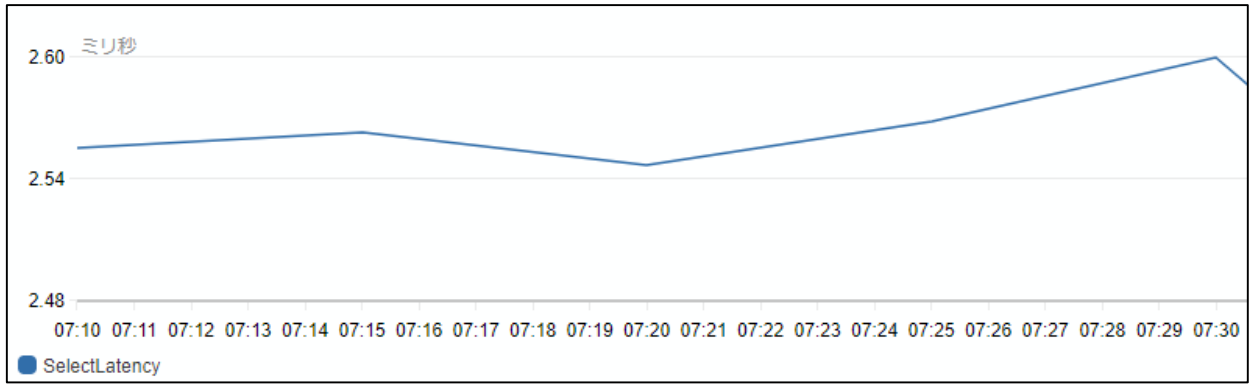
確認観点（2/3）

DB

- Throughput
  - 特定のDBのみ高くないか
  - 本番と比較して少なくないか
- Latency
  - 目標値は以下の通り

項目	目標値
SelectLatency	1ms
InsertLatency	1ms
UpdateLatency	1ms
CommitLatency	10ms

【参考】垂直分割構成の本番のSelectLatency



確認観点（3/3）

DELiGHT WORKS Inc.

