AWSを活用した Yahoo! MOBILE INSIGHTの 構築事例

2018年6月22日

ヤフー株式会社

坂田 悠

自己紹介

◆ 坂田 悠 (さかた ゆう)

◆ ヤフー株式会社

メディアカンパニー
プラットフォーム統括本部
データ開発本部
プロダクト開発1部
Measurement & Attribution1　リーダー

◆ 広告システムのバックエンド開発

スポンサードサーチ, アプリインストール広告
Yahoo! MOBILE INSIGHT

アジェンダ

- ◆Yahoo! MOBILE INSIGHTの概要
- ◆AWS導入の経緯
- ◆現状のバックエンド構成
- ◆システム負荷対策の取り組み
- ◆オンプレとクラウドの共存
- ◆最後に

アジェンダ

◆Yahoo! MOBILE INSIGHTの概要

◆AWS導入の経緯

◆現状のバックエンド構成

◆システム負荷対策の取り組み

◆オンプレとクラウドの共存

◆最後に

会社概要

営業利益の大半はメディア事業 (特に広告などマーケティング領域)

◆メディア事業

・検索、ニュース、天気、路線情報、動画、地図、メールなど

・マーケティング領域

広告配信

プレミアム広告、プロモーション広告、プレミアムDSPなど

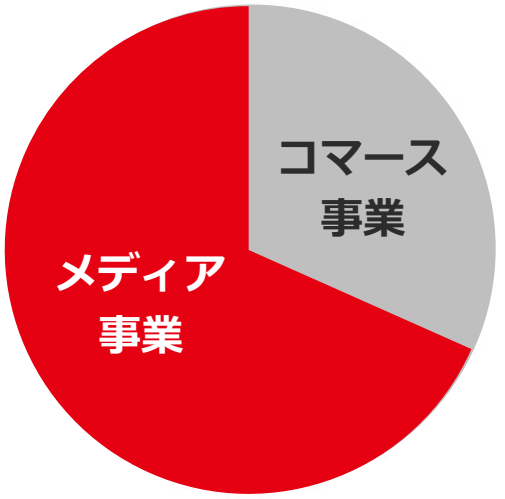
効果測定

YAHOO! JAPAN MOBILE INSIGHT

◆コマース事業

・オークション、ショッピング、決済金融、プレミアム会員など

営業利益 内訳



FY2017

IRより抜粋 (一部改変)

YAHOO! JAPAN MOBILE INSIGHT

アプリ事業者向けのマーケティングツールとして 2015年12月にリリース

モバイルインサイト HP

管理画面

YAHOO! JAPAN MOBILE INSIGHT 特徴

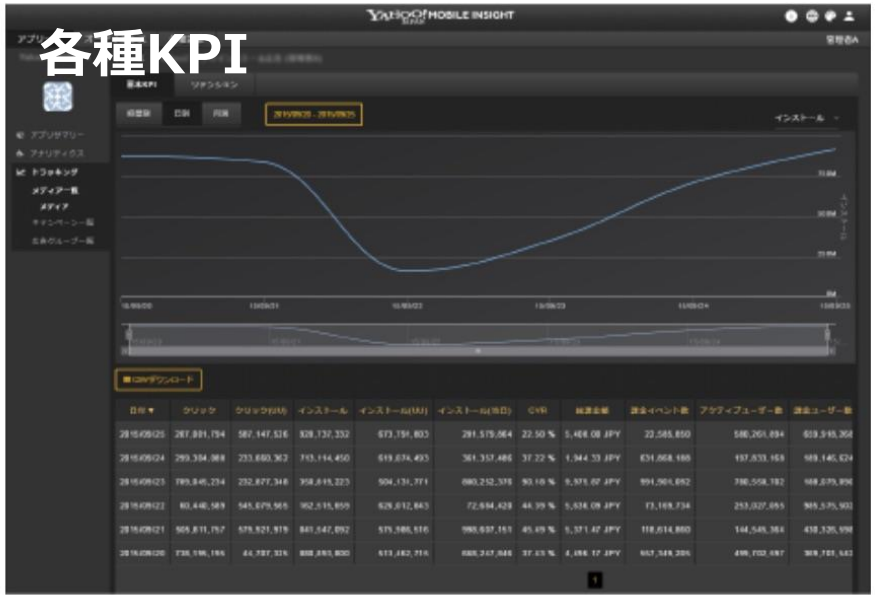
アプリ内にSDKを組み込むだけでご利用可能です

対応プラットフォーム

iOS, Android, Unity, Cocos2dx, Cordova, Monaca, WebView

広告効果測定（トラッキング）機能

広告経由の各種成果(インストール、課金、起動数、継続率など)を計測・比較できる



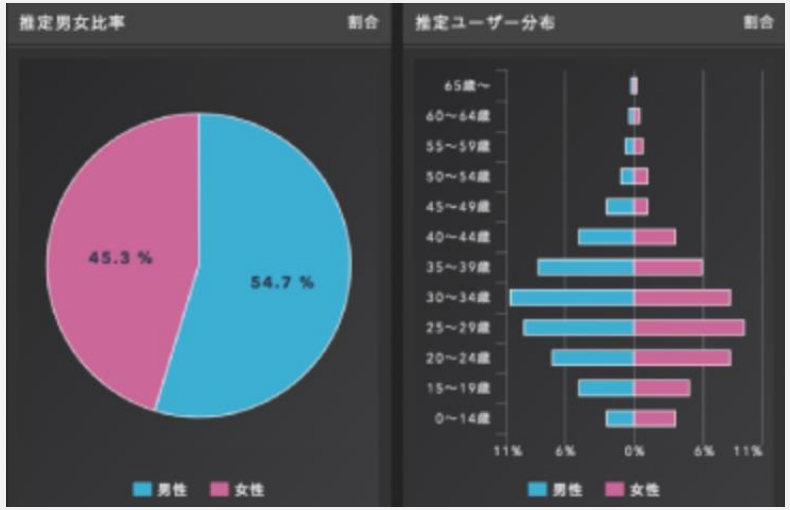
パートナーメディア(一部)　国内主要メディアはほぼ網羅。海外主要メディアとも接続拡大中。

Twitter、nend、i-mobile、FreakOut、AppLovin、Google AdWords、Yahoo!プロモーション広告 など

アナリティクス機能

アプリ内ユーザの解析機能を提供

・各種KPI (アクティブユーザ数、新規ユーザ数、セッション数、イベント数など)
・ユーザ分布 (推定年齢・性別、地域、言語、アプリバージョン、キャリア、課金額など)



ヤフーのデータ使った推定デモグラ分布

※アプリプロモーションを行わない場合でもご利用可能

導入実績

◆アカウント数

150以上 (2018.04時点)

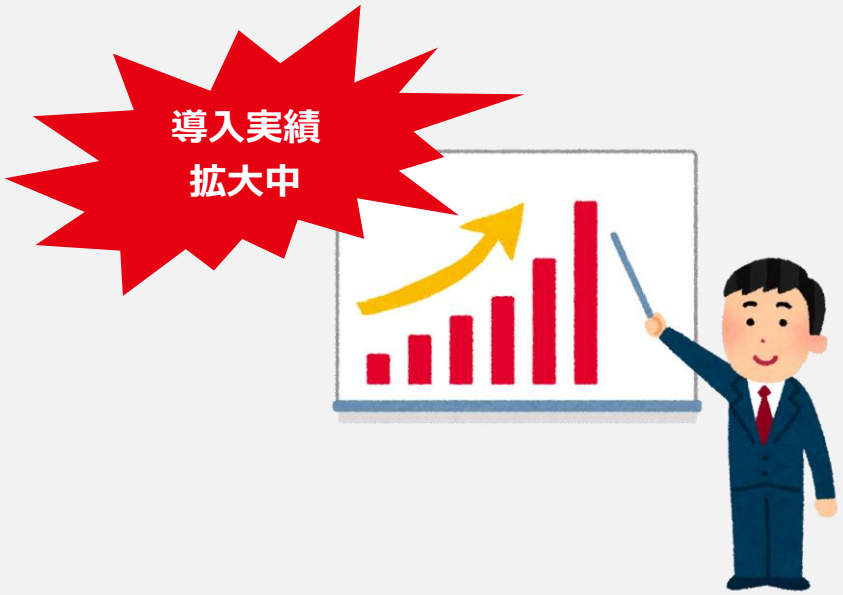
導入済み企業 (一例)

IGNIS様、エイチーム様、グッドラックスリー様など

◆アプリ数

250以上 (2018.04時点)

