いつもNAVI

地図情報を利用して解析する位置情報の「文脈」

株式会社ゼンリンデータコム　高山　敏典／鈴木順一郎

ZENRIN DataCom | Confidential and Proprietary

会社紹介

- 株式会社ゼンリンデータコム（略称　ZDC）
- 業務内容
  - 地図配信（API/SDK、地図切り出し、住宅地図関連）
  - ナビアプリ、カーナビアプリ（スマホ向け、車載向け）
  - その他コンシューマ向けアプリ
  - 店舗案内ASP
  - 動態管理ASP
  - ブラウザ向け地図サイト
  - 位置情報関連のシステム開発・運用委託
  - and more..

→位置情報サービス全般の提供

アジェンダ

- 位置情報サービスとは
- ZDCで行っている行動分析
- ZDCの行動分析の歴史
- マネージドサービスを活用した行動分析
- 質疑応答

位置情報サービスとは

位置情報サービス(LBS)とは

「位置情報サービス(いちじょうほうサービス、英: location-based service, LBS)とは、携帯機器などにより利用者が今いる位置を取得し、それに応じた情報を提供するサービスである。」

ウィキペディア「位置情報サービス」より引用

- 例）
  - 地図やナビのアプリ
  - 位置情報ゲーム
  - Siri
  - iphoneを探す

位置情報サービスの構成要素

いつもNAVI

地図
- タイル形式
- ベクター形式

行動分析
- 分析処理
- 統計処理

検索
- 施設検索
- 経路探索
- ジオコーディング

汎用
- 課金・認証
- 各種キャッシュ
- プッシュ通知

位置情報サービスの構成要素

いつもNAVI

- 地図
  - タイル形式
  - ベクター形式
- 行動分析
  - 分析処理
  - 統計処理
- 検索
  - 施設検索
  - 経路探索
  - ジオコーディング
- 汎用
  - 課金・認証
  - 各種キャッシュ
  - プッシュ通知