Redis

- CPU
  - 50%程度に留まっているか
- メモリ
  - 十分な空きがあるか
  - 急激に上昇していないか

結果

すべて目標値を達成

- この時点で、目標としていた3周年は乗り切れそう

負荷試験実施（水平分割環境/2倍の負荷）

- 次に、2018年1月のピーク時の2倍の負荷をかけました
  - 先ほどの倍であるminion（c5.4xlarge）× 60台で実施

結果

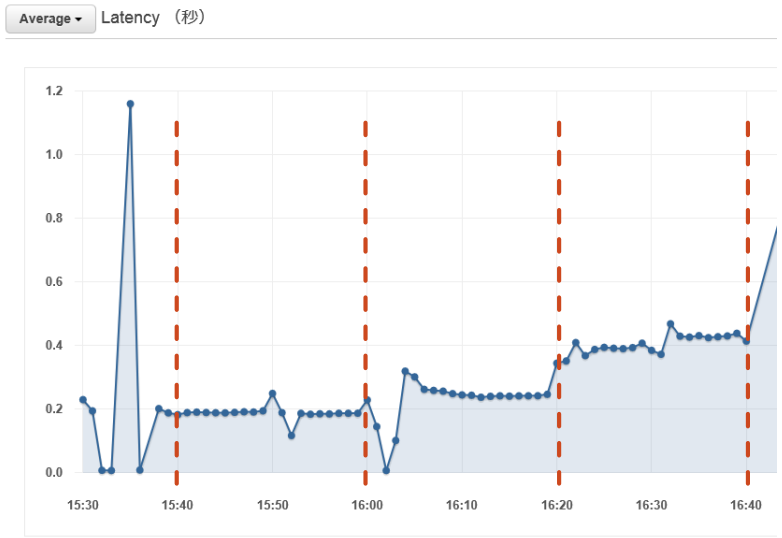
APIサーバのLatencyが上がりました

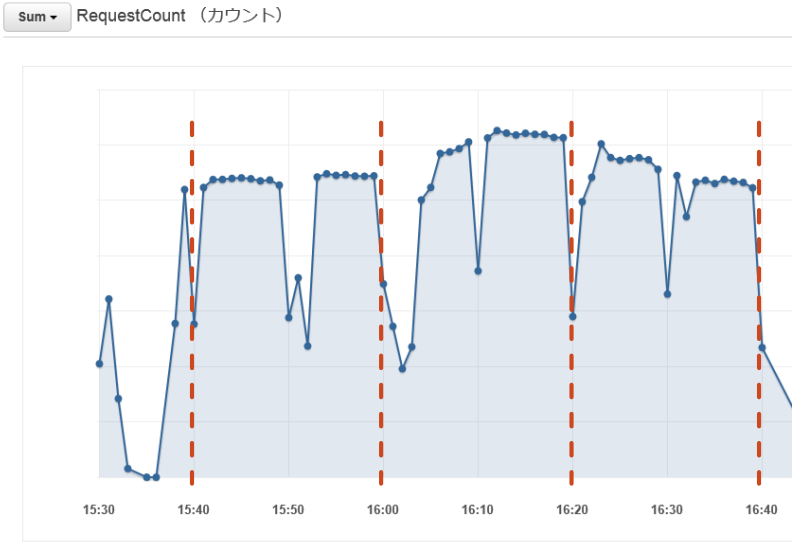
- DBが先にボトルネックになっていたために顕在化しなかったボトルネックが発覚

参考までに、段々とユーザ数を増やすとどうなるかも試してみました

- 20分ごとにユーザ数を増加
- Latencyが段々と上がり、

RequestCountは伸びなかった





新たなボトルネックと対策

RedisのCPU使用率が100%に張り付いた

- Redisはシングルスレッドで動作するため短時間の集中的なアクセスに強くない
- データの特性的にMemcachedで代用可能だったため一時的に置き換えて検証
  - MemcachedとRedis の比較

    https://docs.aws.amazon.com/ja_jp/AmazonElastiCache/latest/mem-ug/SelectEngine.html

APIサーバのCPU使用率も70%ほどまで上昇した

- APIサーバの台数を一時的に増やして検証（45 -> 100台）
  - DBとのコネクション数は減らした

複数のshardを参照するクエリが重い

- 最低限必要なカラムのみSELECTするようにした

再検証結果

一部、目標値を上回ったものの、ピーク時の2倍でこれであれば許容範囲と判断

APIサーバ

項目	目標値	結果
Average Latency	100ms	約120ms

DB

項目	目標値	結果
SelectLatency	1ms	達成
InsertLatency	1ms	達成
UpdateLatency	1ms	約2ms
CommitLatency	10ms	達成