※ヤフーのアプリにも導入が続々と進行中



アジェンダ

◆Yahoo! MOBILE INSIGHTの概要

◆AWS導入の経緯

◆現状のバックエンド構成

◆システム負荷対策の取り組み

◆オンプレとクラウドの共存

◆最後に

そもそもヤフーには膨大なオンプレ資産

◆国内最大級のサーバリソース保有企業

◆データセンターは国内外あわせて8箇所以上

◆ヤフーのほぼすべてのサービスはオンプレで構築

北九州データセンター外観

なぜAWS導入したのか？

経緯① 新規事業としてスタート

社内ベンチャー あるあるな状況

◆予算が少ないところからスタート

スモールスタートでグロースしなかった場合は容易にクローズできるように

◆最小の人員構成

運用コストをできる限り抑えたい

◆シビアな開発工数

AdTech Tokyo 2015 で当時のサービス責任者がリリース宣言するために、
開発開始から 約半年でローンチ

経緯② クライアント・パートナーデータの分離

◆ 一部のクライアントやパートナーデータは契約上ヤフーで利用不可

データソース		データの帰属
クライアント（SDK）		
	オープンプラン契約 (無料)	ヤフー + クライアント
	プライベートプラン契約 (有料)	クライアントのみ
メディアパートナー（広告媒体）		メディアパートナーのみ

◆ ヤフーのサービスとはデータを分けて管理したい

ACLの問題などがあるため、お客様のデータをヤフーのデータパイプラインにのせない

クラウド利用でネットワーク的にデータ分離させることで実現

経緯③ キャパシティ予測が困難

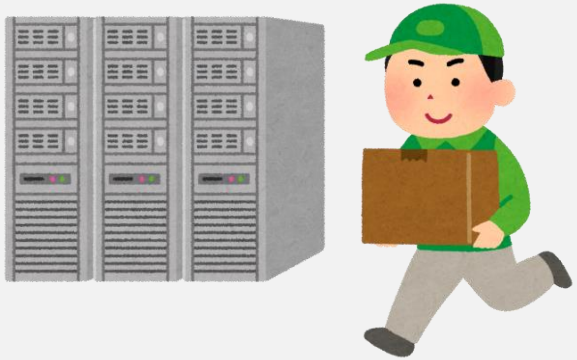
◆導入アプリが増えるたびにリクエストが増加

ビッグアプリが1個導入されただけで激増

◆(大規模な)リソース追加のリードタイムを短くしたい

SDKの導入は容易 (最小限の導入なら１日以内で完了)

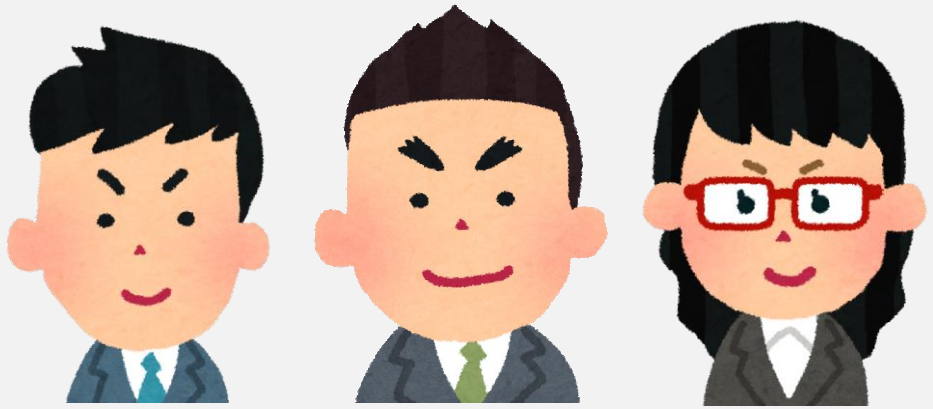
※現在のオンプレでは数百台規模なら数時間以内で増設が常時可能だが
開発開始当初はリソース空き状況によっては常時可能とは限らなかった



経緯④ クラウド開発への興味

◆社内でも新技術に挑戦したい開発メンバーが揃っていた

◆当時の開発リーダー(ベンチャーからの転職組)が AWS経験者だった



新技術にチャレンジしたいメンバー

AWS経験あるリーダー

AWS導入経緯のまとめ

◆新規事業でスモールスタート

◆ヤフーのサービスデータとネットワーク的に分離

◆キャパシティ予測が困難

◆新技術にチャレンジしたいメンバー
(+AWS経験者のリーダー)

アジェンダ

◆Yahoo! MOBILE INSIGHTの概要

◆AWS導入の経緯

◆現状のバックエンド構成

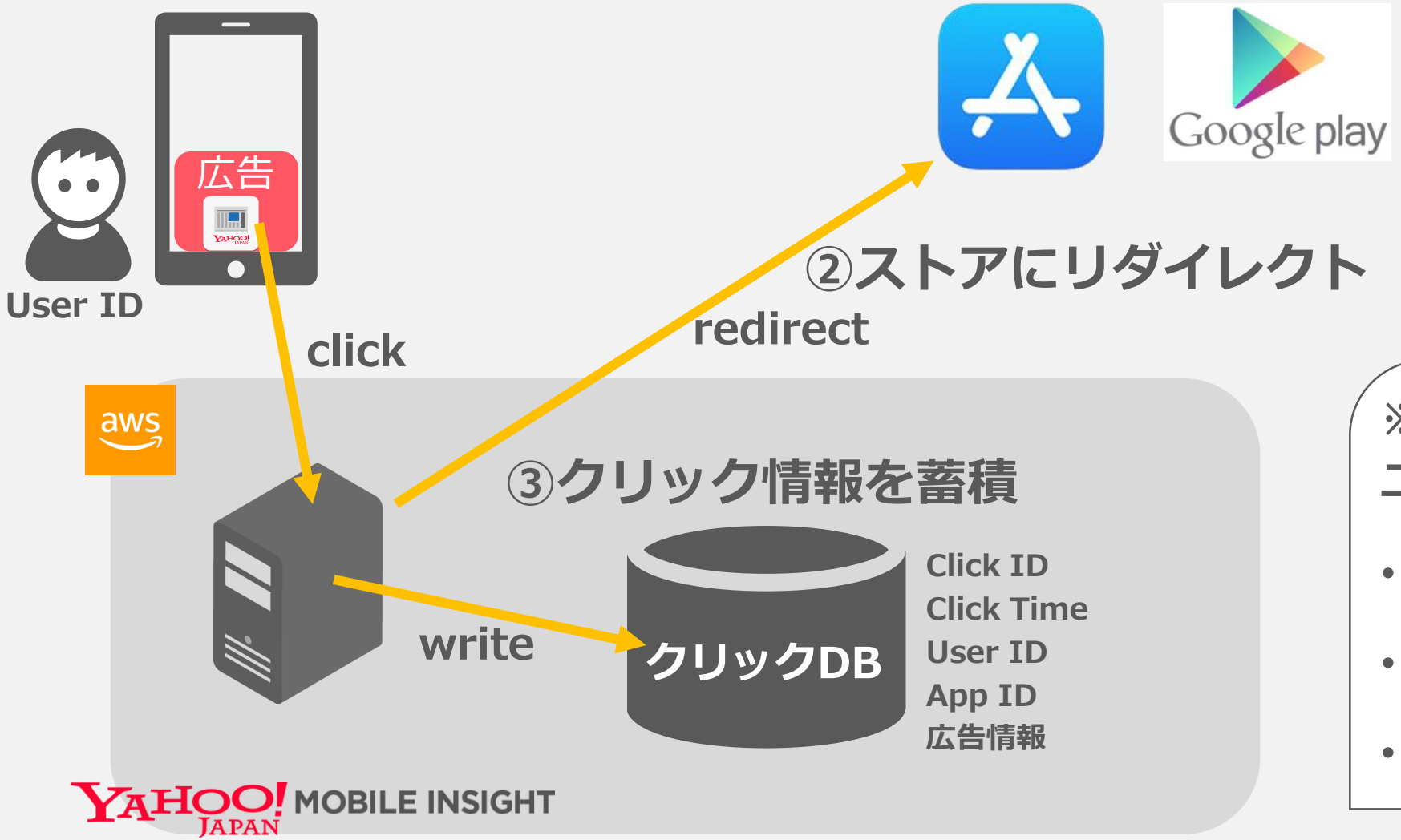
◆システム負荷対策の取り組み

◆オンプレとクラウドの共存

◆最後に

広告効果計測(トラッキング)の仕組み

① エンドユーザが広告をクリック



※参考
ユーザ識別子 (User ID)

