aws DEV DAY
ONLINE JAPAN

20-22.10.2020

DEV DAY

In Partnership with intel

aws DEV DAY
ONLINE JAPAN

F - 6

Tensorflow on Lambda
~機械学習予測処理の低コスト化~

中川帝人
上席課長/シニアデータサイエンティスト
株式会社オープンハウス/情報システム部　ディスラプティブ技術推進G

In Partnership with intel

自己紹介

- 氏名：中川帝人(@TeitoNakagawa)
- 所属：株式会社オープンハウス
- 役職：シニアデータサイエンティスト/上席課長/宅地建物取引士
- 得意分野：データベース、GIS、データ分析・機械学習、AIの企画・開発
- 経歴：日本オラクル、ソフトバンク等で分析コンサルタントから分析・新規事業企画系のマネジメントを担当。2018年3月よりオープンハウスでAI・データ活用の研究開発と活用のチームを牽引。不動産×AI・MLのソリューションを複数実現。
- 好きなAWSのサービス：Lambda

aws DEV DAY
ONLINE JAPAN

まとめ

In Partnership with intel

まとめ

不規則な社内予測サービスはLambdaで実行する

Usecase

○：社内システム、不規則なアクセス

Ｘ：コンシューマ向け、規則的なアクセス

Good

- インフラ考慮不要
- スケールを待たない
- レスポンスタイムも早い
- 同時実行数も多い

Risk

- 環境構築の検証
- モデルサイズ上限（EFSを利用）

Agenda

1. オープンハウスにおけるAI活用
2. オビ付け業務の自動化
3. 課題：不規則かつ急激に上がる負荷
4. Tensorflow on Lamda
5. 検証結果

In Partnership with intel

aws DEV DAY
ONLINE JAPAN

オープンハウスにおけるAI活用

In Partnership with intel

会社紹介

- 株式会社オープンハウス
  - 総合不動産会社
  - 戸建開発と販売が主力事業
  - 営業中心の会社
- 情報システム部
  - 内製主義＝コード力重視
  - ほぼ全てのシステムでOn Cloud
  - 最近はPythonが主流に
  - AI・RPAで事例多数あり

https://oh.openhouse-group.com/kanto/about/

In Partnership with intel

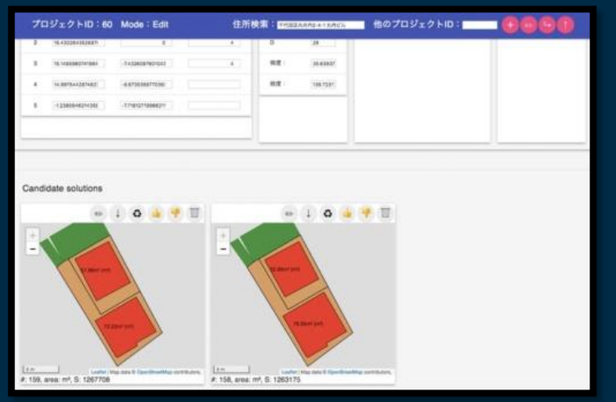
不動産×AIはニーズ大

- ニーズないでしょ
⇒あります
- 価格予測でしょ？
⇒ニーズない
- 実は結構できることある
  - 超アナログ
    ⇒ITの活躍の余地が大きい
  - 書類がとにかく多い
    ⇒AI・MLの活躍の余地が大きい
  - 単純な作業が多い
    ⇒現状のAIでも十分実現可能

実際に取り組んできたこと

AI・RPA事例

- **物体検知モデルでオビ付け業務を自動化**
- 画像分類モデルで社内物件資料を自動生成
- GISによるレコメンド処理
- 遺伝的アルゴリズムによる区割設計の自動化
- 物件資料自動取得RPA
- オンラインチラシ全自動作成システム
- 機械学習モデルによるWeb広告効果評価
- OpenCVによる類似物件検索



