課題3

設計説明書

児玉　新次

- 課題のキモは、リクエスト数が多いこと
  - アクティブユーザ数 : 20,000/h（要求事項より）
  - リクエスト数: 16,666/m
    （平均5回/hのアクセスが均等に行われるとして、20,000 × 5 ÷ 60）
  - 合計レコード数: 2,400,000/d
    （20,000 × 5 × 24）

- 収集したデータは折角なので使い切りじゃなく再利用したい
  - S3 + Redshift spectrumを選択

- リクエストをもらさず遅いメディア(S3)に書き込む必要あり
  - 段階的にリクエストをまとめる処理を行う

段階的にリクエストをまとめる処理の中身

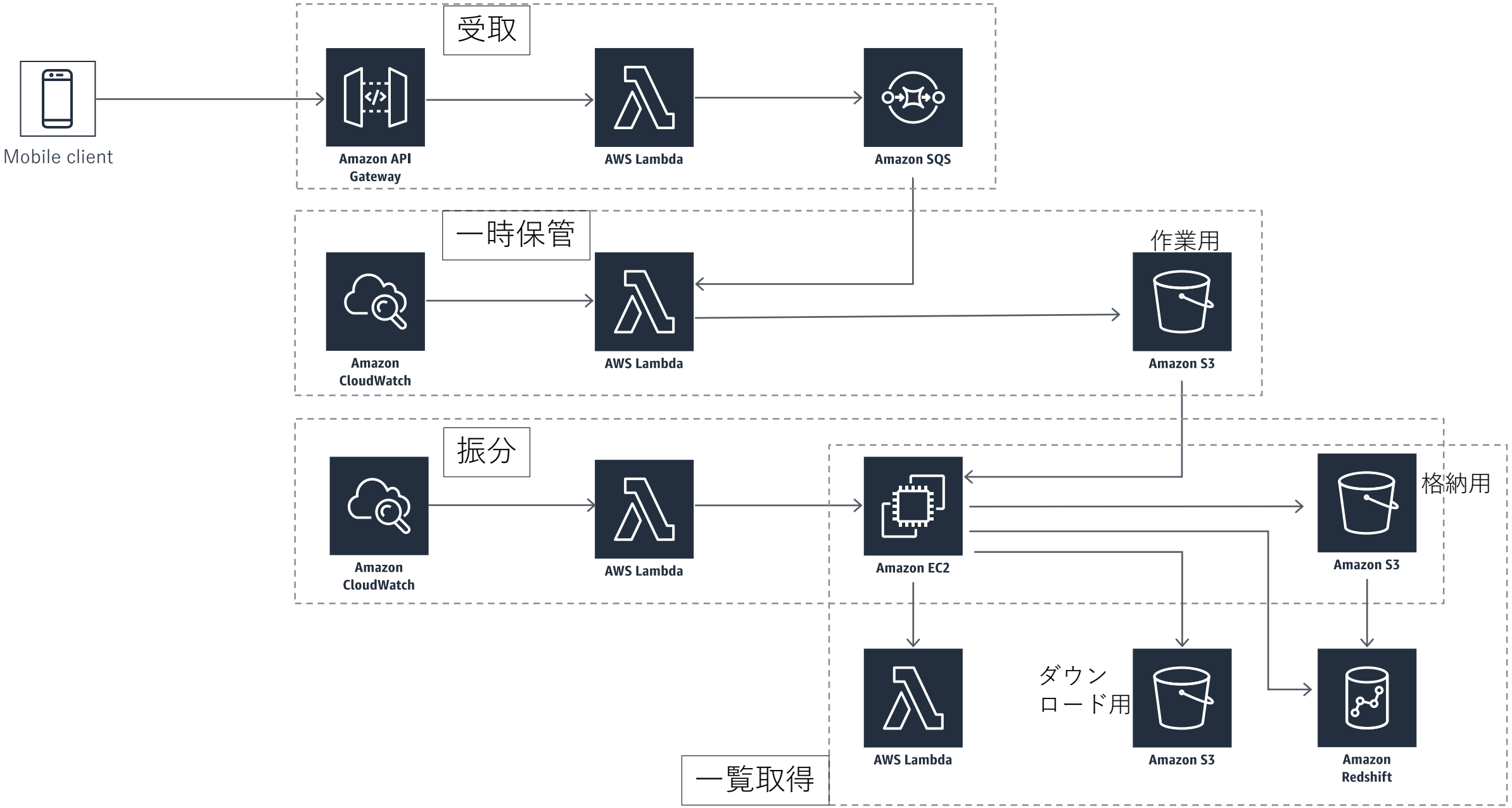
1. 受取
   位置情報のリクエストをAPI Gatewayで受けて、SQS(標準キュー)に格納する