位置情報サービスの構成要素

いつもNAVI

地図

- タイル形式
- ベクター形式

行動分析

- 分析処理
- 統計処理

検索

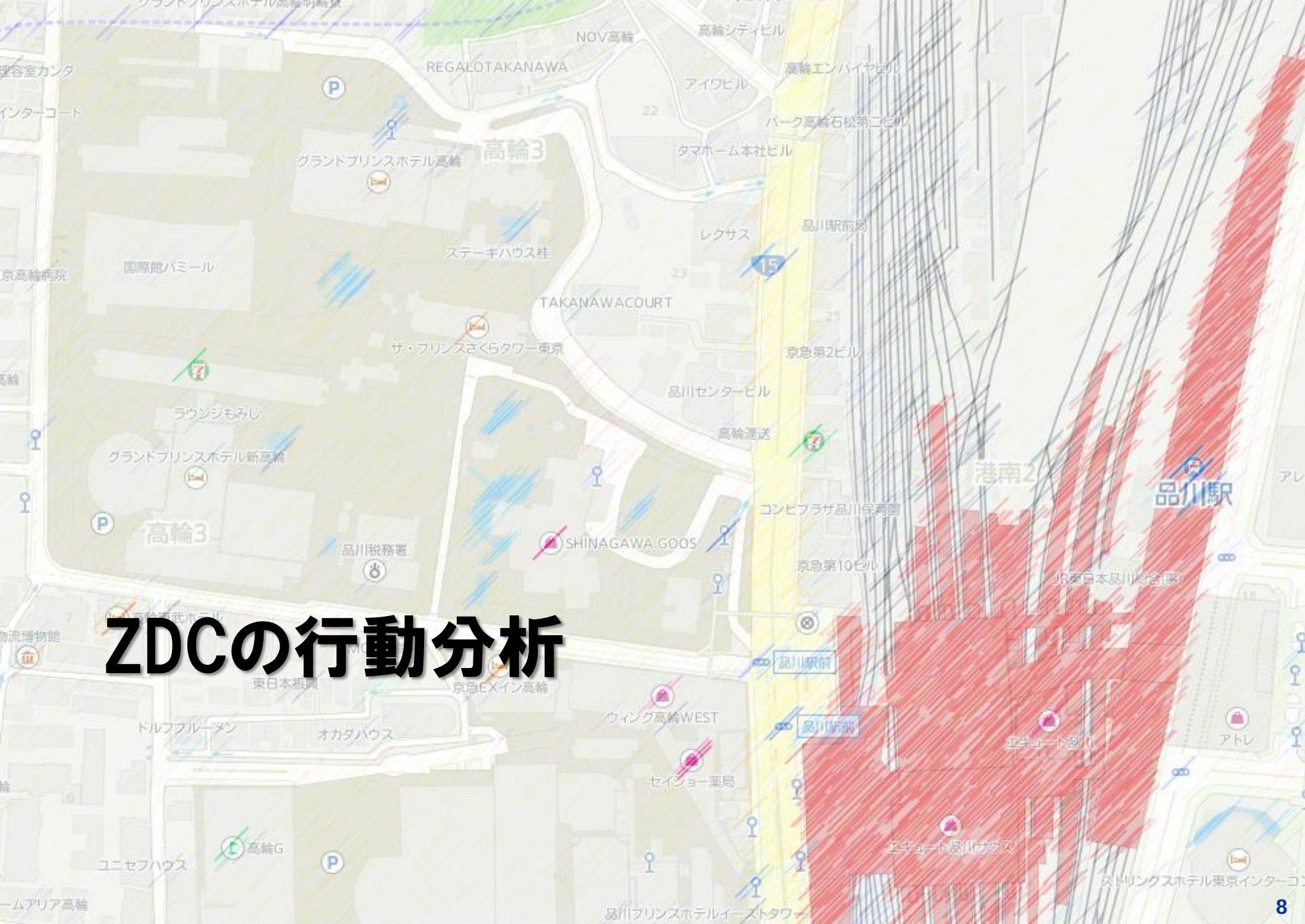
- 施設検索 

- 経路探索
- ジオコーディング 

汎用

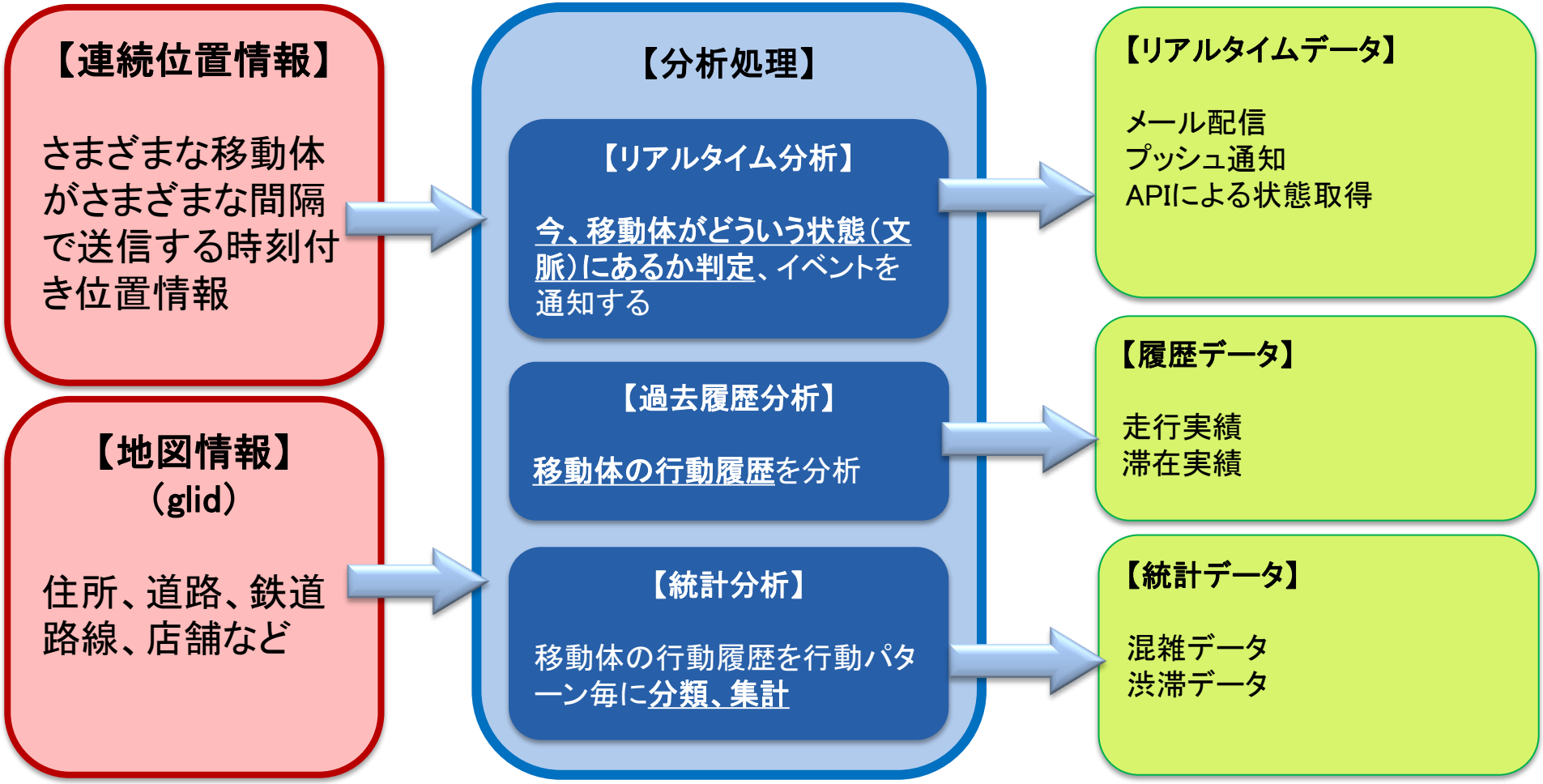
- 課金・認証
- 各種キャッシュ 

- プッシュ通知 


NOV高輪
高輪シティビル
REGALOTAKANAWA
高輪エンバイヤビル
アイワビル
パーク高輪石松第二ビル
高輪3
タマホーム本社ビル
グランドプリンスホテル高輪
品川駅前局
レクサス
ステーキハウス柱
国際館パミール
TAKANAWACOURT
サ・プリンスさくらタワー東京
京急第2ビル
品川センタービル
ラウンジもみじ
高輪運送
グランドプリンスホテル新高輪
港南2
品川駅
コンビプラザ品川保育園
高輪3
SHINAGAWA GOOS
品川税務署
京急第10ビル
品川駅前
東日本銀興
京急EXイン高輪
ウィング高輪WEST
ドルフブルーメン
オカダハウス
エキュート品川
アトレ
セイジョー薬局
高輪G
ユニセフハウス
エキュート品川サウス
品川プリンスホテルイーストタワー