検証結果からの考察

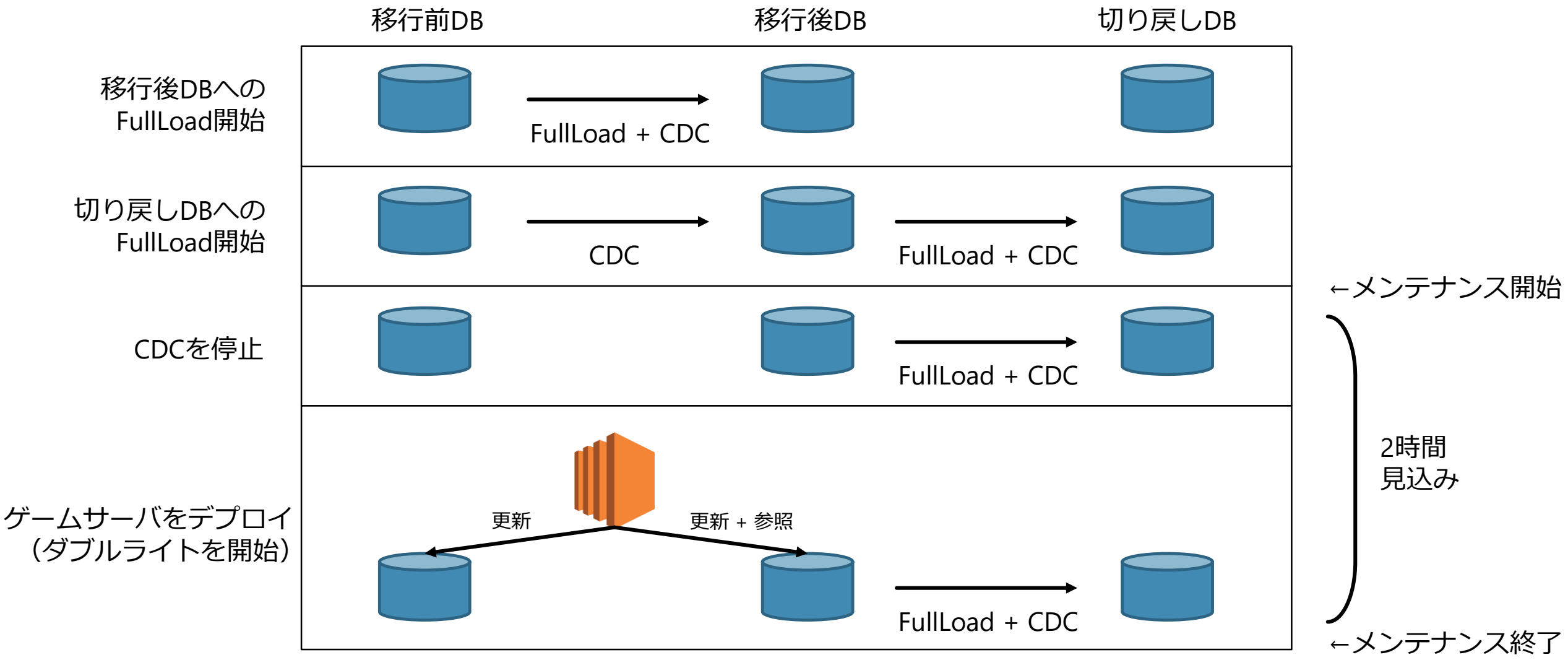
DB移行とRedis -> Memcached移行を同時に実施すると複雑さが増すため、
後追いで実施することにしました

アジェンダ

DELiGHT WORKS Inc.