2. 一時保管
   定期的(5min程度)にSQSからリクエストを取り出し、S3(作業用フォルダ)にまとめて書き出す

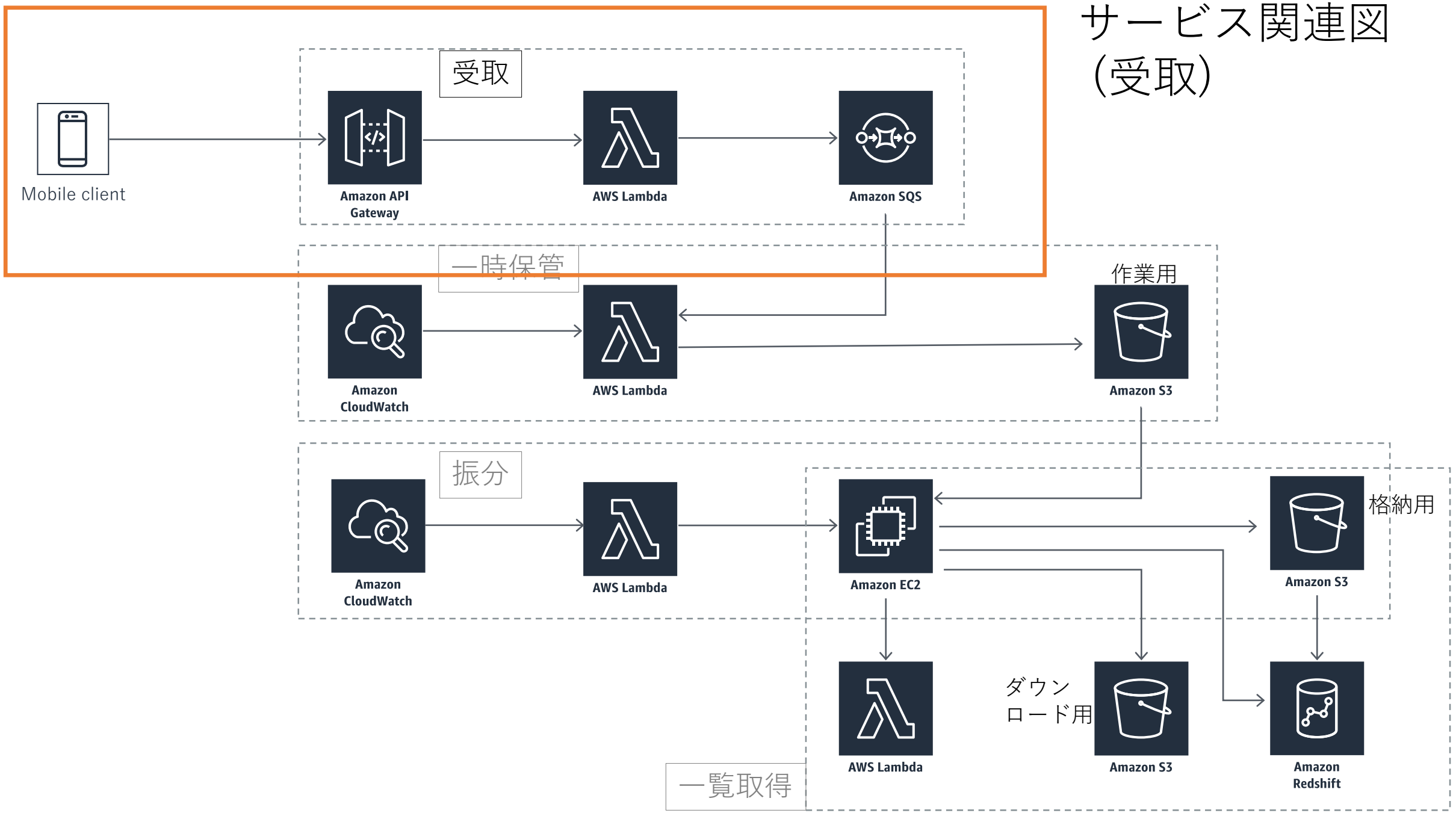
3. 振分
   定期的(1～数回/日)にS3(作業用フォルダ)のデータを読み込み、S3(格納用フォルダ)にリクエストの日付毎に分けて格納する。これがRedshiftのソースになる。