ZDCの行動分析

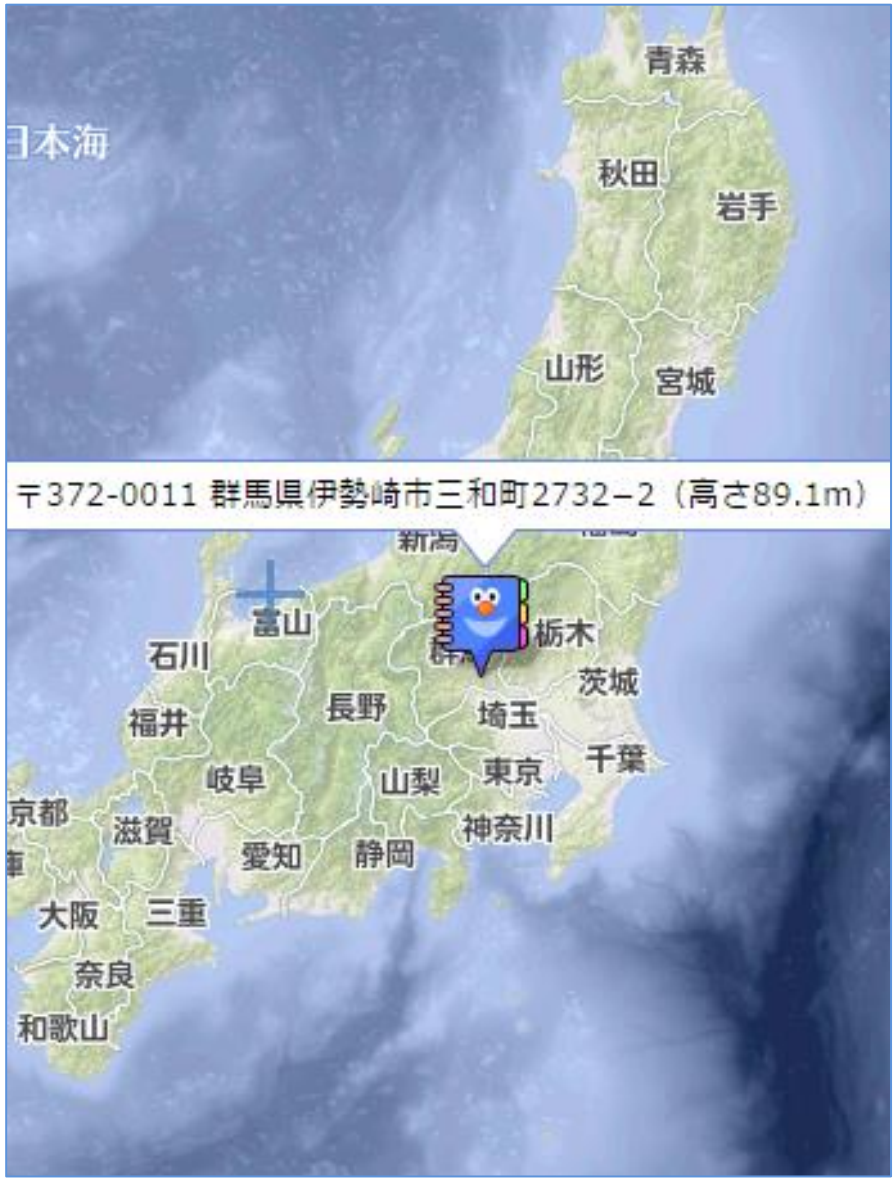
ZDCの行動分析の概要

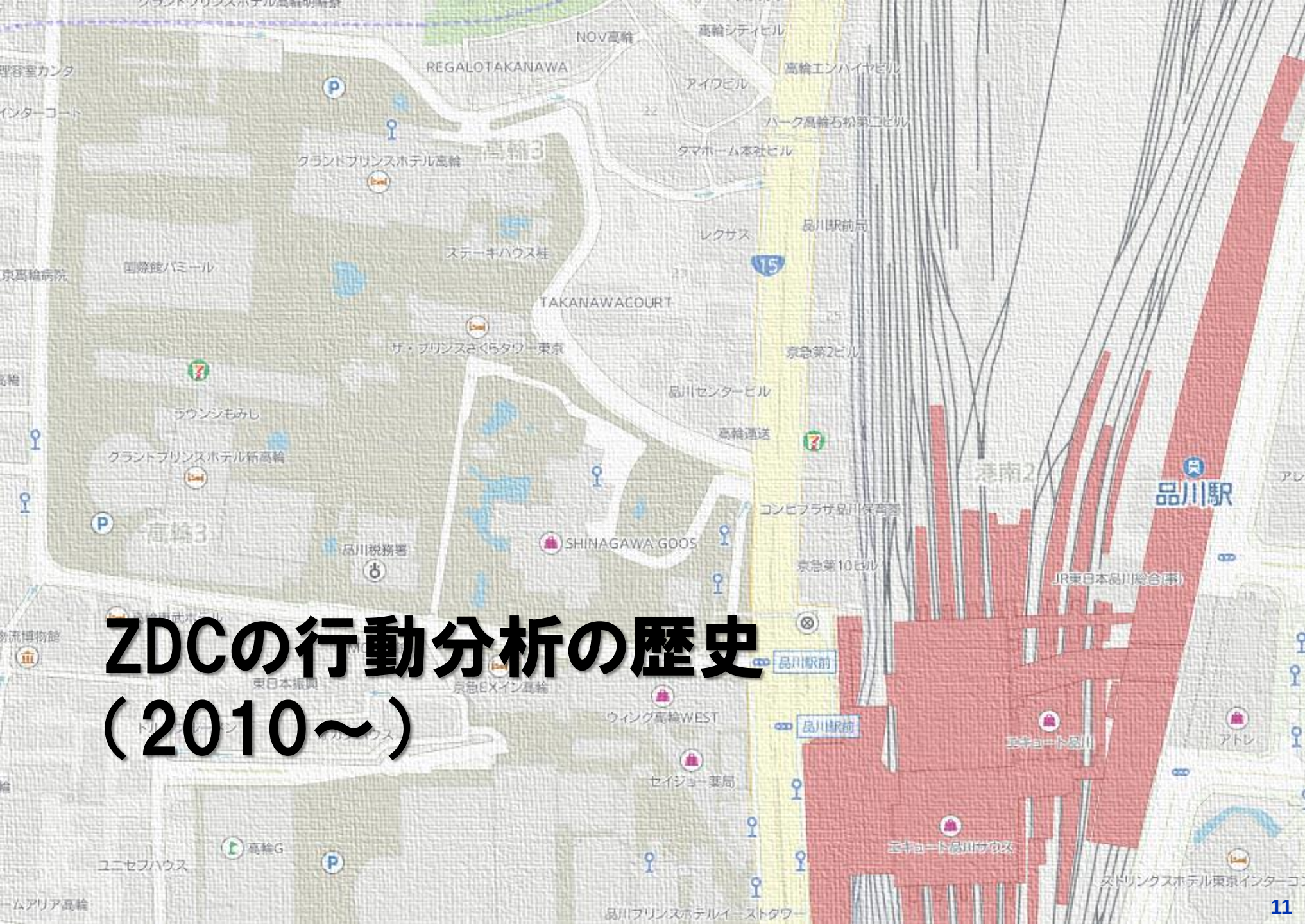


※「統計データ」とは、利用許諾を得た上で送信される位置情報を、委託により当社が個人が特定されないよう集計・処理したものです

glidって？

- glid（グリッド）
  - 汎用逆ジオコーダ
    - 緯度経度→地図データ
    - KVS的なもの
  - さまざまなデータを取得可能
    - 付近の施設
    - 郵便番号
    - 最寄り駅
    - 住所
    - 付近の道路、鉄道路線
  - 実装事例
    - 地図から住所検索
      http://lab.its-mo.com/glid-addr/



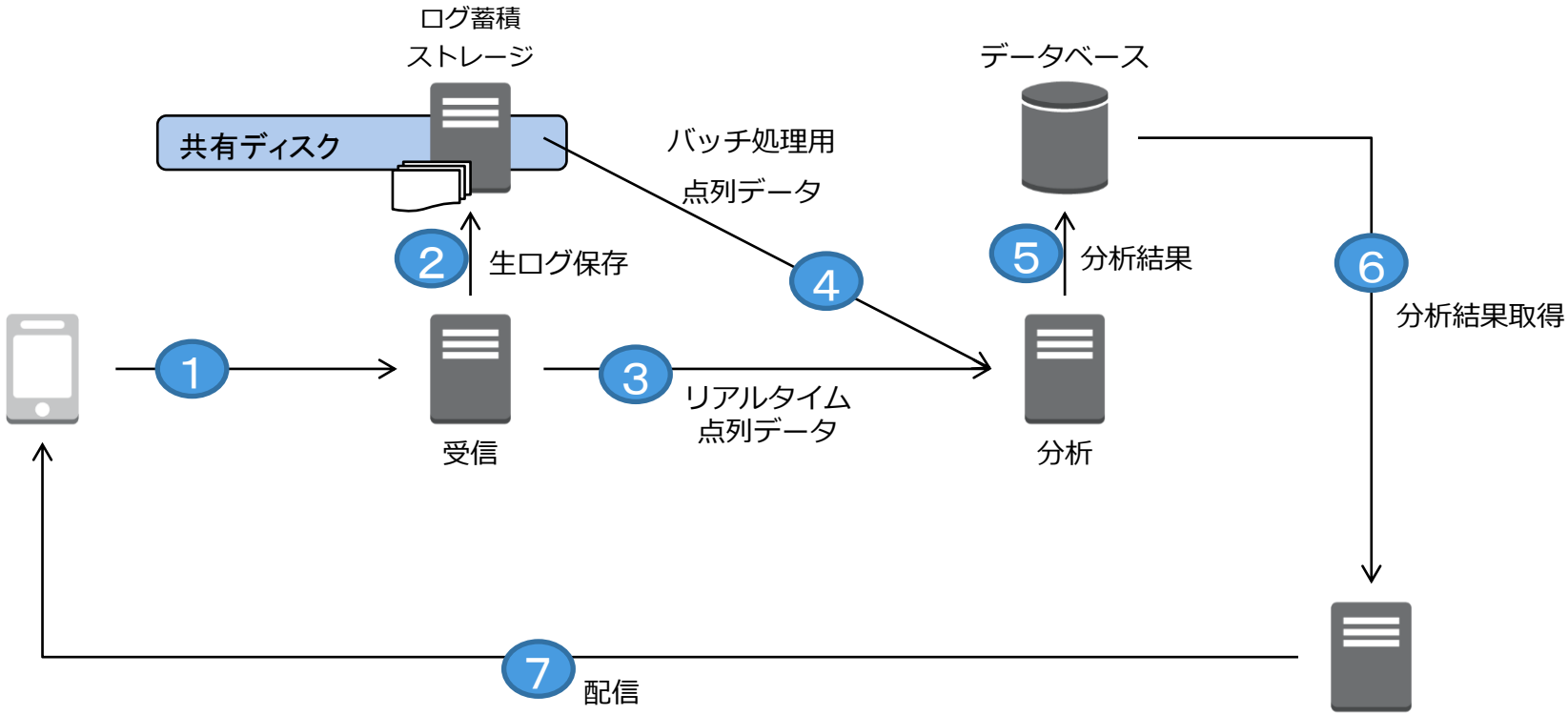


ZDCの行動分析の歴史（2010～）

第1世代　オンプレで自前システム(2010ー)