1. Fate/Grand Orderについて
2. DB移行の経緯
3. DB移行の方法
4. 負荷試験の準備
5. 負荷試験の実施
6. **移行リハーサルと段取り**
7. 移行結果

移行の流れ（当初想定）



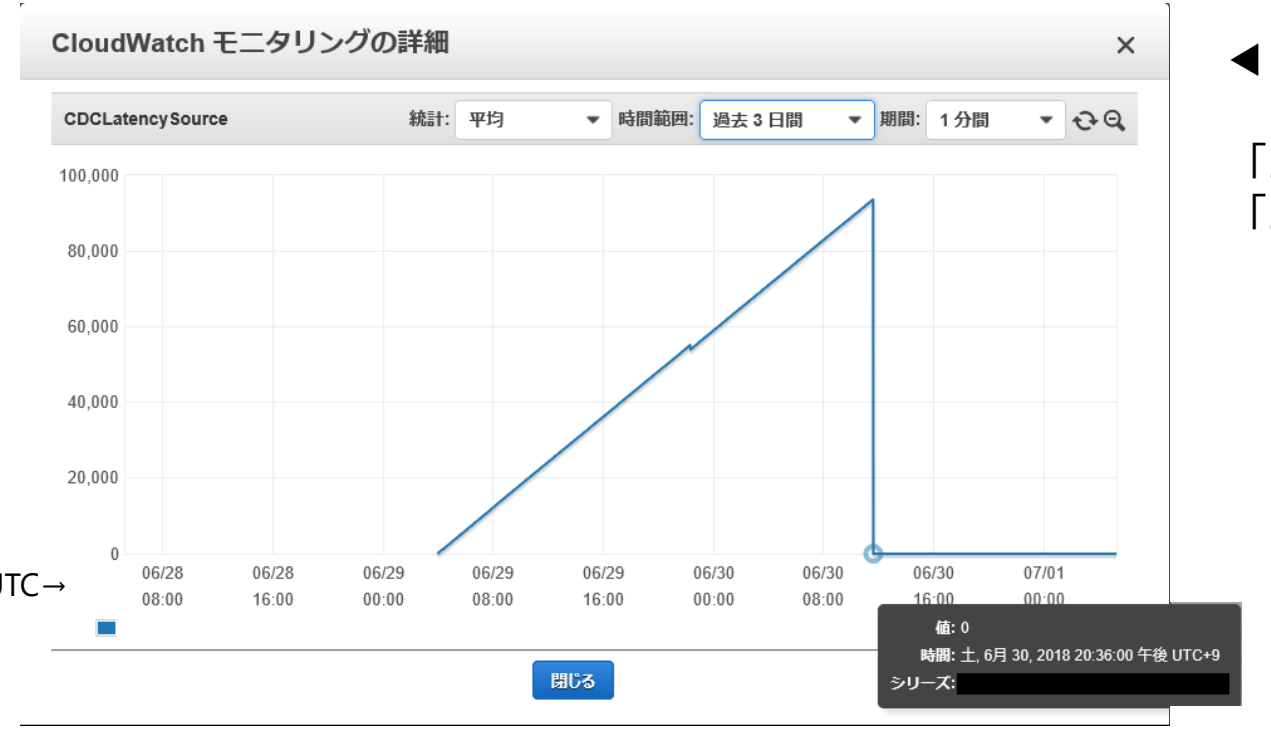
移行リハーサル

- 負荷試験環境にて、本番と同等の負荷をかけながら移行リハーサルを実施

リハーサル結果（1/3）

移行前DB→移行後DBのCDCが追い付かないことが判明

- binlogスレッドが複数起動し、mutexの奪い合いが発生



※UTC→

◀DMSタスクのLatencyの一例

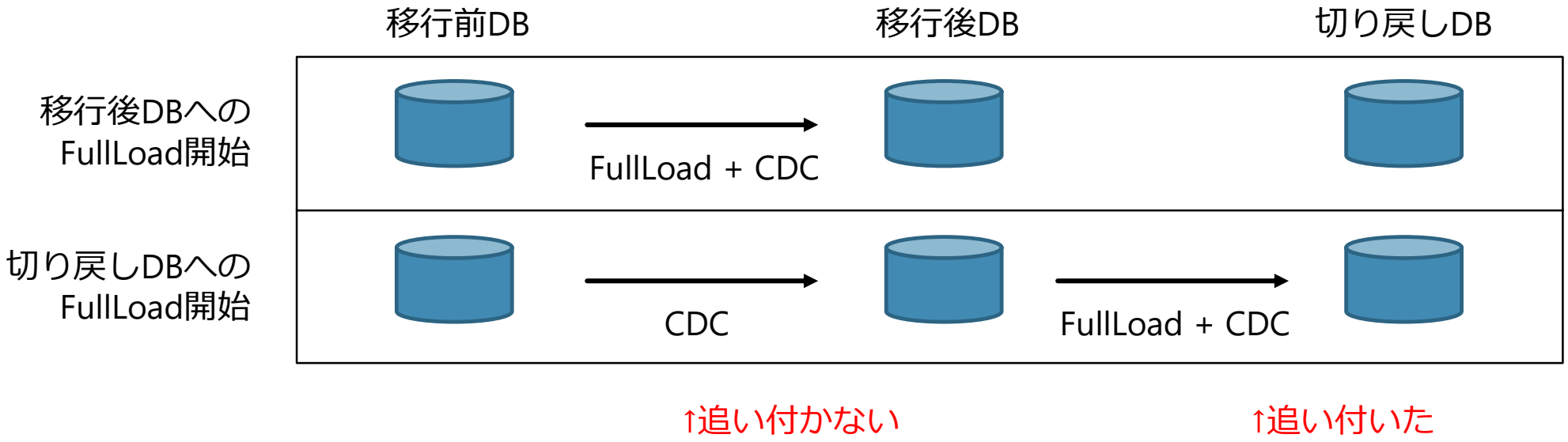
「2018/6/29 18:35」に負荷を停止し、
「2018/6/30 20:36」にようやく追い付いた