4. 一覧取得
   Redshiftから対象日のデータを取り出し、CSVに変換して、S3(ダウンロード用フォルダ)に格納する

サービス関連図


受取
Mobile client
Amazon API Gateway
AWS Lambda
Amazon SQS
一時保管
作業用
Amazon CloudWatch
AWS Lambda
Amazon S3
振分
Amazon CloudWatch
AWS Lambda
Amazon EC2
Amazon S3
格納用
AWS Lambda
ダウンロード用
Amazon S3
Amazon Redshift
一覧取得

サービス関連図（受取）


受取
Mobile client
Amazon API Gateway
AWS Lambda
Amazon SQS
一時保管
Amazon CloudWatch
AWS Lambda
作業用
Amazon S3
振分
Amazon CloudWatch
AWS Lambda
Amazon EC2
格納用
Amazon S3
AWS Lambda
ダウンロード用
Amazon S3
Amazon Redshift
一覧取得

受取フェーズの流れ

受取

① 端末で取得した位置情報のリクエストをAPIGatewayのエントリに送る
(POSTメソッド, content-type: json, body部にjsonデータ)

② APIGatewayがLambda(parse_request) を呼び出す
LambdaはPython(3.7)で記述される (以後同じ)

③ parse_requestでSQSにリクエストを積む
この際、JSONの値チェックなどのリクエスト単体のvalidationは行う
積む際にCSV形式の文字列に変換している
なお、SQSは標準キューである