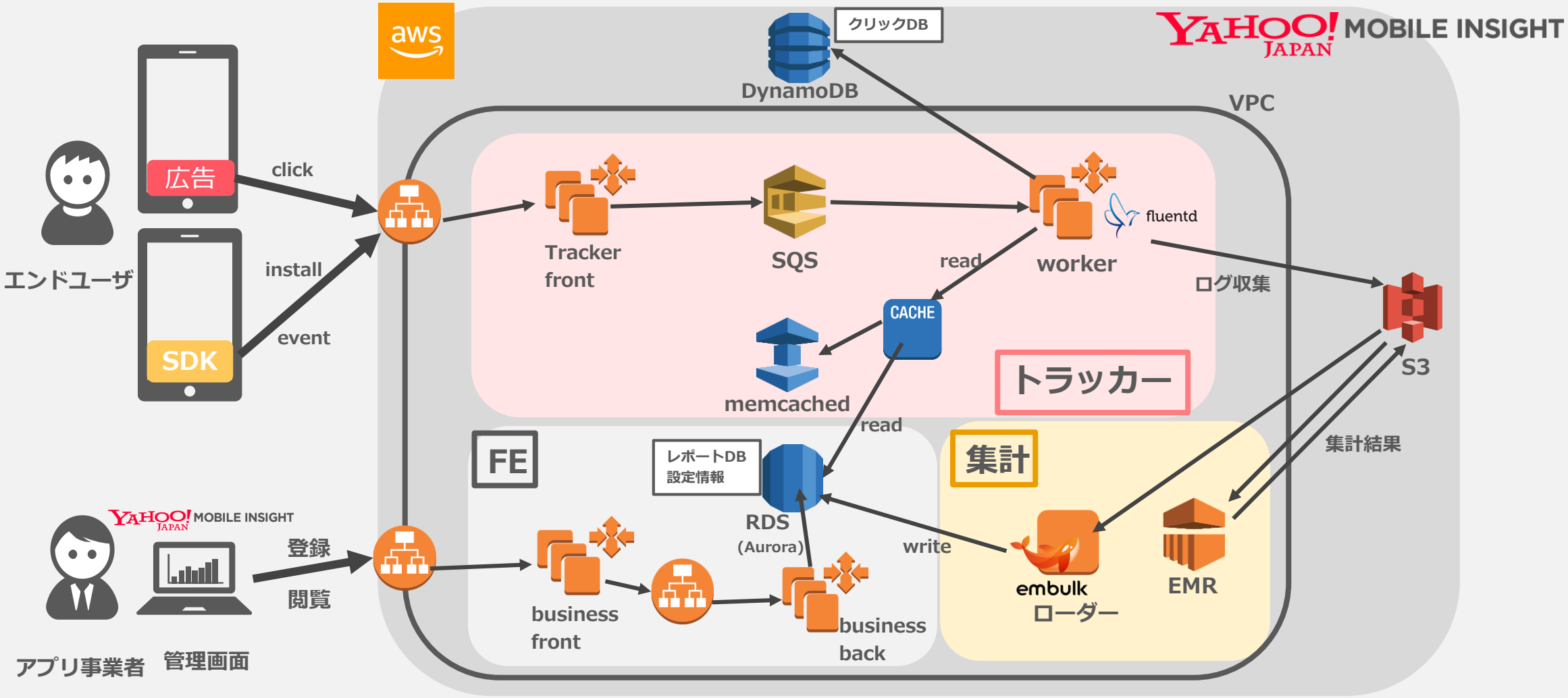
- IDFA (iOS)
- GAID (Android)
- Fingerprint (Web)

YAHOO! JAPAN MOBILE INSIGHT

広告効果計測(トラッキング)の仕組み

④ ストアでAppをインストール＆初回起動

バックエンド構成 (全体像)

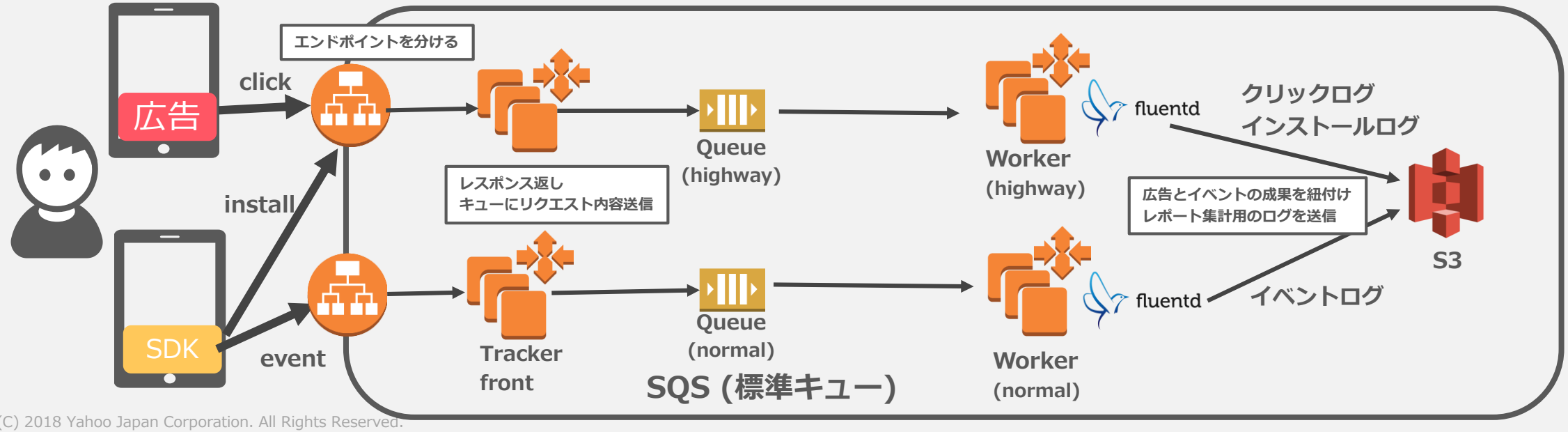


コンポーネントは大きく トラッカー、集計、FE の3つ

トラッカー構成

キューワーカー アーキテクチャを採用

- ストアへのリダイレクト などレスポンスを即座に返さなければならない
  遅延すると 広告主に金銭的被害
  処理に時間のかかるトラッキング処理は非同期でワーカーが担当する
- キューはスケーラビリティとパフォーマンスを考慮し SQS (標準キュー)を採用
- 2つのキュー構成 (highwayとnormal) クリックや初回起動など先に処理完了すべきものをhighwayに流す



集計構成

◆ストレージとコンピュートの分離

生ログ、中間データ、集計結果などすべてS3に保存 (S3の高可用性を享受)

◆EMRクラスタを毎回新規構築 (定常集計ジョブフローごと)

クラスタ運用コストが不要 (デッドノード交換やセキュリティパッチ対応からの解放)

費用面でもメリット (秒単位課金．リスクなしで完全スポットインスタンス化)