In Partnership with intel

aws DEV DAY
ONLINE JAPAN

オビ付け業務の自動化

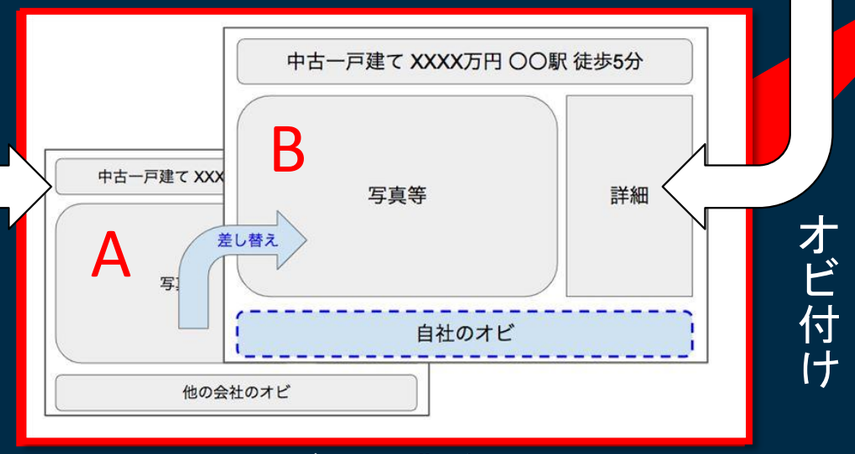
In Partnership with intel

オビ付けとは？

マイソク（＝チラシ）の他社のオビ（＝情報）を
自社のオビに差し替える作業

仲介の流れ

売主 → 仲介A → レインズ等 → 仲介B → 買主



作成

オビ付け

オビ付け業務

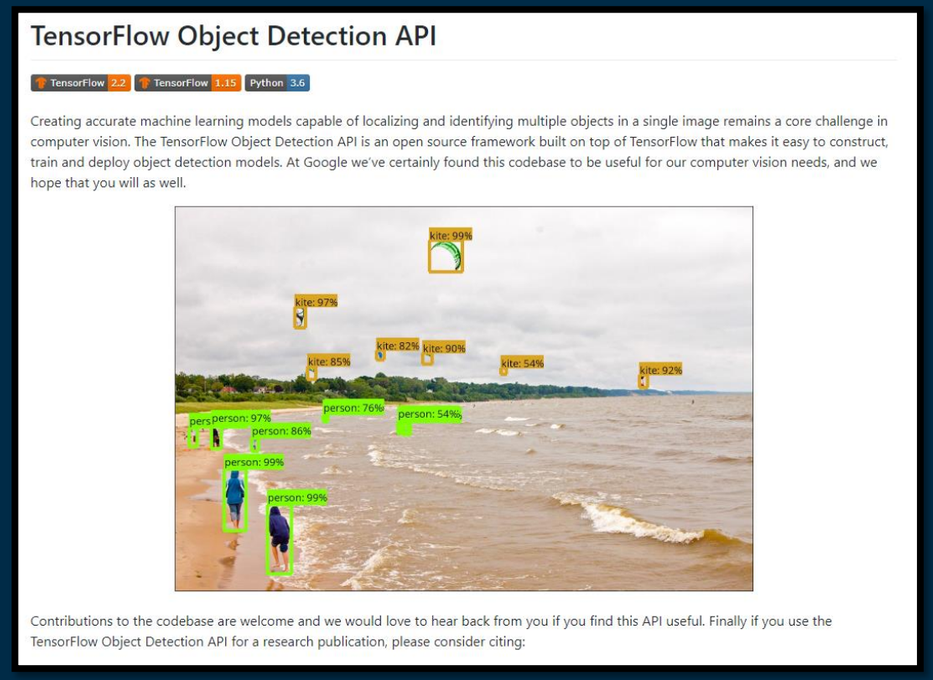
⇒画像処理で実現

In Partnership with intel

物体検知とは

オビ付けは画像処理＝物体検知モデルで実装可能

- 実装はオープンソース
- サンプルはネットに腐るほどある
- 有名なソフトウェア
  - YOLO
  - Tensorflow Object Detection API
- モデル
  - SSD
  - Faster R-CNN