リハーサル結果（2/3）

- **再度負荷をかけながら、今度は移行後DB→切り戻しDBを検証**
  - こちらは追い付いた
    - ソースデータストアへのCommit数が少ないための模様
- **すべてのFullLoad完了後に全データ整合性を確認**
  - 問題なし

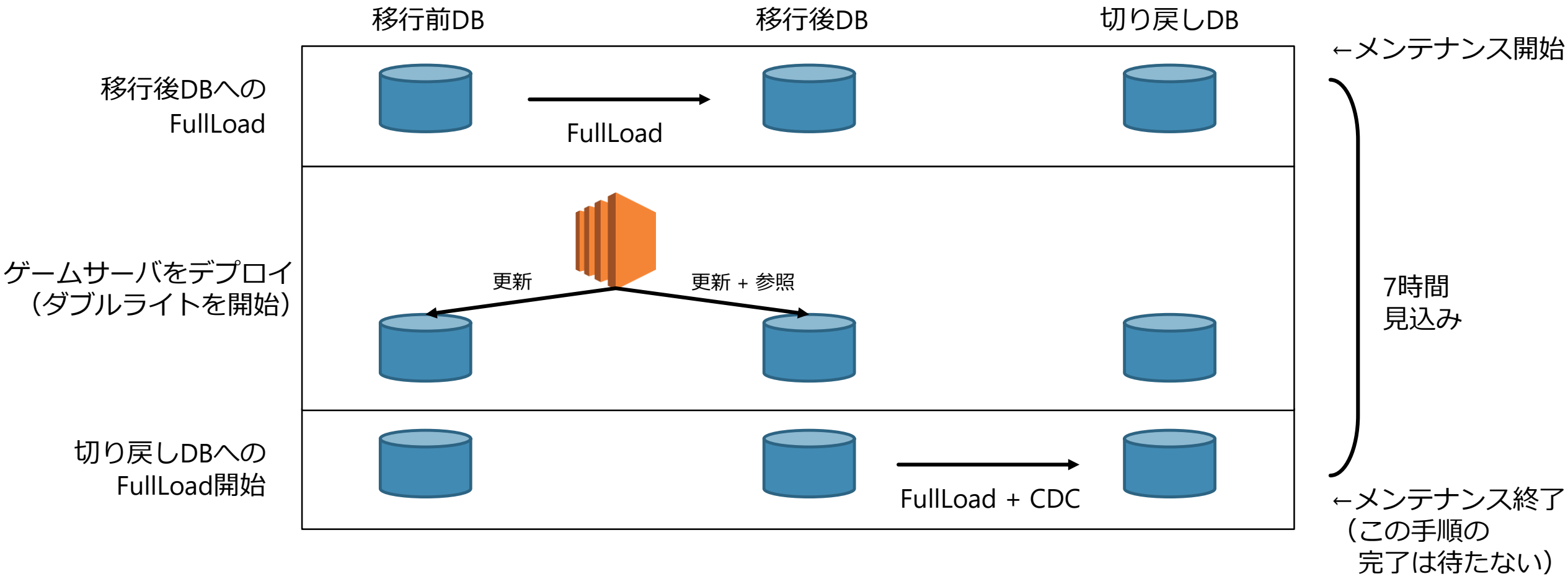
リハーサル結果（3/3）

状況まとめ



メンテナンス手順を再考

移行の流れ（リハーサル結果を考慮後）



異常系試験

本番同等の負荷をかけつつ、QAチームが端末でプレイしながら、Auroraの障害挿入クエリで異常系試験を実施しました

- https://docs.aws.amazon.com/ja_jp/AmazonRDS/latest/AuroraUserGuide/AuroraMySQL.Managing.FaultInjectionQueries.html

試験内容は以下の通りです

- ダブルライト期間中の障害検証
- ダブルライト期間後の障害検証

ダブルライト期間中の異常系試験結果（1/2）

DELiGHT WORKS Inc.