構成

- 測位点列を受信してID毎に共有ストレージに保管(ファイルベース)
- 分析結果をRDBMSに格納
- 分析結果を集計、加工



第1世代　オンプレで自前システム　課題

いつもNAVI

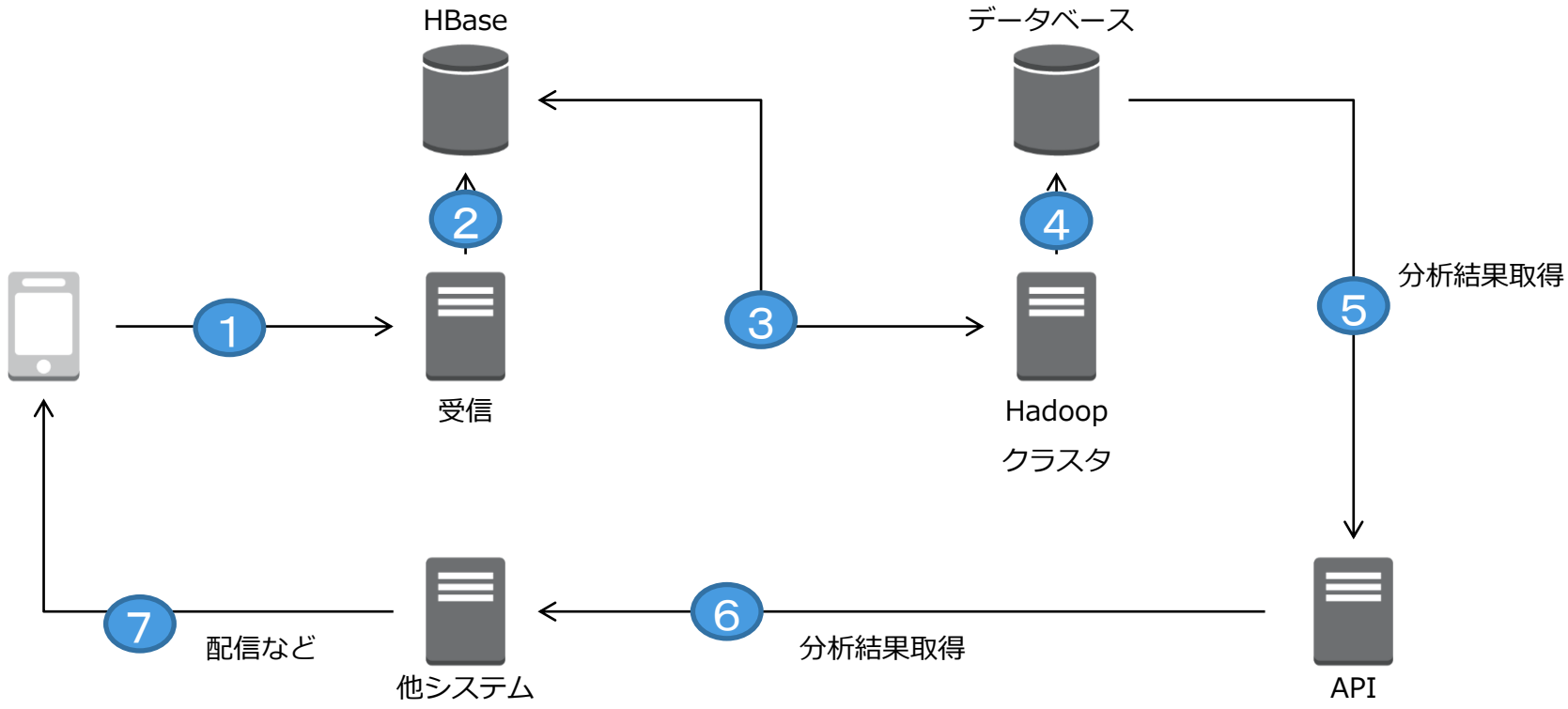
- システム性能を向上させるための時間をかけてた
  - ファイルシステムあれこれ変えたりinode調整したり
  - 並列化、冗長化が難しい(共有ディスクって難しい)
  - Cで書いたマルチスレッドプログラムで分析処理してたが並列化に限界があった(バグの原因が特定しづらい)
- ハードウェア障害の原因特定に時間をかけてた
  - 共有ディスクのベンダに問い合わせしたり、いろんなパターンでテストしたり。いまとなってはよく覚えてない

ボトルネックの解消や障害対策に時間がかかっていて、ロジック開発に専念できず。

第2世代　オンプレでHadoop（2011-）

構成

- 測位点列を受信してID毎に共有ストレージに保管（hbase）
- バッチ実行の分析結果をRDBMS、KVSに格納
- 分析結果を集計、加工



第2世代　オンプレでHadoop　課題

いつもNAVI

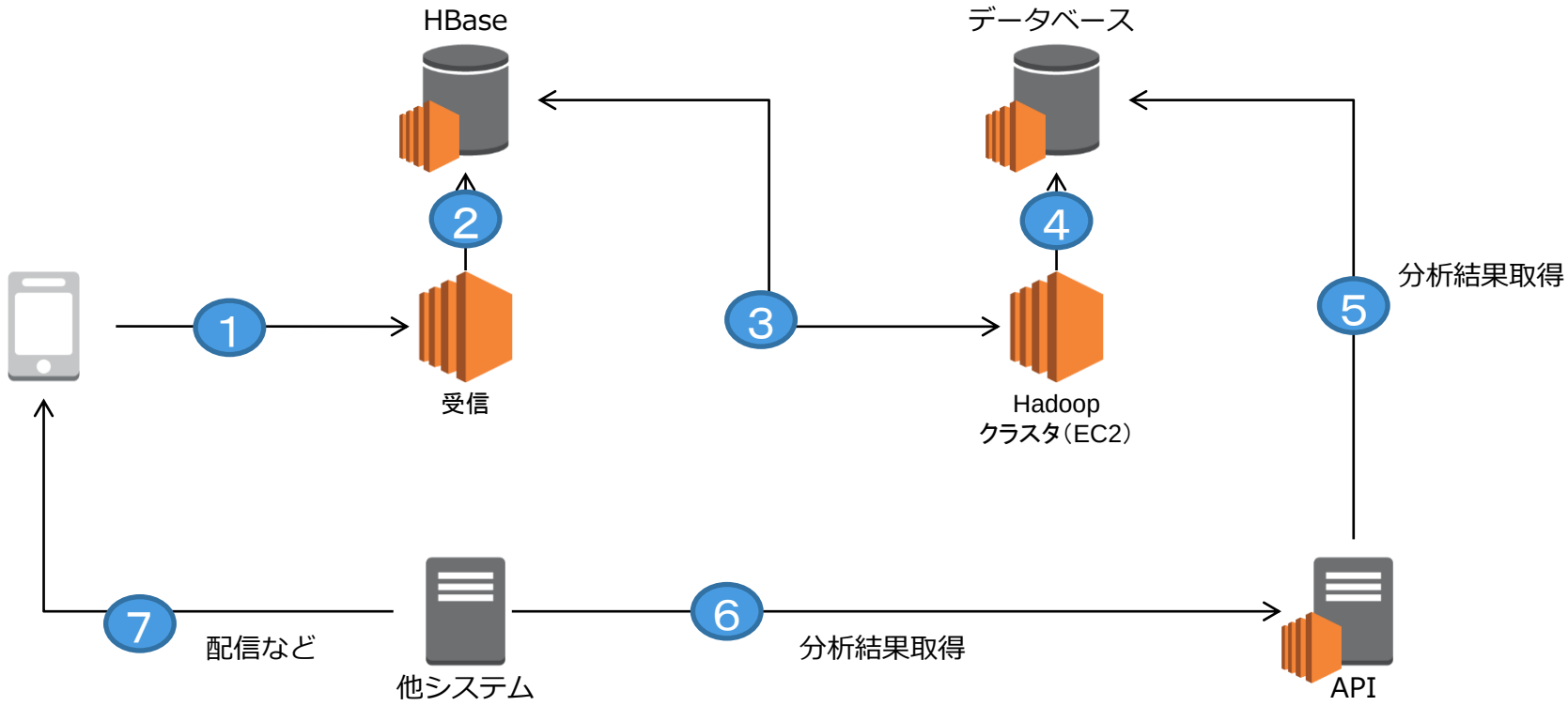
- 並列化部分は楽になった
  - Hadoopに丸投げできるようになった
  - 台数を増やすことがプログラムを変えずにできるようになった
- 人よりも計算機のスケジュール管理が大変だった時代
  - マシンの手配がつかないために依頼を断ることも
  - 仕事の依頼があってからサーバ発注しても間に合わない
  - 故障したせいでデータ生成スケジュールがリスケに
  - 暇なときはまったく稼働しない日もあった