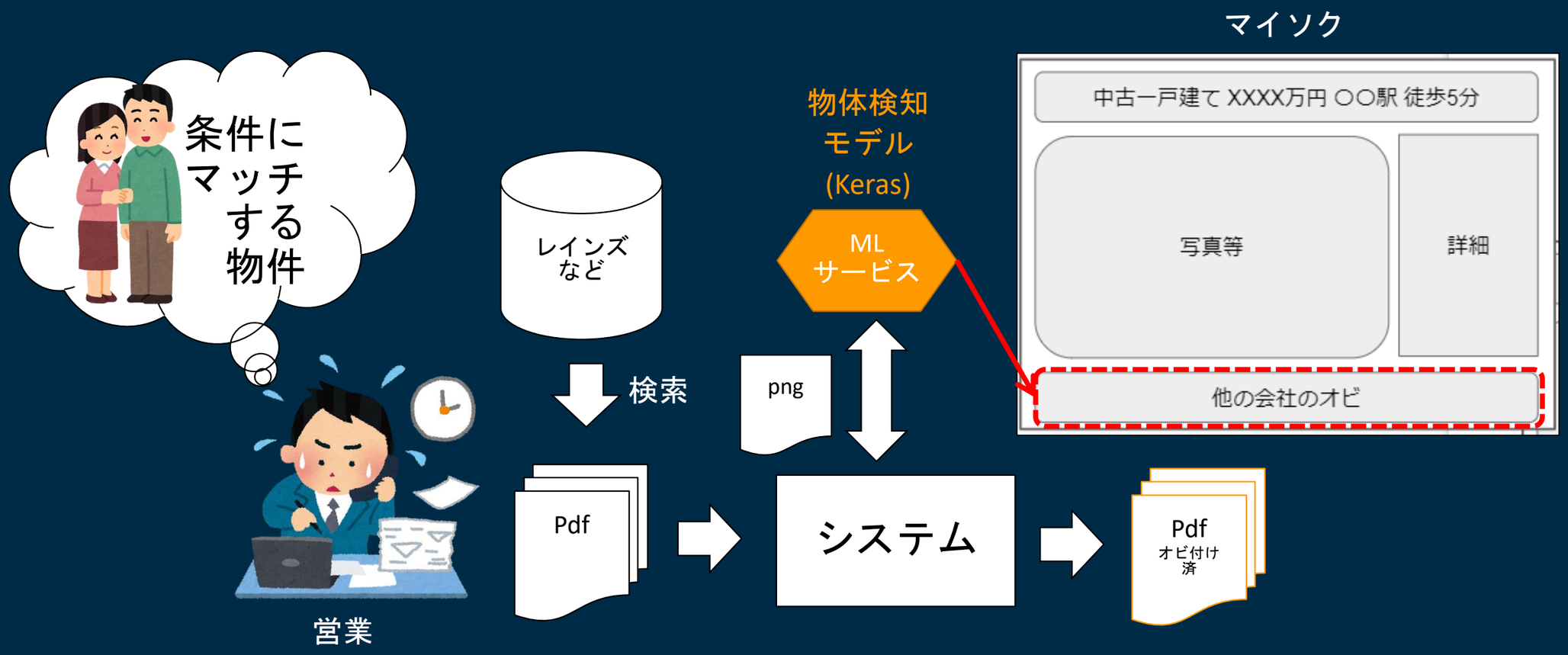


https://github.com/tensorflow/models/tree/master/research/object_detection

In Partnership with intel

作ったシステムの利用イメージ

PDFで送られたマイソクに物体検知モデルでオビを貼り付ける



In Partnership with intel

aws DEV DAY
ONLINE JAPAN

課題：不規則かつ急激に上昇する負荷