- 障害挿入後、移行前DBと移行後DBの全データをチェック
  - XA（TransactionScope）実装済みにも関わらず不整合が発生

ダブルライト期間中の異常系試験結果（2/2）

以下の理由によりXAは外しました

- どのみちデータ不整合が発生する
  - 詳しい原因を調査する時間もなく…
- Auroraでは使用しないことを推奨されている
  - https://docs.aws.amazon.com/ja_jp/AmazonRDS/latest/AuroraUserGuide/AuroraMySQL.BestPractices.html#AuroraMySQL.BestPractices.XA
- 極稀にtrx_mysql_thread_id=0のトランザクションが残る
  - 特定のレコードがロックされたままになる

ダブルライト期間中（10日間）は処理中断モードで乗り切ることに

- DB Commit失敗時、一時的にすべての処理を止めて被害の拡大を防ぐ

ダブルライト期間後の異常系試験結果

- データ不整合、欠損などなし
- DB自動復旧後のプレイにも影響なし

移行リハーサル（最終チェック）

- 移行の全作業を最後に通しでリハーサルし、滞りなく完了

▶実際の移行手順

メンテナンス開始

No.	作業内容
1.	バッチ処理を停止
2.	全DBのスナップショットを取得
3.	移行前DB→移行後DBのFullLoadを開始（CDCは行わない）
4-1.	移行後DBのインデックスを作成
5-1.	正常系確認（各shardごとに検証）
	予備

No.	作業内容
4-2.	APIサーバをデプロイ バッチ処理を更新
5-2.	移行前DB→移行後DBのVerify Tool実行 （1時間で完了する範囲でレコードの正当性を検証）

No.	作業内容
5-3.	移行後DB→切り戻しDBのFullLoadを開始
6.	切り戻しDBのインデックスを作成
7.	移行後DB→切り戻しDBのCDCを開始

メンテナンス終了

本番作業

- **2018年7月11日**
  - 計画メンテナンス実施
    - 水平分割移行の実作業
- **2018年7月13日**
  - 計画メンテナンス実施
    - スナップショット取得
  - 全データチェック
    - 移行後DB（本番使用中）は問題なし
    - 移行後DB→切り戻しDBにてデータ移行漏れあり
- **2018年7月19日**
  - DMSの設定（wait_timeout）を見直し、移行後DB→切り戻しDBのFullLoad + CDCをオンラインで再実行
- **2018年7月20日**
  - ダブルライトを停止
  - 全データをチェックして不整合がないことを確認