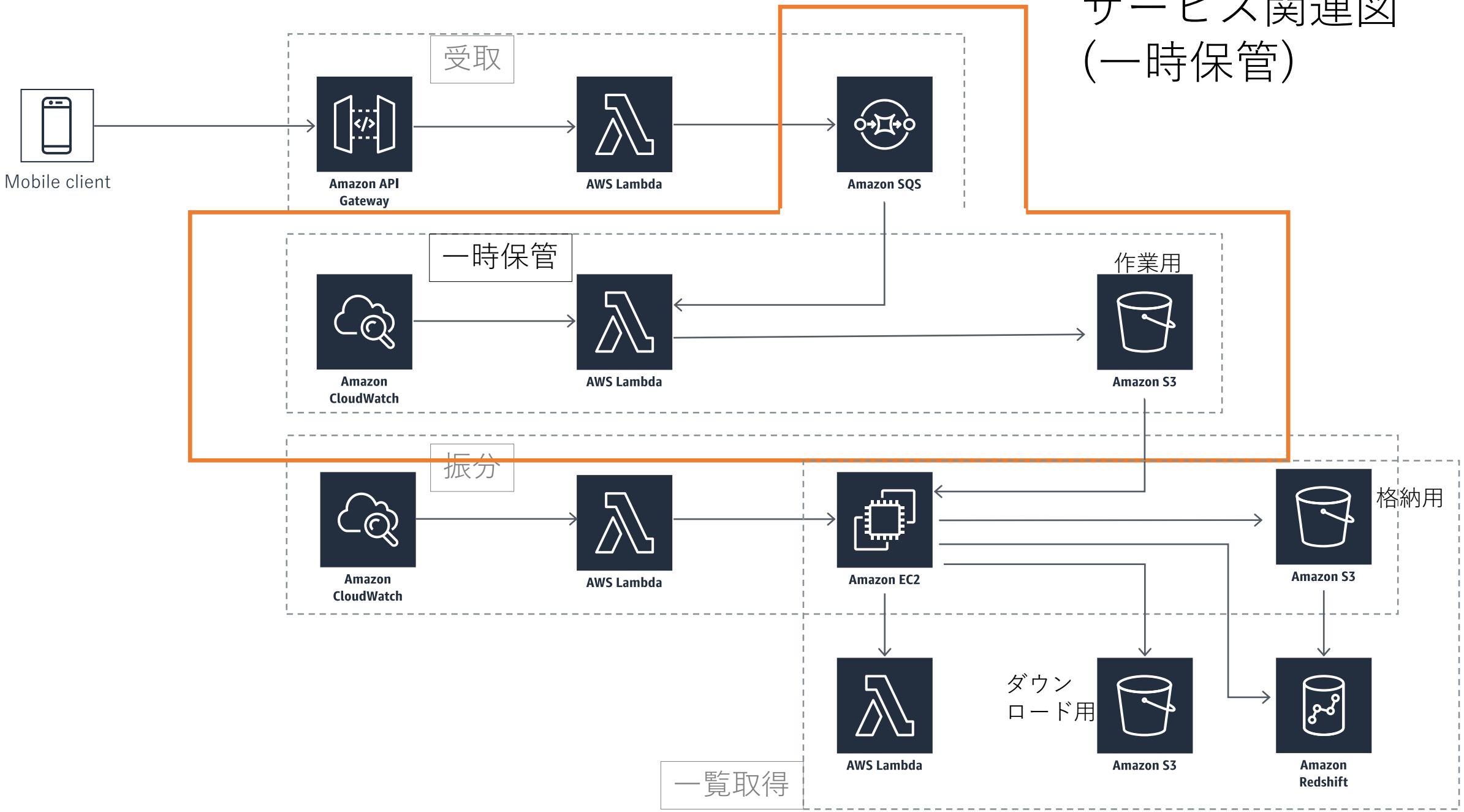
受取フェーズ補足

- リクエストをいったんSQSに積むのは?
  → 想定より毎秒277回のアクセスがあり、リクエストを確実に受け取るため

- SQSをFIFOではなく標準キューにした理由は?
  → 順序が重要ではなく、メッセージ数などの制約が少ないから

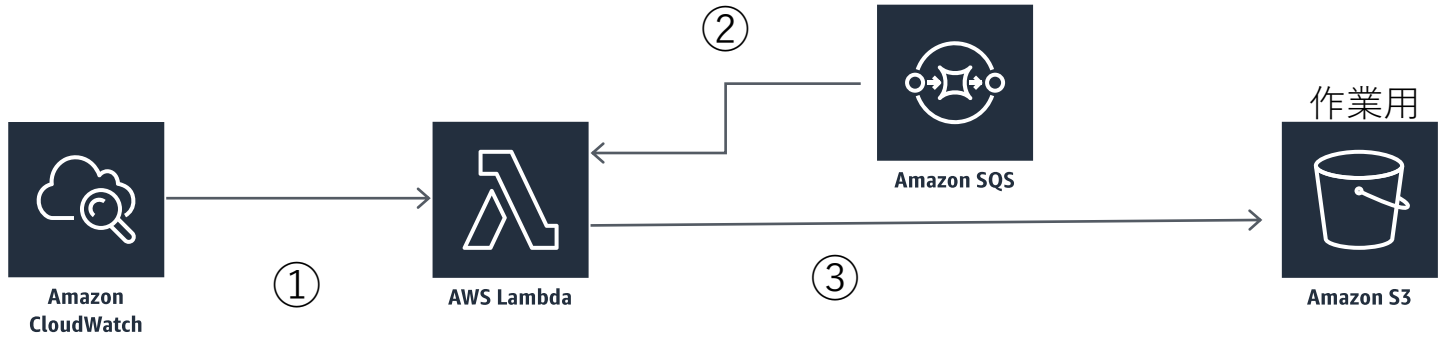
- SQSメッセージの重複対策は?
  → 次のフェーズで同一内容のメッセージは削除する処理を行っている

サービス関連図
(一時保管)