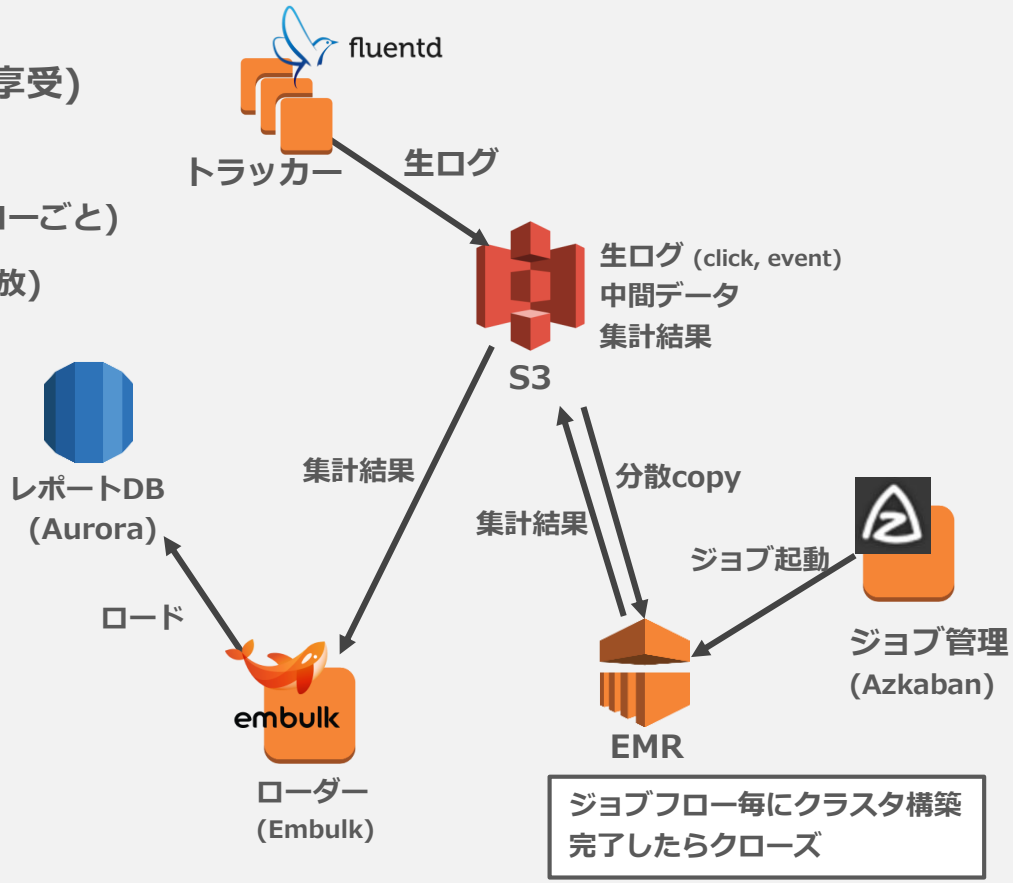
※ただしクラスタ構築のため5分程度のオーバーヘッド

◆ジョブ管理・ロードは OSSを利用

ジョブ管理: Azkaban

ロード: Embulk

※ 運用コストや可用性の面からAWS Glueの利用などサーバレス化要検討



現状のバックエンド構成まとめ

- **広告効果計測(トラッキング)の仕組み**
  - 広告クリック情報を蓄積
  - インストールなど各種成果に紐付けレポーティング
- **トラッカー・集計・FE の 3コンポーネント**
- **トラッカー構成**
  - キューワーカーアーキテクチャ
  - SQS(標準キュー)で 2キュー構成
  - トラッカーフロント、ワーカーともにオートスケール
- **集計構成**
  - データはすべてS3に保存
  - 集計ジョブフローごとに毎回EMRクラスタ新規作成
  - ジョブ管理・ロードはそれぞれOSS利用し構築

アジェンダ

◆Yahoo! MOBILE INSIGHTの概要

◆AWS導入の経緯

◆現状のバックエンド構成

◆システム負荷対策の取り組み

◆オンプレとクラウドの共存

◆最後に

急増するリクエストと問題点

導入アプリの増加とともに

現状のシステム構成だと対処できない問題 が発生

課題1

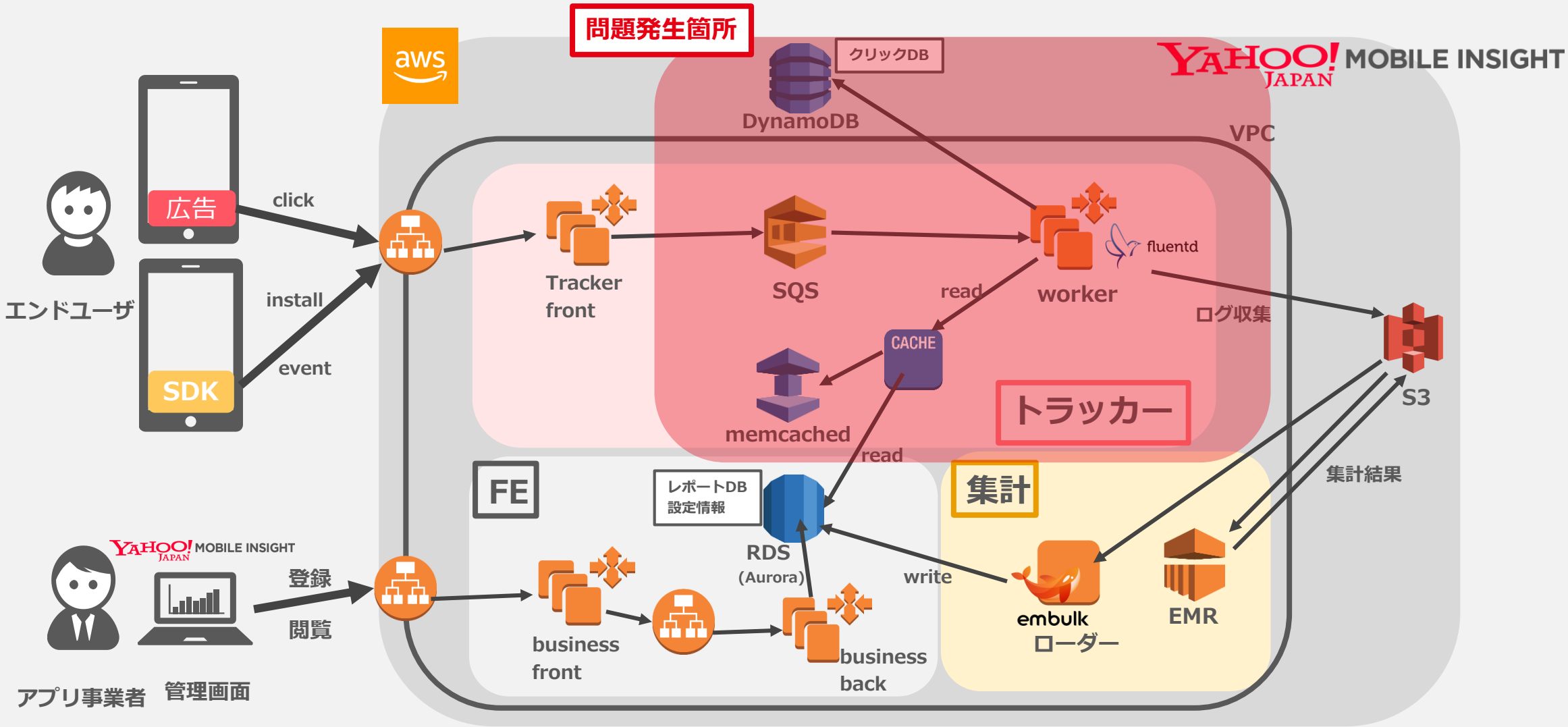
オートスケールで台数増やしても 総スループットがスケールしない問題

課題2

数分以内の急激なリクエスト増に オートスケールが間に合わない問題

現在進行形で取り組んでいる内容を以下でシェアします

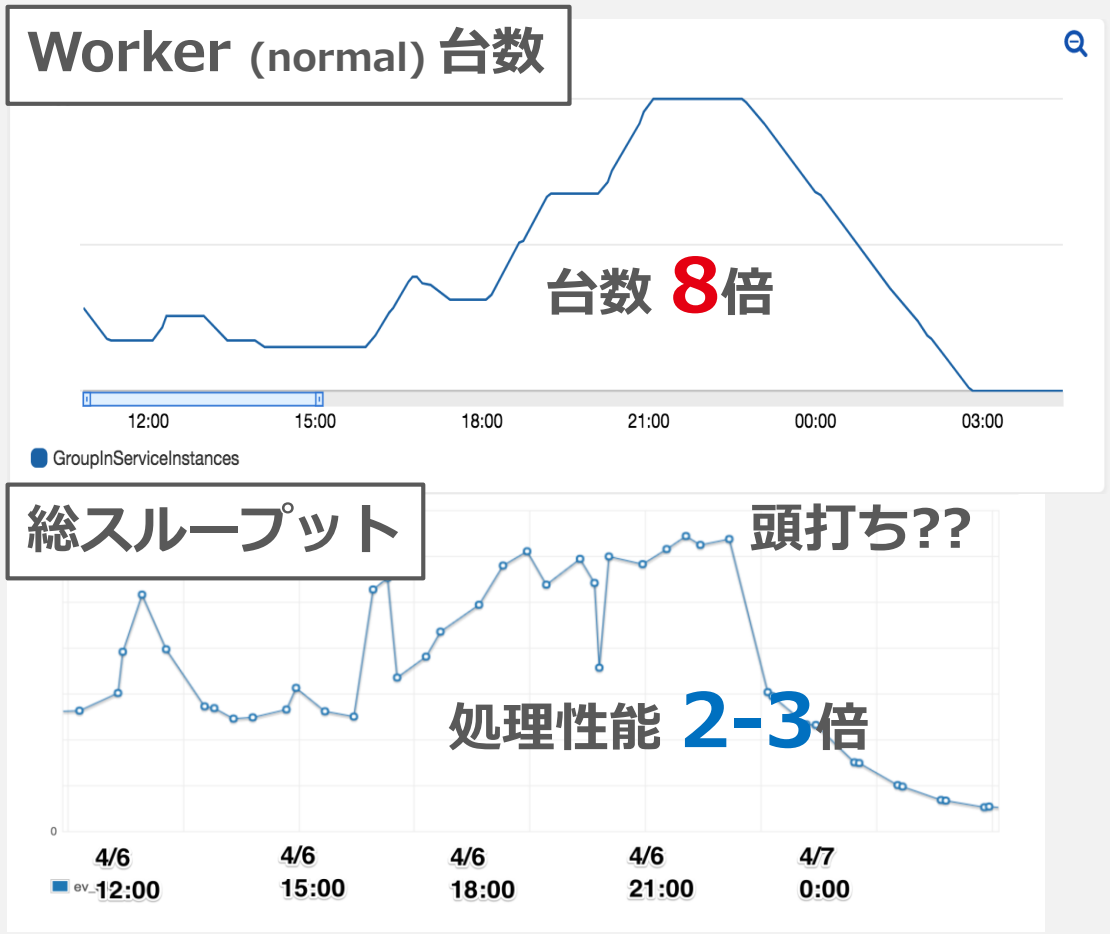
性能問題発生箇所はトラッカー(worker周り)



トラッカー(front)、集計、FEは今のところ性能問題発生していない

課題１： 総スループットがスケールしない

◆オートスケールで台数増えても システム全体の処理性能がスケールしてない!!!