In Partnership with intel

不規則かつ急激に上昇する負荷

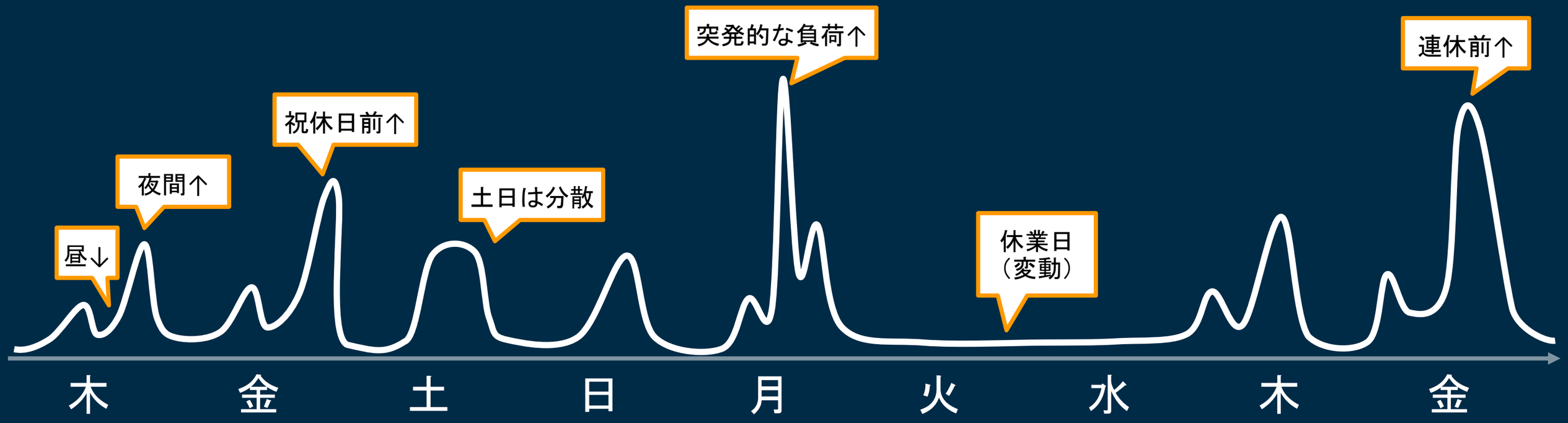
厳しいインフラ要件

1. 1回の予測の負荷が重い：入力画像のサイズを落とすと精度が下がる
2. 1回のリクエストの負荷が重い：一つのpdfが多くて50ページ
3. リクエストが集中：500人近い営業が大体同じ時間に作業
4. 不規則なリクエスト：休みの前日に負荷が上昇、休業日は不規則
5. 高い性能要件：働き方改革で現場の時間は足りない