一時保管フェーズの流れ

一時保管



① CloudWatch Eventsによって、5分毎にLambda(store-request)を呼び出す

② Lambda(store_request)がSQSからリクエストを取り込む
前フェーズでSQSに蓄えられたリクエストを全て取り出し、このとき重複を取り除く

③ 取り出したリクエストを一つのオブジェクトにまとめる
オブジェクトは、S3の所定の作業用フォルダに格納される

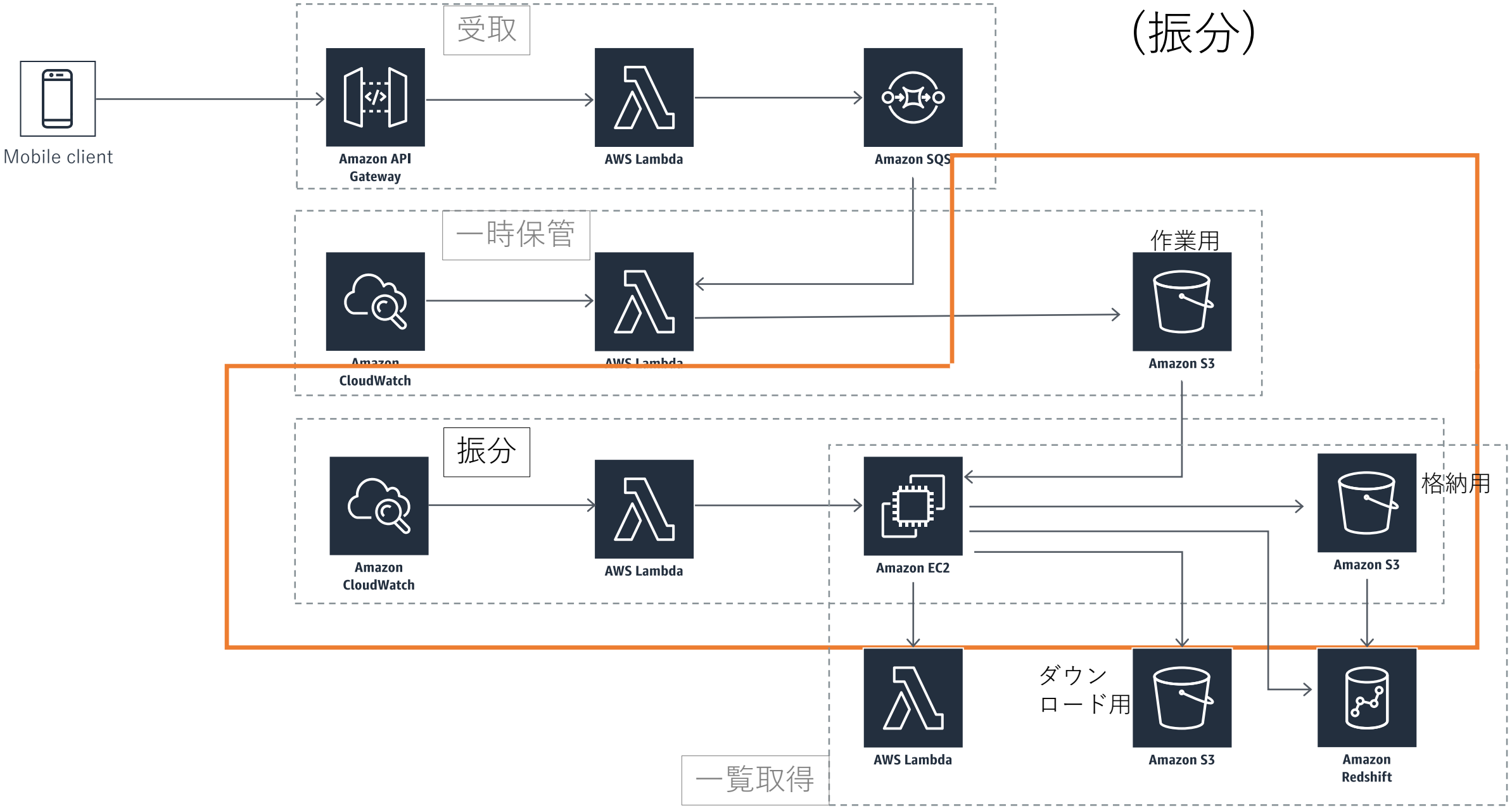
一時保管フェーズ補足

- 5分の根拠は?
  → 特になし。ただし、CloudWatchEventsにより制御しているだけなので、実際のデータ量でテストを行うことで変更が可能。