計算資源の調達や運用が非効率。分散システムの運用の難しさに直面。

第3世代　とりあえずクラウド（2012-）

構成

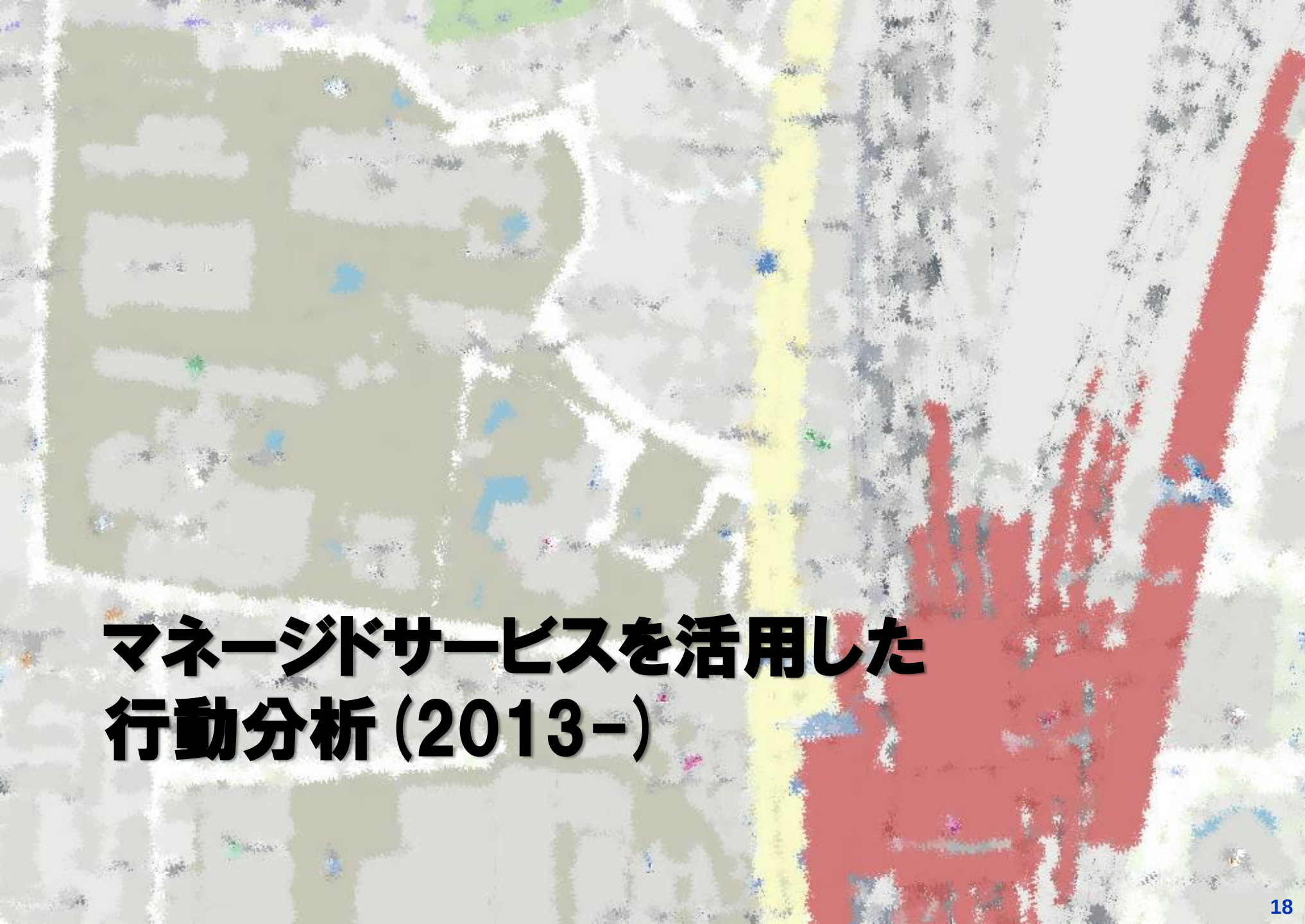
- 第2世代の構成のままAWSに移行



第3世代　とりあえずクラウド　課題

- サーバ調達は楽になった
  - 移行はスムーズだった（サーバをec2に変更するだけ）
  - サーバの納期は早くなった
- そのまま移行するだけではコスト的にメリットなし
  - EC2の利用時間を限定したりリザーブドインスタンス契約をしなければコスト削減にはならない。Hadoopクラスタについてはいつ必要になるかわからないので台数固定は変わらず。
- AWSに関する知識不足に因る問題
  - ELBと既存のロードバランサの微妙な違いにつまづく等

AWSの恩恵は受けられつつあるが、第2世代の課題解決には至らず。



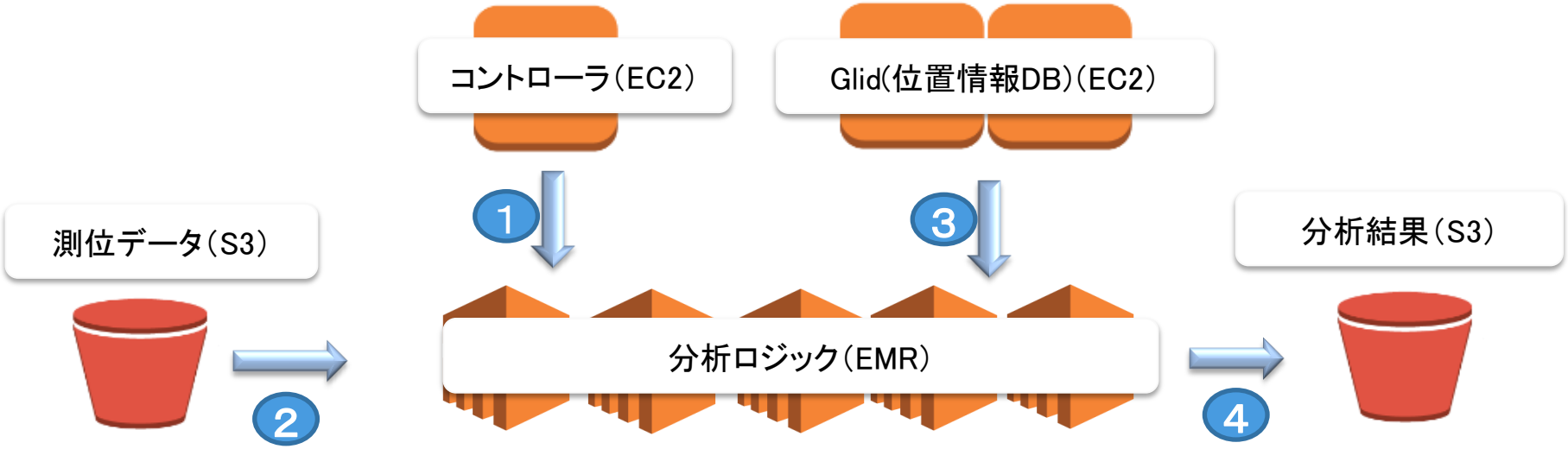
マネージドサービスを活用した行動分析 (2013-)

第4世代　マネージドサービスを活用した行動分析

基本方針

- とにかく疎結合
  - 複雑性を小さくするために、とにかく単体のシステムを小さく少なくする。
- 使えるものを使い倒す（ EMR、SQS、 DynamoDB、SNS ）
  - 適応範囲の広い厳選したものをとことん使う。
- 実行時間は太く短くメリハリつけて
  - どんなバッチ処理も2時間以内を目途に台数を調整。
  - X2large1台よりもXlarge2台で増減させる。
- 細かいことは気にしない
  - エラーがでても最小単位で再実行。欠損データでストップしない。
  - 単体の性能向上よりも並列化が実現できてればOK。