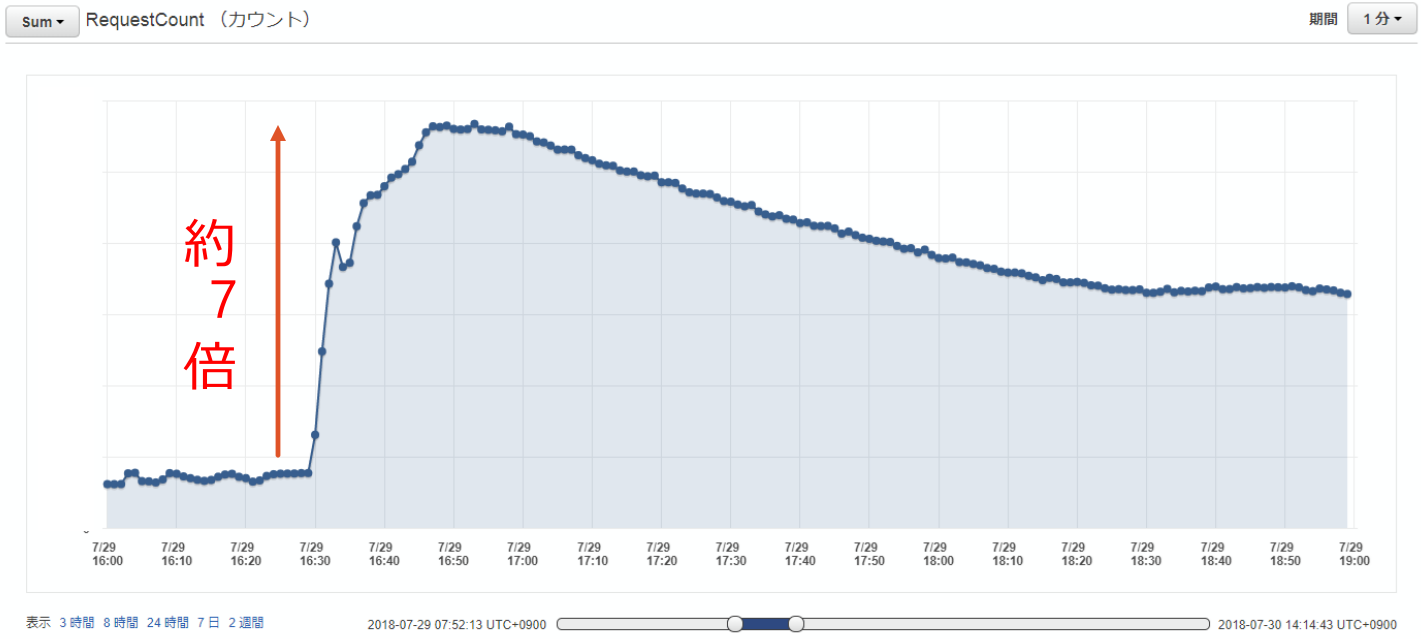
アジェンダ

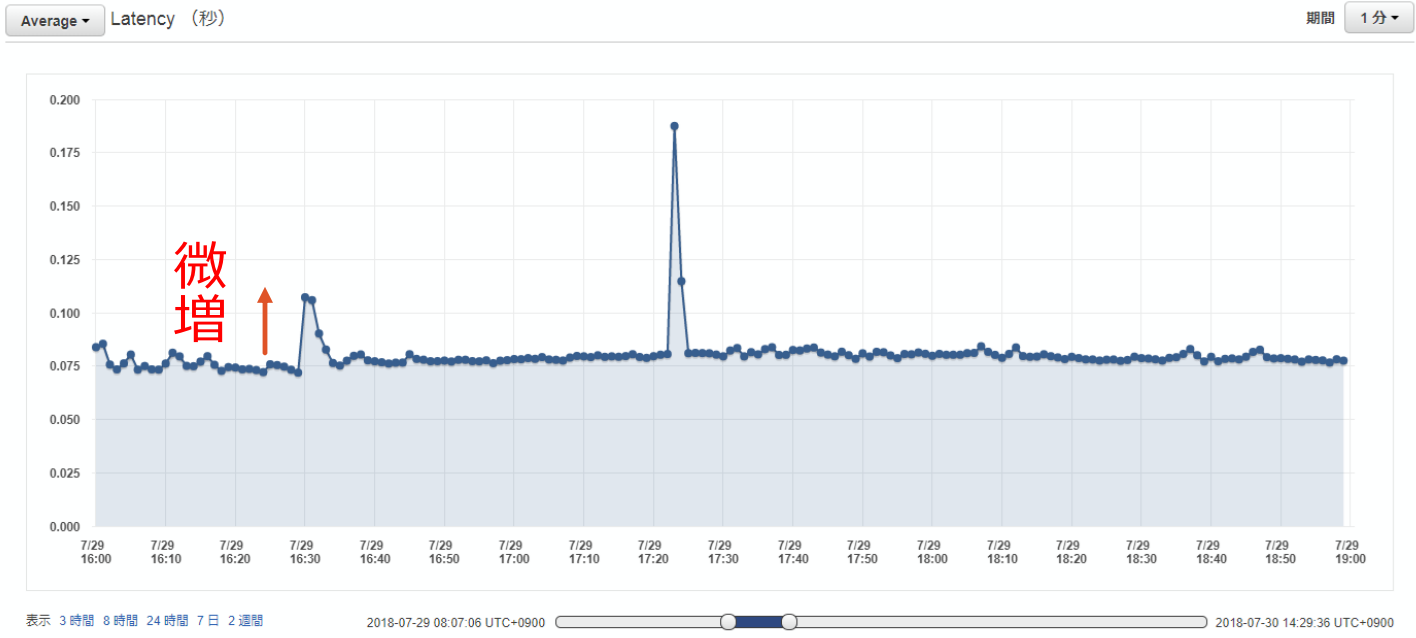
DELiGHT WORKS Inc.