In Partnership with intel

実際の負荷イメージ

変動要因が多すぎて負荷を予測できない
いつ負荷が上がってもいい形で実装せざるを得ない

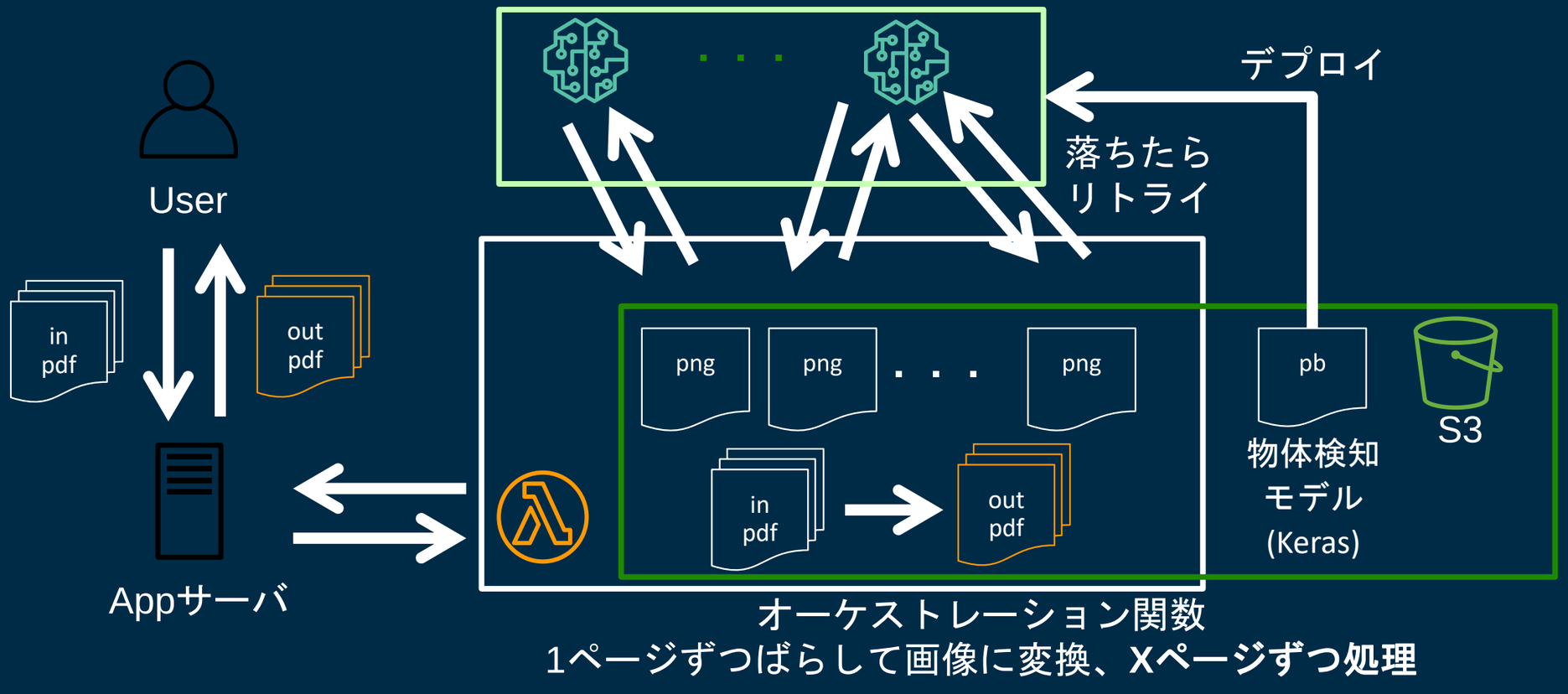


In Partnership with intel

アーキテクチャ（ASIS）

MLサービスを複数デプロイ、急激な負荷上昇に耐える

MLサービス：常時X並列のEPで処理を受け付ける。
高負荷の時はそれぞれがスケールする



In Partnership with intel

実装上の課題

- 常時複数サービスを起動した結果、コストが上がってしまった
- 複数サービスでもさばけない事が多々発生し、リトライを実装しなければならなかった
- スケーリングに時間がかかるため無駄なリソースを確保した