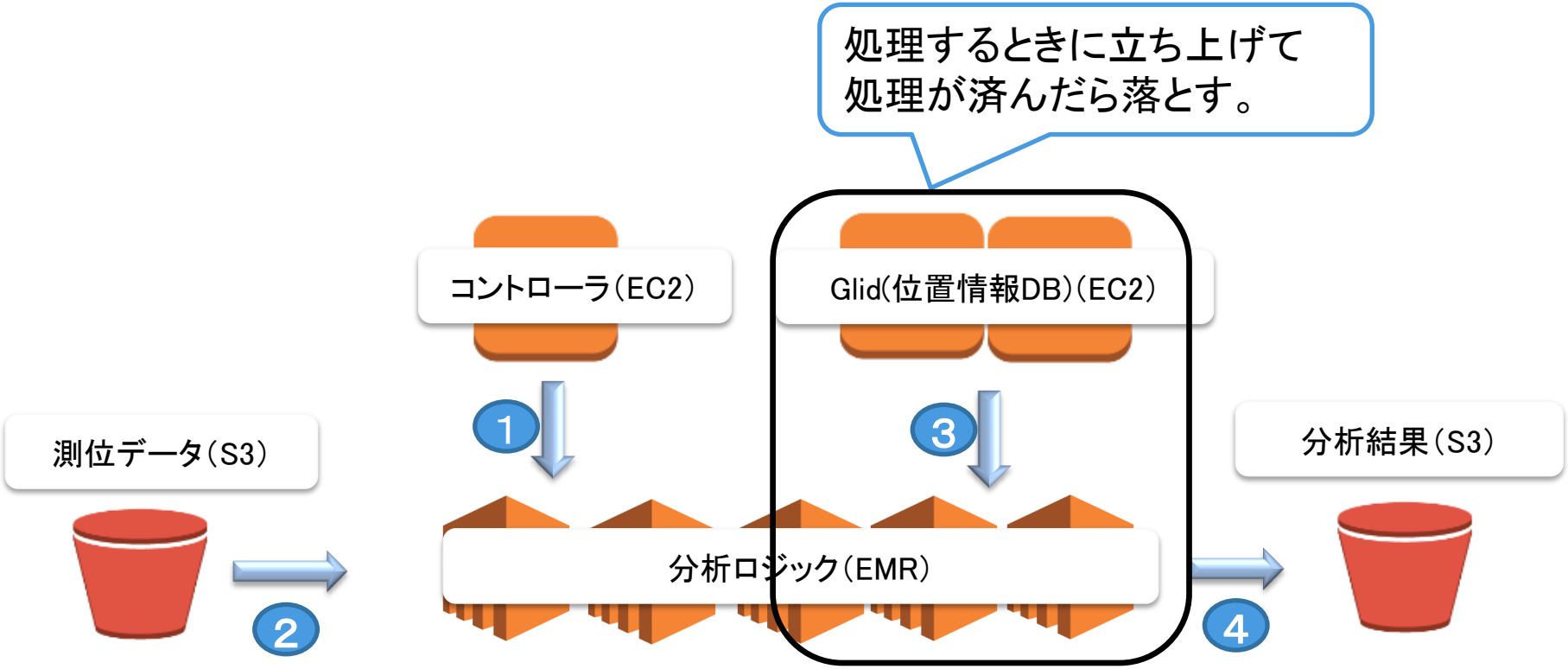
1、EMRのメリット（高速化）



12台で20時間が120台で2時間で終了！

平日の時間帯でも処理がながせるようになった。

1、EMRのメリット(トータルで安い！)

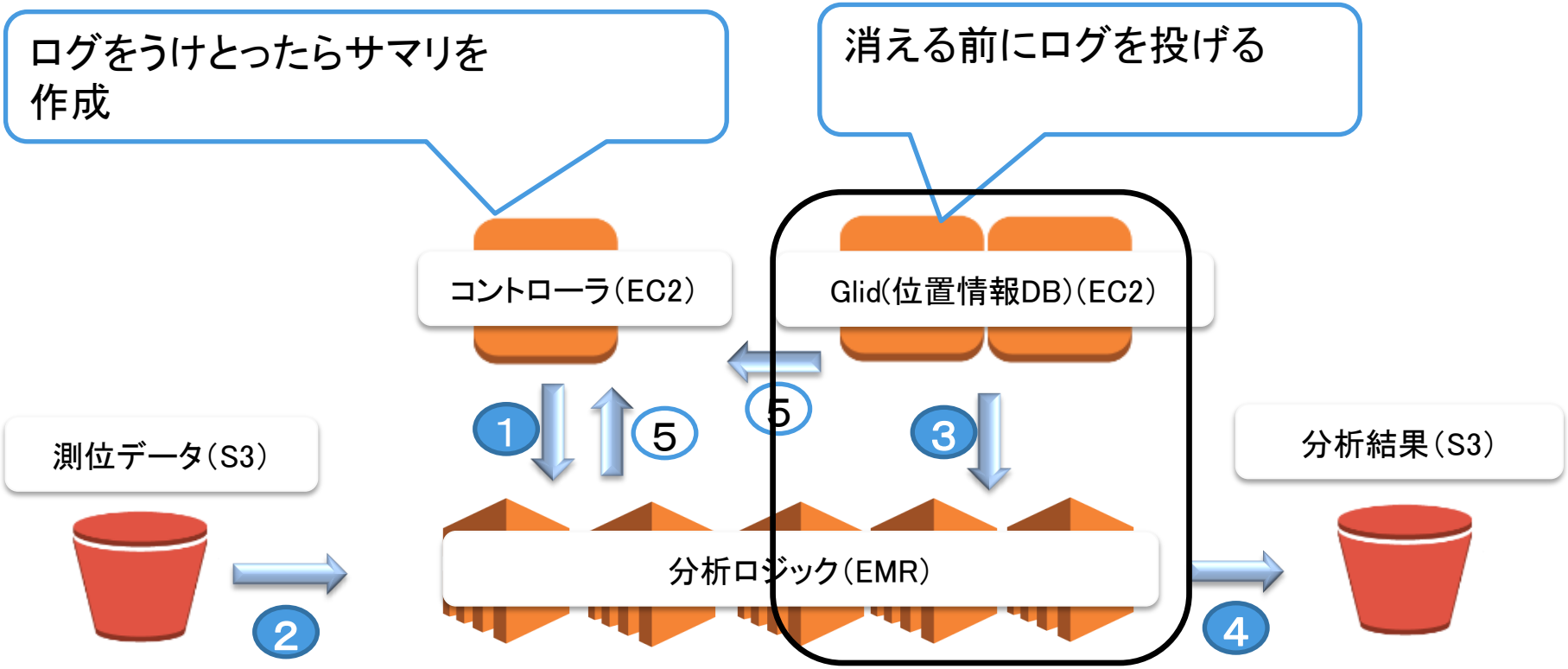


120台使っても2時間で終われば3万円！

処理が終了したら使ったものはきちんと落としておく(落とさないと週末で80万円くらい損する！)。
時間がかかるとそれだけ損するので処理の高速化に気を使うようになった。

2、EMRで気を付けること(ログの回収)