  なお、想定値では、83,000件ほどのデータを処理する。

- 処理したデータをいったんS3に置かなくても、そのままSQSでも良いのでは?
  → なんらかのトラブルがあっても復旧できるよう、永続的な形でデータを保持するため。
  また、S3をRedshift Spectrumのデータソースに指定することで、まだ集計されていないリアルタイムな位置情報にアクセスすることができるように(拡張性への布石)。

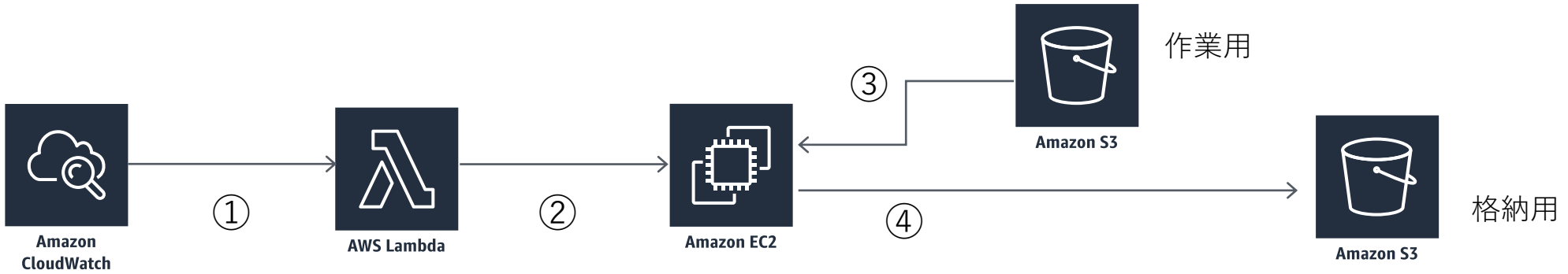
- 丁度Lambdaが起動したときに、SQSメッセージに重複したメッセージが来た場合は?
  → 対策していない。おそらくレアケースなので、最終結果を受け取った処理系で対応することを期待している(二通案内が届くことを容認するなど)。