ボトルネックが何処かに存在しているはず

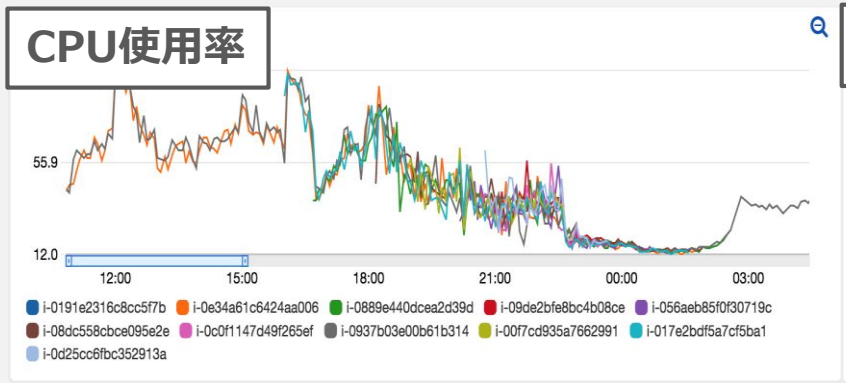
※総スループット = Worker(normal)が単位時間にキューから取得し処理完了したリクエスト数

ボトルネック調査

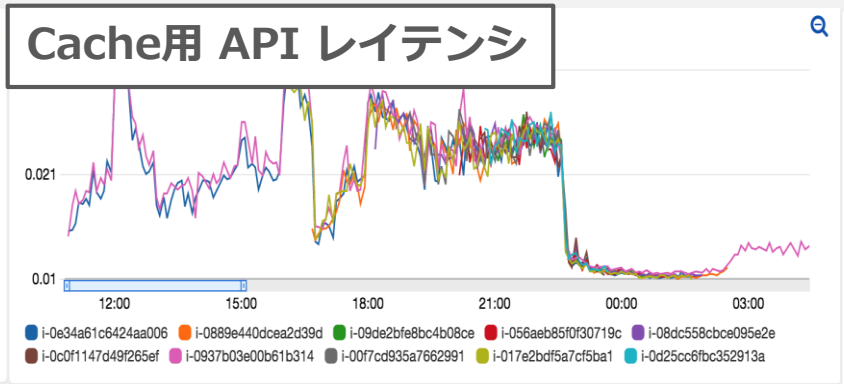
◆Cloud Watchを利用し、各種メトリクスを取得

AWSが提供するメトリクスだけでなくレイテンシ計測などカスタムメトリクス追加

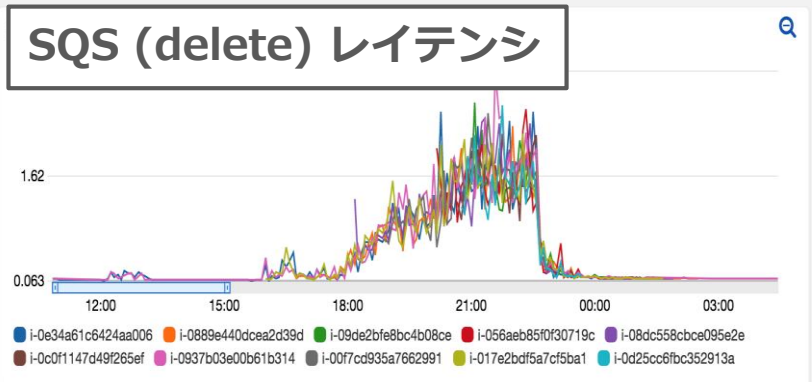
CPU使用率



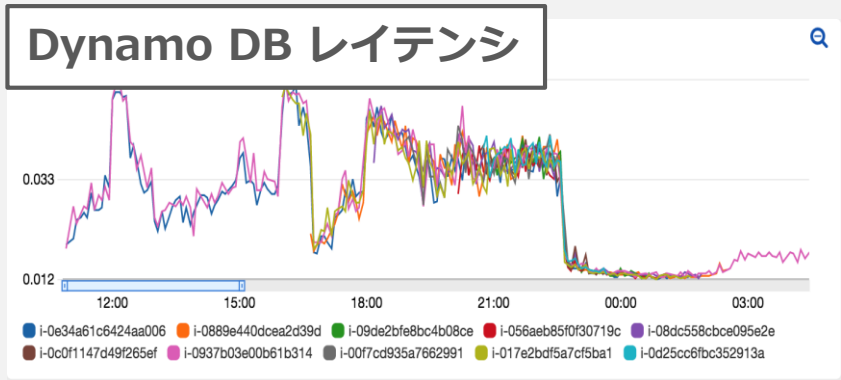
Cache用 API レイテンシ



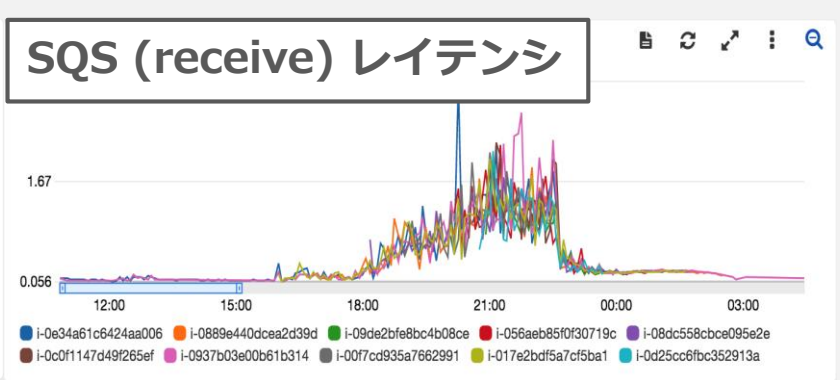
SQS (delete) レイテンシ



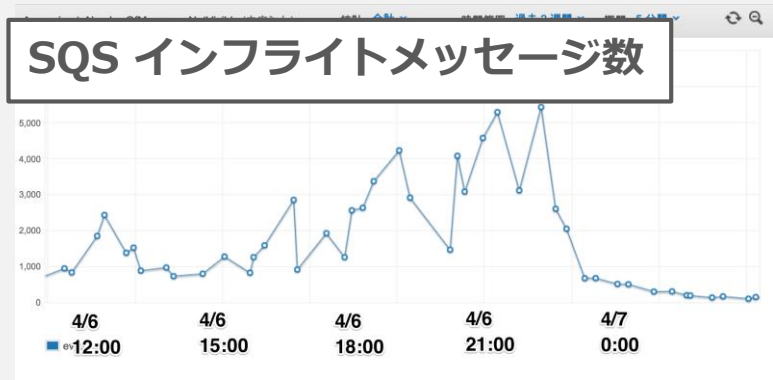
Dynamo DB レイテンシ



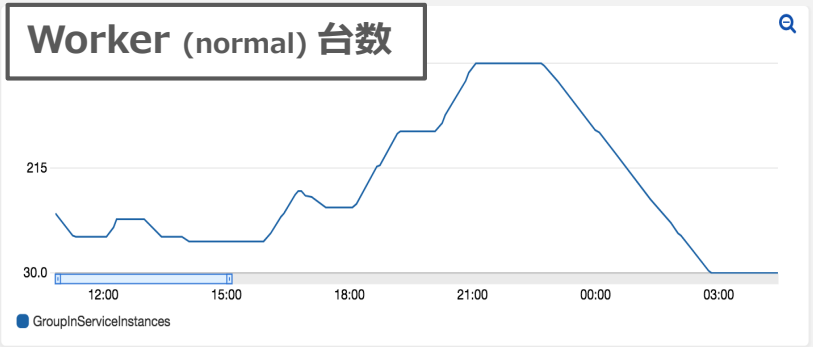
SQS (receive) レイテンシ