1. Fate/Grand Orderについて
2. DB移行の経緯
3. DB移行の方法
4. 負荷試験の準備
5. 負荷試験の実施
6. 移行リハーサルと段取り
7. **移行結果**

FGO3周年キャンペーン

- かなりのアクセスがあったものの
全く問題ありませんでした

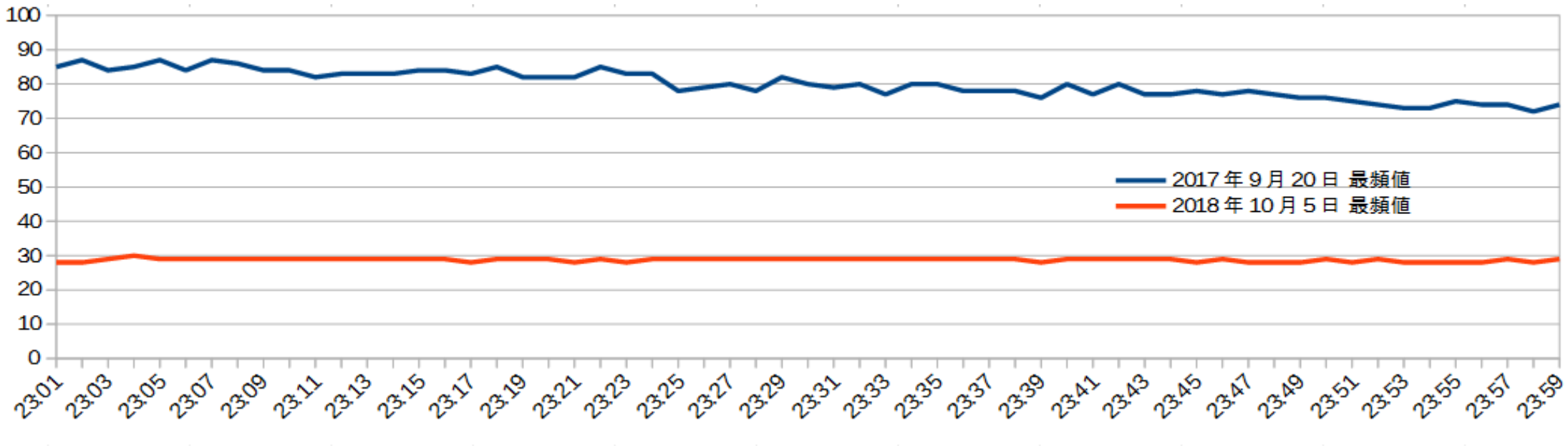




ボックスガチャイベント

2017年秋と2018年秋のボックスガチャイベントの最終日を比較

- ボックスガチャAPIのLatencyが半減以下



2019年元旦

1時間ほどアクセスしづらい時間があったが復旧

- DBに問題はなし
- 原因は判明しているので対策を検討中

まとめ（DB移行）

DELiGHT WORKS Inc.

- スケール可能な構成をリリース前から考えておくことの重要性を痛感しました
- AWS DMSで大規模なDB移行を行うことは可能です
  - ただし、入念なリハーサルは必須です

まとめ（負荷試験）

DELiGHT WORKS Inc.

負荷試験の手法は、その目的やWebサービスの特性によって大きく異なります

しかし、共通して言えることは以下の通りです

- インフラ（Webサーバ、DBサーバ、モニタリングツール、プロファイリングツール）の知識、プログラムの知識、ゲーム仕様の理解が必要で、なおかつ細かい調整を繰り返す根気が必要となる泥臭い作業ですが、地道に一つずつ問題を解決していけば必ず成果を得られると実感しました
- 負荷試験は決して片手間にやるものではなく、専任の担当者が、余裕を持ったスケジュールで行うべきです

大規模移行であっても本番想定の試験をしておけば

理屈上はうまくいくはずですし、実際にうまくいきました

終わりに

このたびの移行において、

- Amazon Web Services様
- スカイアーチネットワークス様（MSP）
- 時雨堂様（負荷試験ツール開発、テストシナリオ作成）

には多大なる助力を賜りました

この場を借りて御礼申し上げます

ディライトワークスは
エンジニア積極採用中です！

https://recruit.delightworks.co.jp/