サービス関連図（振分）


受取
Mobile client
Amazon API Gateway
AWS Lambda
Amazon SQS
一時保管
作業用
Amazon CloudWatch
AWS Lambda
Amazon S3
振分
Amazon CloudWatch
AWS Lambda
Amazon EC2
格納用
Amazon S3
ダウンロード用
AWS Lambda
Amazon S3
Amazon Redshift
一覧取得

振分フェーズの流れ

振分



① CloudWatch Eventsによって、0時過ぎにLambda(start_collect_server)を呼び出す

② Lambda(start_collect_server)はEC2サーバを起動する
サーバ起動後、自動的にretrieve_request.pyを実行する

③ retrieve_request.pyによって、S3(作業用)に蓄えられたリクエストを日付毎に振り分ける
(Redshift Spectrumで、日付によるパーティショニングを行っているため)

④ 日付毎に振り分けられたデータをS3(格納用)に格納する
S3(格納用)がRedshift Spectrumのデータソースになる

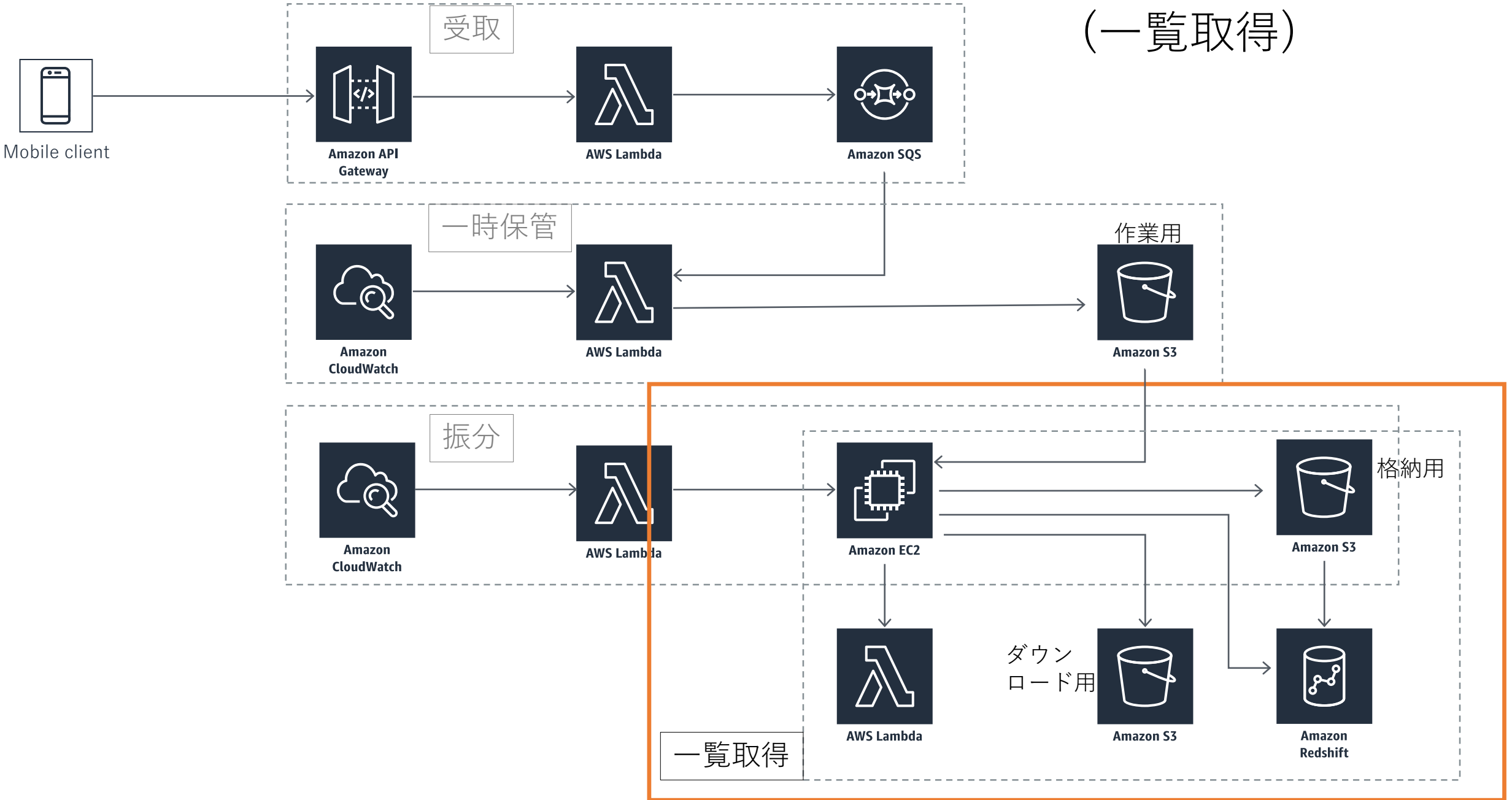
振分フェーズ補足

- 一日一回しか実施しない理由は?
  → Redshift Spectrumでパフォーマンスが出やすくするために、パーティション用の日付フォルダ内のファイル数をできるだけ少なくするため。
  なお、データ数などの理由により一回で処理が行えない場合でも、複数回処理を実施しても大丈夫なように設計している。