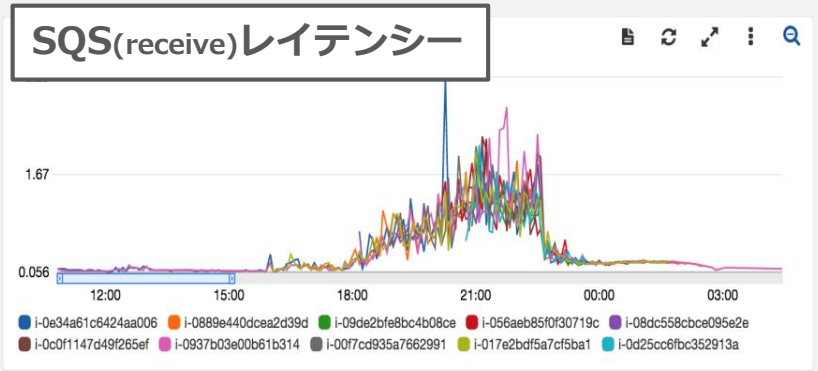


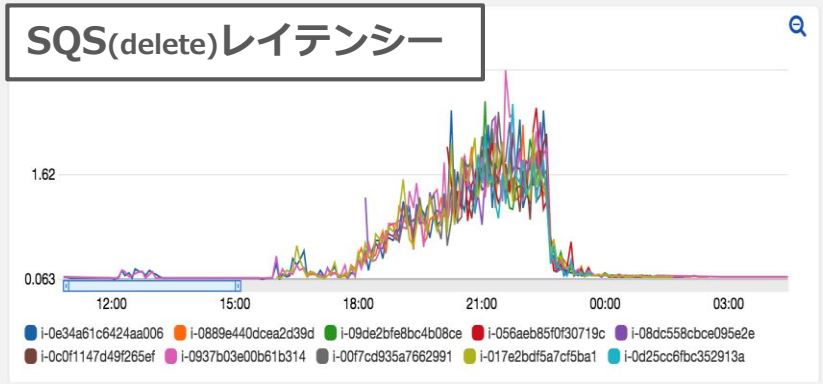
SQS インフライトメッセージ数



SQSがボトルネックだった







SQS (標準キュー) receive/delete レイテンシー

- 通常時　60ms以下
- リクエスト増大時　1000ms以上

1キュー辺りreceive/deleteの
スループットに性能限界が存在

SQSがボトルネックだった

スケーラビリティ

伸縮自在にコスト効率良くスケールする

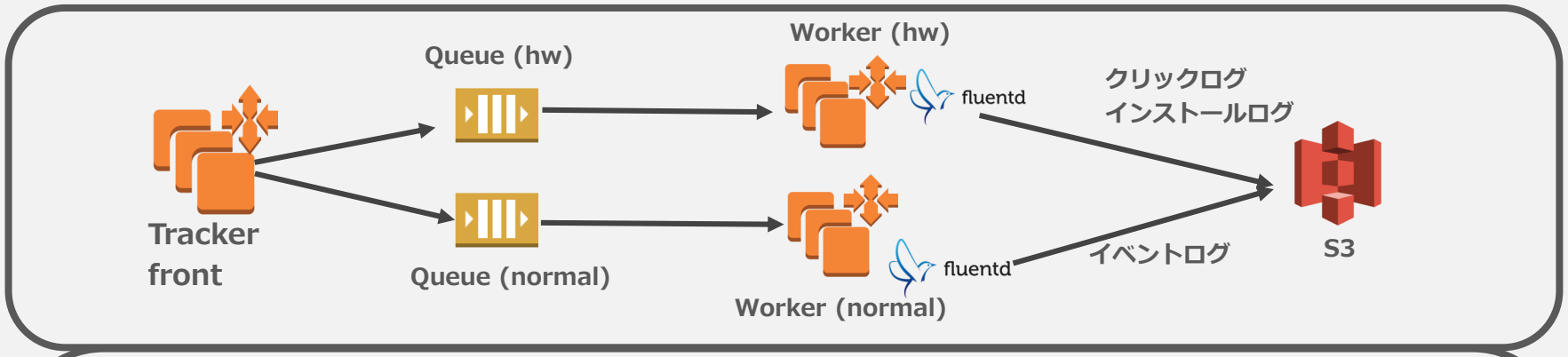
Amazon SQS では、AWS クラウドを活用して需要に基づいて動的にスケールできます。SQS はアプリケーションで伸縮自在にスケールされるため、容量計画や事前プロビジョニングについて心配する必要がありません。キューあたりのメッセージ数に制限はなく、標準キューではほぼ無制限のスループットが提供されます。使用量に基づいて課金されるため、自己管理型メッセージングミドルウェアを使用した "常時オン" モデルと比較して大幅な費用削減を実現できます。