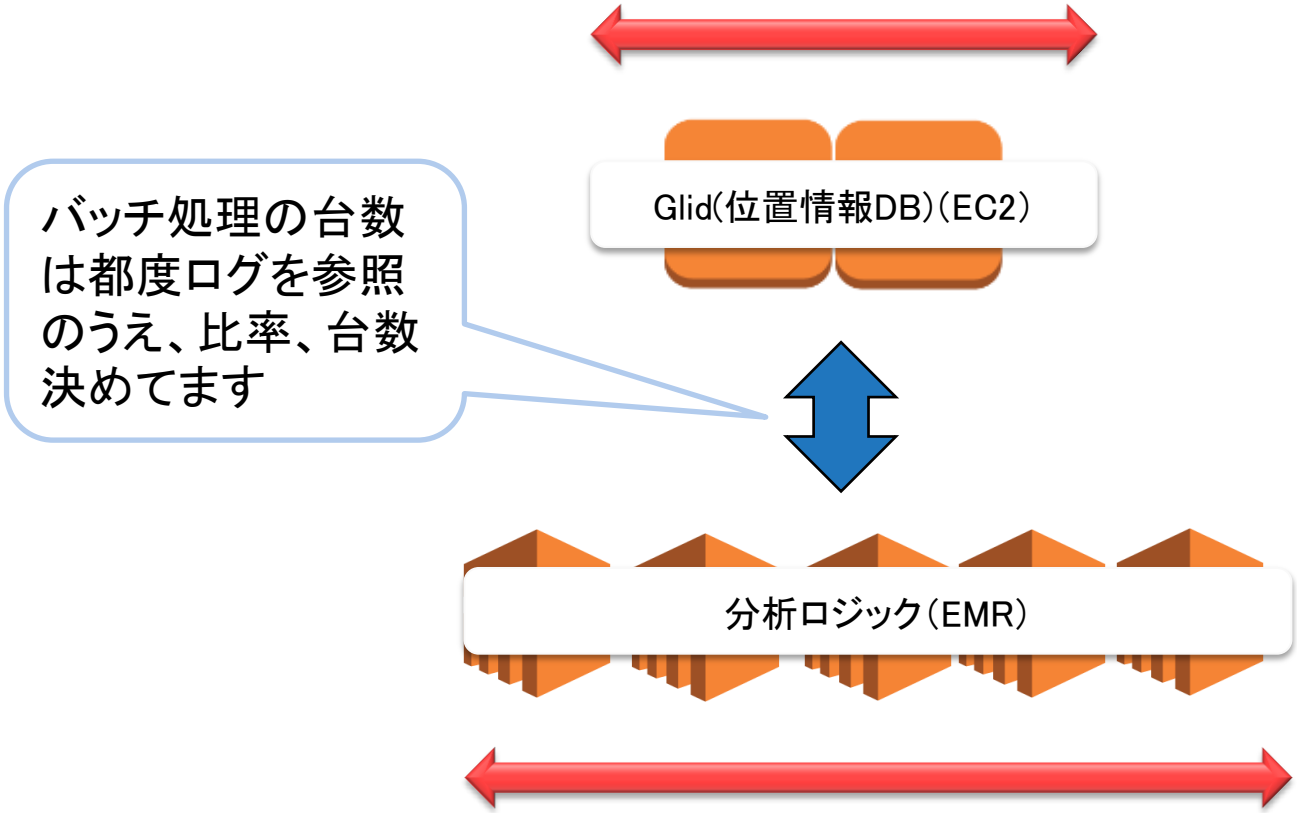
いつもNAVI



ログの回収は忘れずに

マシンを落としてしまえばログは消えてしまいます。なぜ失敗したのか、なぜ実行時間が余分にかかったのかがわからなければ余分に費用が掛かることになります。一台ごとのログと全体サマリどっちも重要です。

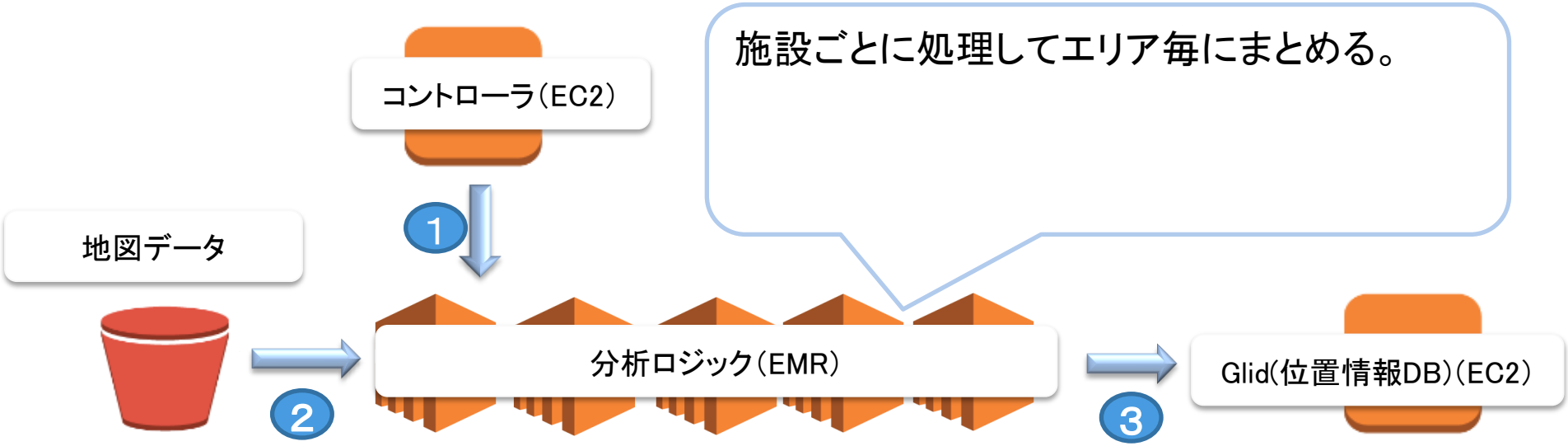
2、EMRで気を付けること（ボトルネックの把握）



ボトルネックを常に把握

バッチ処理に関わっている各サーバでCPUが忙しいのかメモリがいっぱいなのか、DBが忙しいのか分析サーバが忙しいのか。データや処理の追加や削除、分析対象の変化によって常に変化しています。処理時間が伸びてきたらログで確認して台数の比率やm/cのタイプを変更しています。