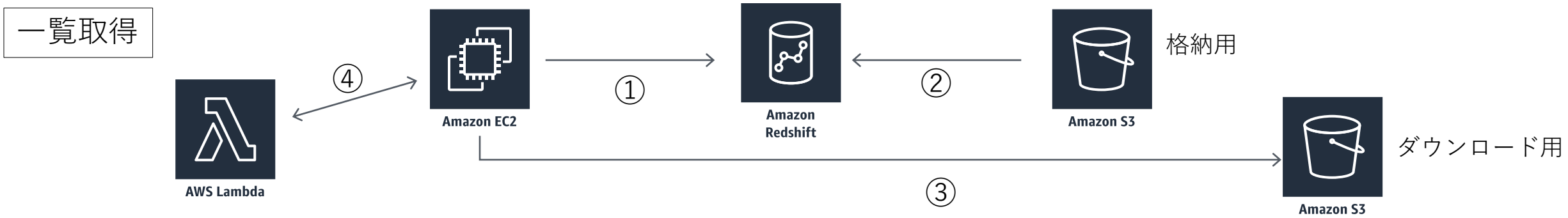
- 処理のたびにEC2サーバを起動し、終了するのは?
  → コスト削減のため

- Redshift SpectrumのデータソースとしてCSVを使っているが、Parquetは使用しないのか?
  → 想定された処理件数を考えると、性能上の問題で変換が難しい

サービス関連図
(一覧取得)



一覧取得フェーズの流れ



① 前のフェーズの後にcollect_request.pyが実行される
前日(または指定された日付)の位置情報一覧をRedshift Spectrumから取得する

② Redshift Spectrumは、S3(格納用)をデータソースとする

③ collect_request.pyは、対象日付のすべての位置情報を取得する
結果をS3(ダウンロード用)に格納する
S3(ダウンロード用)は公開されていて、ここから結果をダウンロードできる

④ 処理終了後、Lambda(stop_collect_server)を呼び出す
この関数はEC2サーバを終了させる

一覧取得フェーズ補足

- 前のフェーズの処理と一体化しない理由は?
  → 現在は両方とも一日一回で、実施するタイミングも同じだが、前のフェーズは一日数回実施する可能性を考慮して、処理を分けている。

- この処理は一日一回であることを想定しているのか?
  → 複数回実施しても、結果が上書きされるだけで問題が無いようにしている。

- ダウンロード用S3をPublicにするとセキュリティ上問題があるのでは?
  → 問題はあるが、セキュリティ上の要件は現段階では想定していない。