AWS SQS公式ドキュメントより抜粋

クラウドと言えども 性能が無限にスケールするわけではない (当たり前)
サービスの性能限界を検証した上で利用したほうが良い

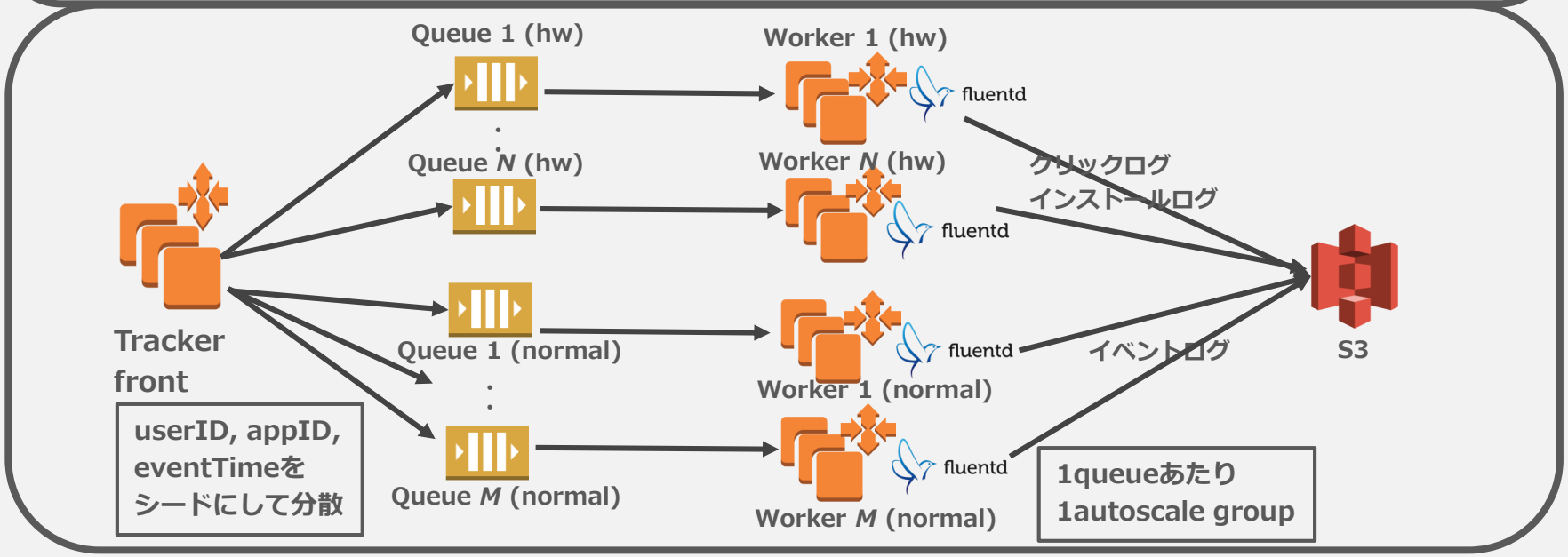
対策: Queueを横に並べることにした

対策前



対策後

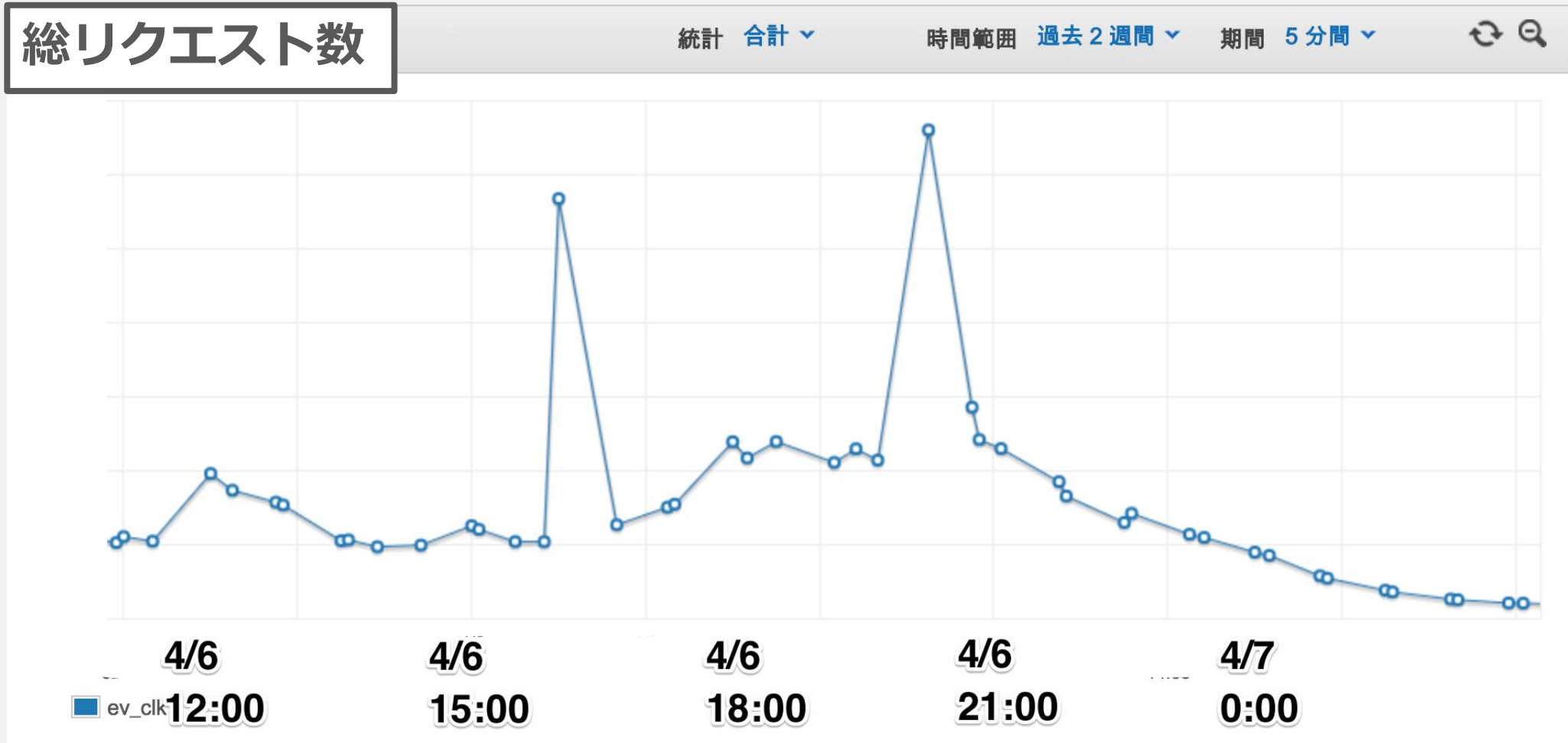
※開発中



※このやり方だと、キューの数をオートスケールできないなど運用面の課題
SQS をやめて Kinesis Stream利用も今後検討

課題2: 突発的にリクエストが急増する

総リクエスト数



数分以内にリクエスト数が平常時の**5-20倍以上急増**することがある

原因はヤフーアプリのPUSH通知

定期PUSH　朝刊・昼刊・夕刊 (それぞれ 8:00, 12:00, 18:00)

臨時PUSH　地震速報・豪雨速報などの災害系
重大ニュース系 (訃報、逮捕、解散など)

ここ数年で最大のリクエスト増は
SMAP解散

現状の課題

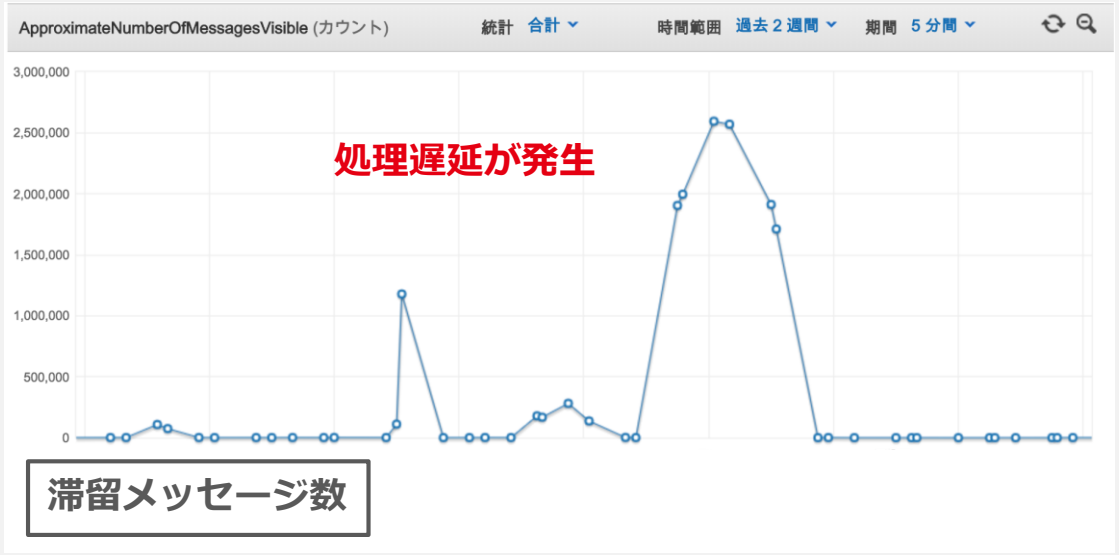
◆ オートスケールが間に合わず 処理遅延が発生

Cloud Watchが検知しオートスケール開始までラグがある

Golden AMI方式によるデプロイで起動まで最低1分前後かかる

◆ スケジュール実行によるスケールだけでは解決できない

臨時PUSHはいつ来るか基本予想不可



対策案

◆スケーリングポリシーの見直し・チューニング

スケールの立ち上がりをより高速に

◆コンテナ起動への切り替え (ECSの利用)

デプロイ速度の高速化

◆PUSHシステムとの連携

臨時PUSHを打った直後にスケール開始 ※そもそもできるか要調査

◆（最終手段）最初から急増想定した台数を並べておく

スポットインスタンス化などコスト面に配慮した上で実施。
※ただし スポット価格上昇による強制シャットダウン対策必須 (fluentd => Kinesis Firehose)

システム負荷対策まとめ

- 現状のシステム構成だと対処できない2つの性能問題
- 処理性能がスケールしない問題

  ボトルネック調査をCloud Watchメトリクスで実施
  原因はSQSの 1キューあたりのreceive/deleteスループットの性能限界
  キューを複数並べることで対処
- オートスケールが間に合わない問題

  突発的なリクエスト増の原因は ヤフーアプリのPUSH通知
  スケーリング速度改善や事前ウォームアップなど考えうる対策を現在検証中

アジェンダ

- ◆Yahoo! MOBILE INSIGHTの概要
- ◆AWS導入の経緯
- ◆現状のバックエンド構成
- ◆システム負荷対策の取り組み
- ◆オンプレとクラウドの共存
- ◆最後に