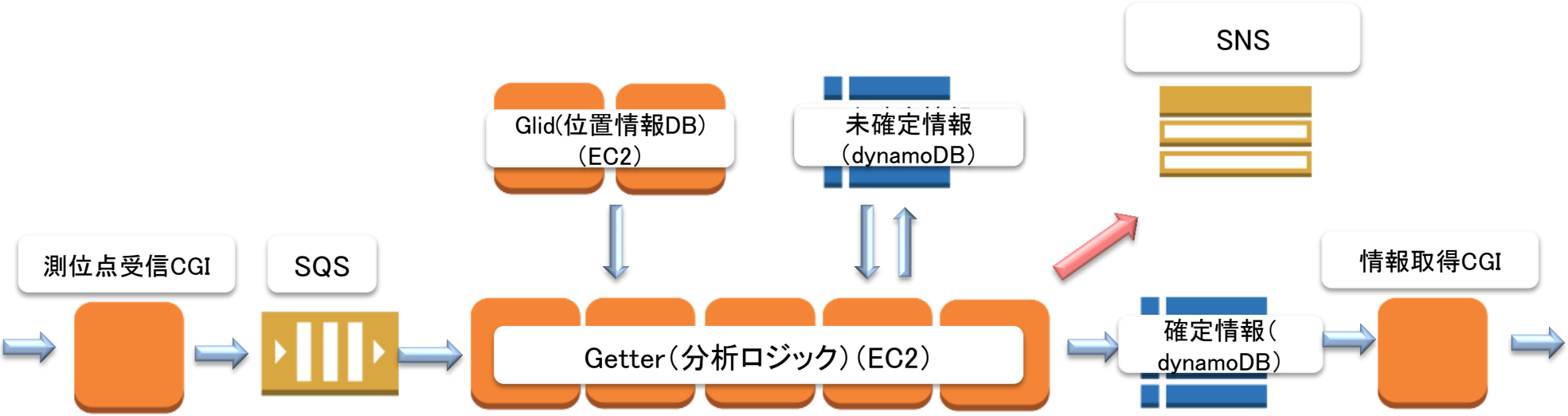
3、EMR処理の活用（Glidデータ生成）



マスタデータの編集もEMRへ

領域の確定部分をmap,重なり部分の処理を一部reduceに割り振ることで並列化を実現しています。
実行時間が一桁以上削減する見込みです。

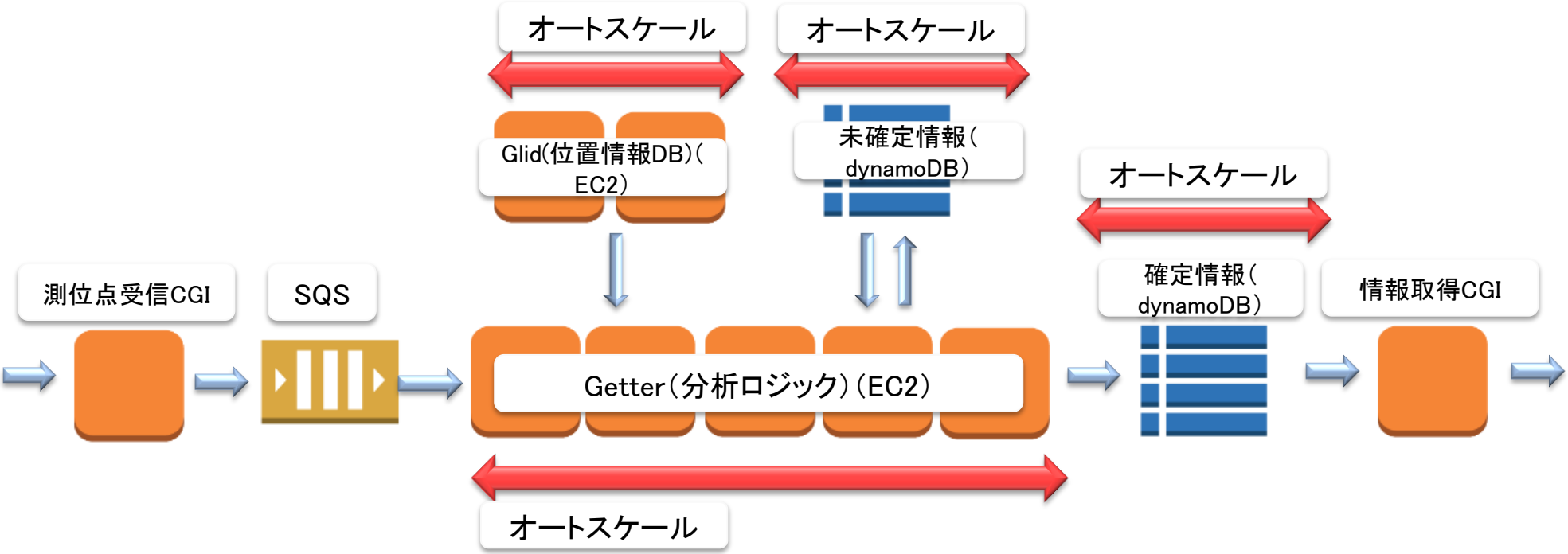
4、SQS+dynamoDBでリアルタイム分析(ロジックの共有)



分析ロジックはEMRと同一　イベント通知はSNS

EMRでは入力データが確定していないと処理がはじめられません。リアルタイムでデータ処理をするためにSQSをつかった並列処理システムを構築しています。分析ロジックはglidへの問い合わせ含めてEMRと同一です。イベント発生時はSNS経由で別システムへ通知されます。

5、SQS+dynamoDBで最適化（オートスケール＋ログ分析）



処理量に応じて台数を最適化

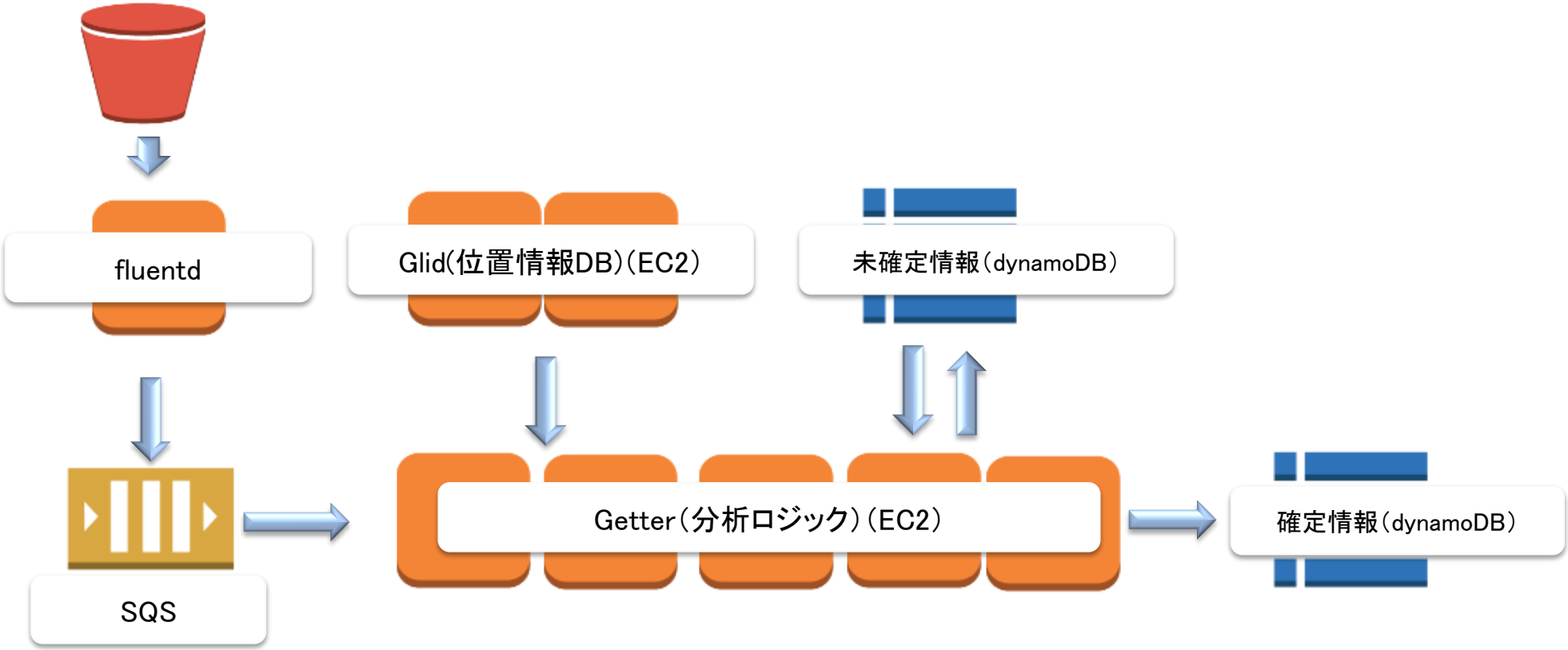
SQSで待ってる処理量や負荷に応じて処理能力を変えてます。
SQSがボトルネックになったり障害発生点になったことはない（SQS最高）。
ログの確認による手動での調整も必要です。

6、SQS+dynamoDBで便利なこと(再実行)

必要があれば指定時間後に再実行

Delay seconds を設定して再投入することで任意の秒数後にSQSからレコードを受け取ることができる。
(Visibility timeoutだと転送中メッセージ数に12000の上限がある。)
ID毎にタイマーをgetterサーバ側で設定するなんて不可能。

6、SQS+dynamoDBで便利なこと(バッチ処理も可能)



バッチ処理も実行可能

過去データをfluentd経由で流し込むことでバッチ処理的な処理も実行することができます。ただし、未確定状態のデータをどう処理するかは決めておく必要があります。

まとめ

現状の評価

- 柔軟な計算資源調達が可能になった
  - システム側の都合でビジネスを止めずに済む
  - 需要予測が難しい要件でも最小コストで対応可能に
- ロジック開発に集中できるようになった
  - チューニングの対象がビジネスロジック中心になった。
  - ハードウェア、ミドルウェアはAWSにまかせる。

今後の展望

- 性能を最適化してコスト削減
  - スポットインスタンス、リザーブドインスタンス
  - レッドシフト、ラムダなどの活用検討

- より高度な分析
  - 統計データのリアルタイム生成
  - 未来予測
  - 特異現象発生の自動検出

ご清聴ありがとうございました