オンプレとクラウドの共存

オンプレ社内システム・データ と AWSを連携させる必要性

◆コード管理・CI/CD

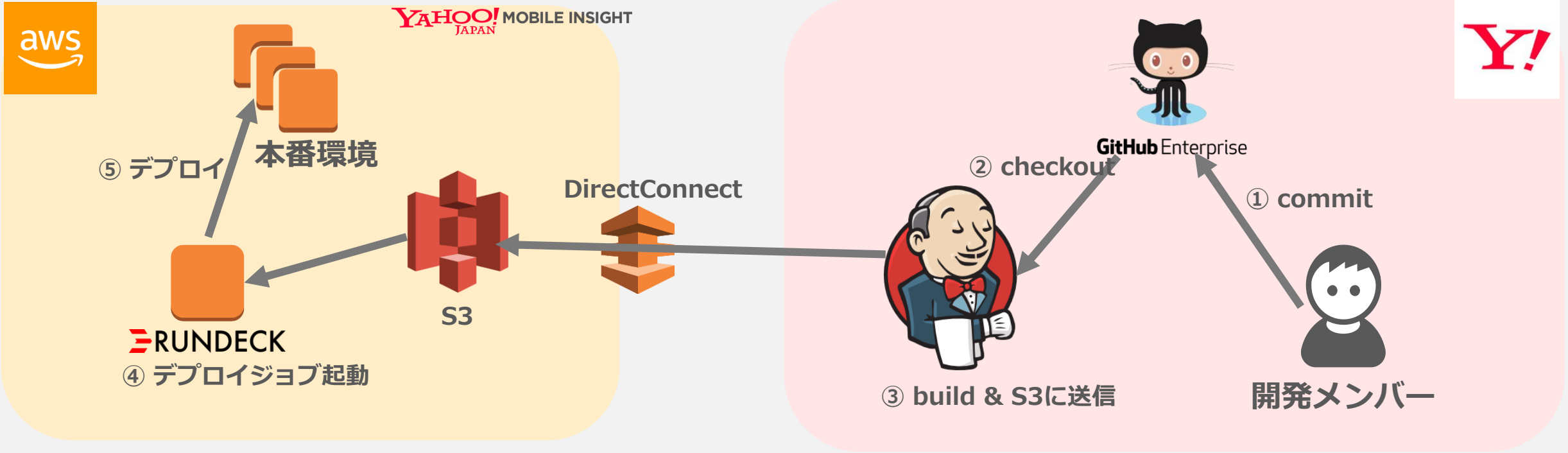
◆監視・アラート通知

◆ヤフーのデータ連携

オンプレとクラウド共存の取り組みの一例を以下でシェアします

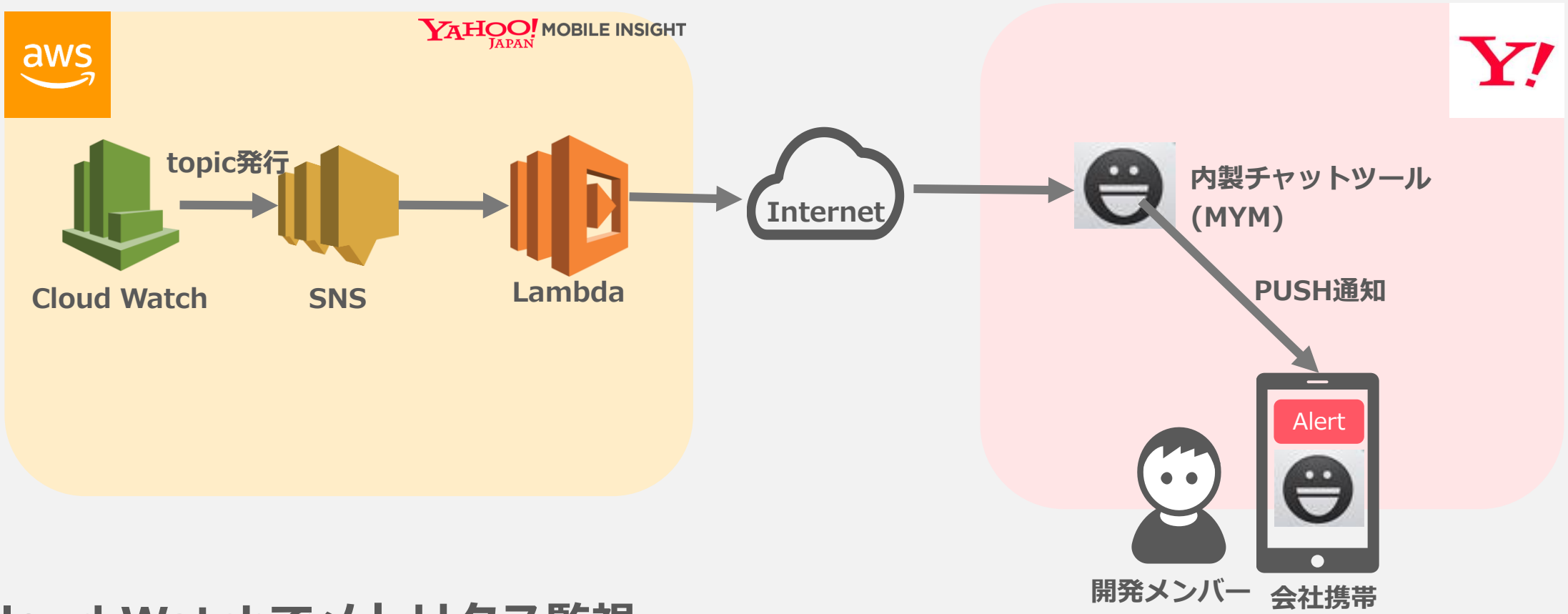
コード管理とCI/CD

前提： ヤフーで開発したコード資産は
オンプレのGitHub Enterpriseで管理している



オンプレJenkinsで一度ビルド&パッケージ化してから専用線経由でS3に送信.
その後、 AWS上に構築した Rundeckインスタンスのデプロイジョブ起動

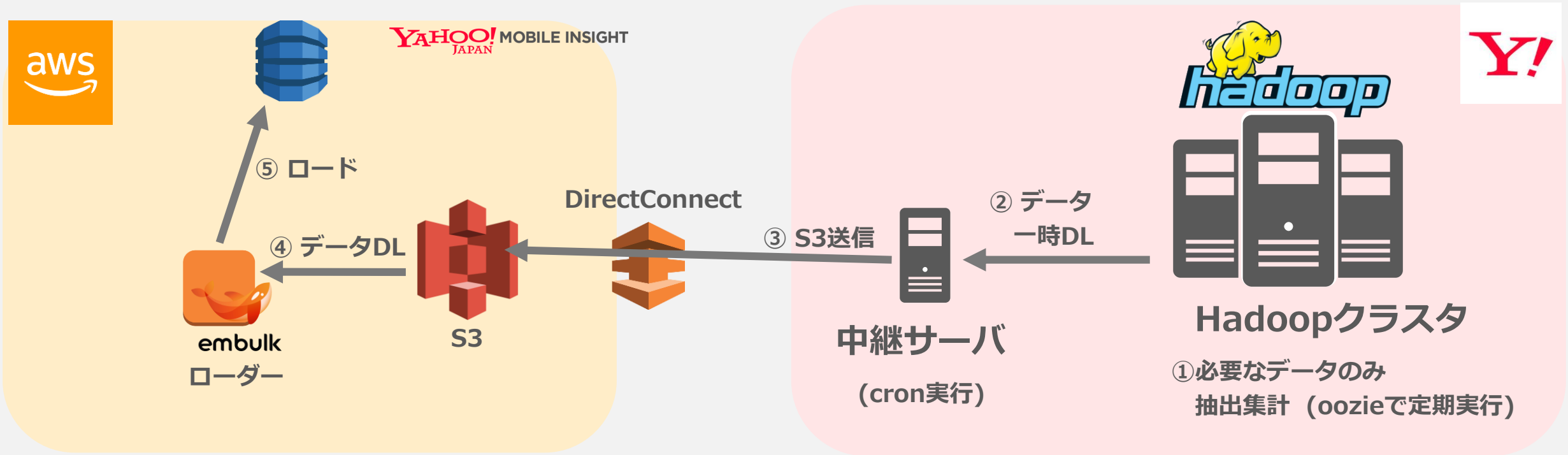
監視・アラート通知



Cloud Watchでメトリクス監視

アラート -> SNS -> Lambda -> 社内チャットツールAPI -> 会社携帯に通知

ヤフーサービスのデータ連携



オンプレのヤフーデータを必要なデータのみ定期抽出
専用線経由で AWS S3に送信し、MOBILE INSIGHTで活用

オンプレとクラウドの共存まとめ

◆コード管理・CI/CD

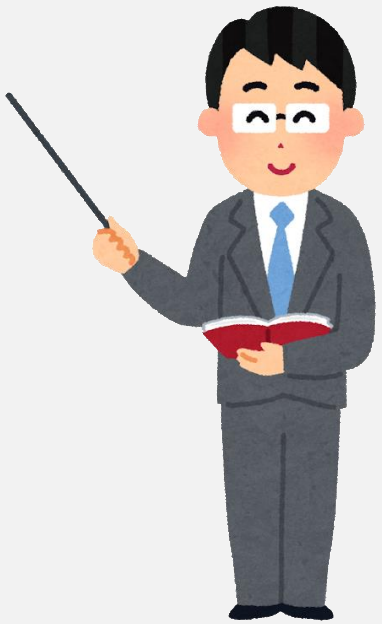
オンプレGitHub EnterpriseからJenkinsでビルド＆パッケージング
専用線経由でパッケージをAWS S3に送信
AWS側のRundeckインスタンスでデプロイ

◆監視・アラート通知

Cloud WatchからSNS, Lambda経由で社内内製チャットツールへ連携

◆ヤフーのデータ連携

オンプレのHadoopクラスタから必要なデータを定期抽出
専用線経由でAWSへ連携



アジェンダ

◆Yahoo! MOBILE INSIGHTの概要

◆AWS導入の経緯

◆現状のバックエンド構成

◆システム負荷対策の取り組み

◆オンプレとクラウドの共存

◆最後に

AWSを利用してみて個人的感想

GOOD

◆開発・検証が高速にできる

フルマネージドサービスの利用など検証が容易
開発メンバーの技術力や意識向上に繋がっている気がする

◆ドキュメントの検索が容易

ググ・・ヤフー検索すれば大体出て来る

◆オンプレとの共存もなんとかやれている

アカウント連携とかできていないことは多いが・・

BAD

◆公表されていない性能限界などトラップもある

鵜呑みにせず検証が必要

◆コストが割高(サーバー単価で対オンプレ比)

※ヤフーのオンプレは大量サーバ購入によりコストダウン

全体的に大満足

状況に応じてオンプレとAWS使い分けることで
今後もさらに活用していきたいです

ご清聴ありがとうございました

お申込みは サービスHPで受付中

(Yahoo! JAPAN ID で登録可能！)