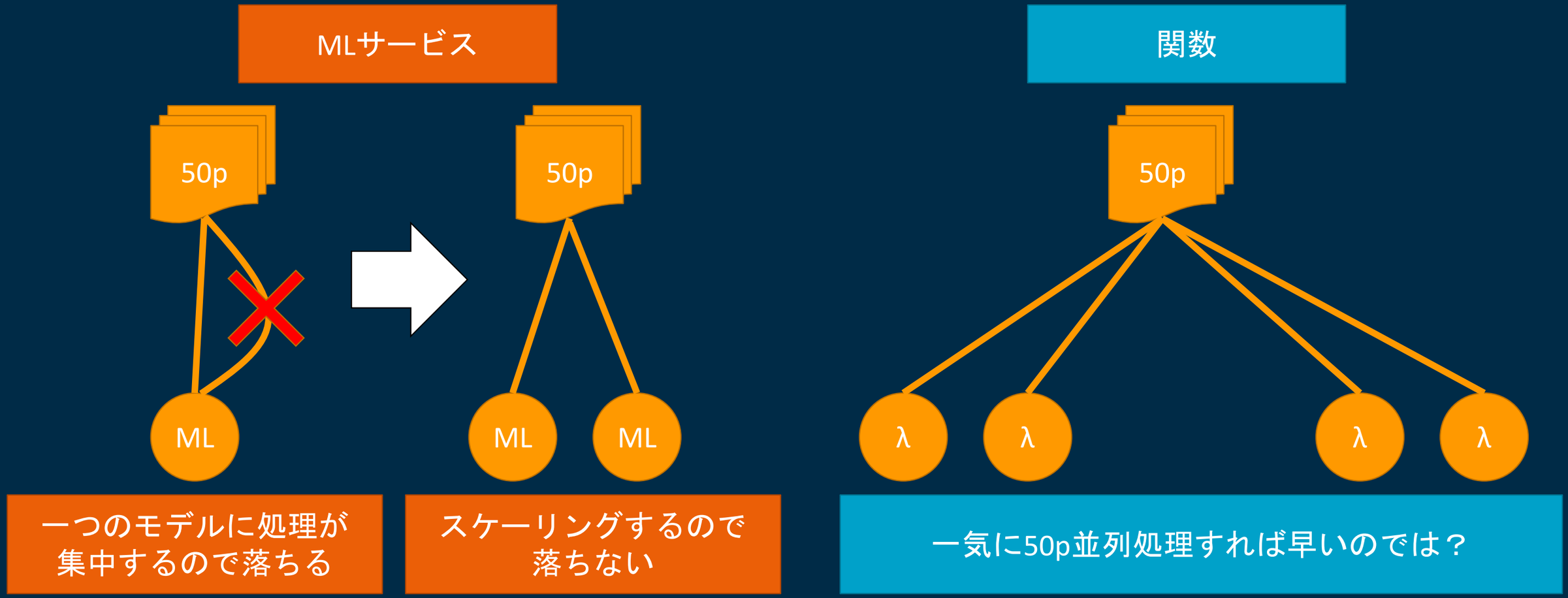
⇒関数で解決できないか？

aws DEV DAY
ONLINE JAPAN

Tensorflow on Lamda

In Partnership with intel

関数コンピューティングで解決出来るのでは？



In Partnership with intel

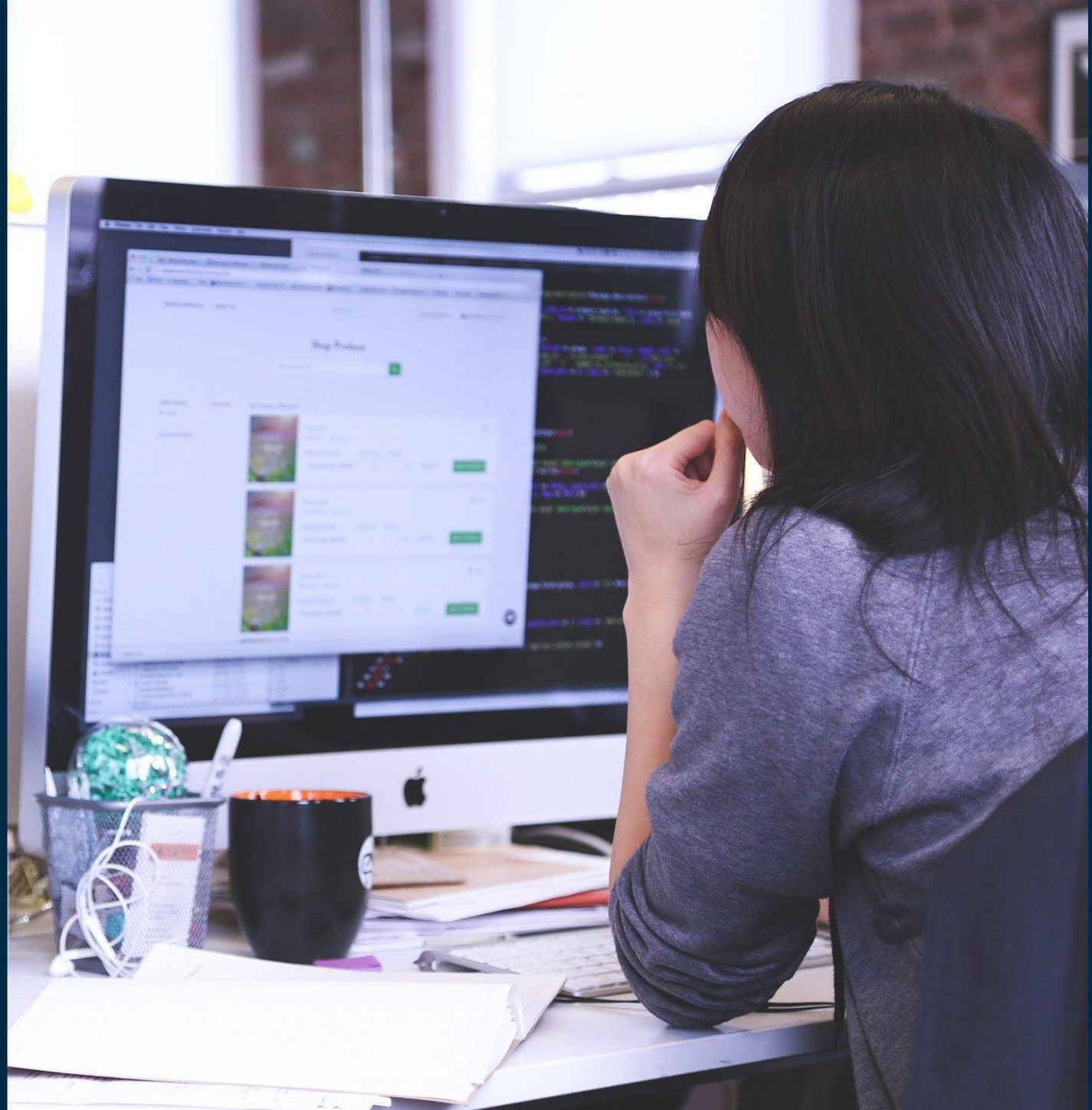
機械学習予測の方式比較

	サーバ	MLサービス	関数
実装	サーバ(コンテナ)構築 EC2,ECRでスケーリング	Sage Maker	Lambda
開発	独自Appサーバ Tensorflow Serving で関数や環境を構築	Modelを配置する	コードと モジュールのロード
メリット	自由に実装できる	スケーリングは 自動で実行	実行した分だけ課金 不規則なアクセス〇
デメリット	スケーリング処理 実装は複雑	スケーリングが遅い	環境の制約が強い 毎回環境が構築される

Lamda実装で工夫したポイント

1. モジュールサイズの削減：Tensorflow自体が大きいモジュールのため、Serverlessを使ってサイズを削減した

2. 初期化処理をinitに：予測の度に変わらない変数の初期化（環境の構築）は全てinitに記載することで実行時間を節約した



実装概要

- ランタイム : Python 3.6
- メモリ： 3008 MB
- モデル： Keras(Tensorflow)
- モジュール
  - tensorflow=2.0.2,
  - numpy=1.7.2,
  - Pillow=6.1.0,
  - h5py=2.10.0
- Severless： 1.78.1

モジュールサイズの削減

Python Lambda最大の問題：パッケージサイズ（特に非ピュアPython）

```
Serverless Error ------------------------------------

 An error occurred: ClassifyLambdaFunction - Unzipped size must be smaller than 176515274 bytes (Service: AWSLambdaInternal; Status Code: 400; Error Code: InvalidParameterValueException; Request ID: f59a1c34-8226-439d-9d86-04e3f6c5946b; Proxy: null).
```

ServerlessでTensorflowのディスク利用を削減してデプロイ可能に

In Partnership with intel

Serverless上の設定

Serverless Framework： 関数コンピューティングのためのフレームワーク

- 複数のクラウドサービスに対応
- ローカルでの実行を簡単に

Serverless Python Requirements： Python Requirementsを扱うためのServerlessプラグイン

- dockerizePip：クロスコンパイルでデプロイを高速化
- slim,slimPatterns:不要なモジュールを削除してディスクを減らす

参考：「AWS LambdaでTensorflow 2.0を使った画像分類」
https://tech.unifa-e.com/entry/2019/09/17/085400?utm_source=feed

初期化処理をinitに

初期化（モデルダウンロード）に1番時間がかかっていた

処理時間を1/6に削減≒コストを1/6に削減

In Partnership with intel

コードの実装方法

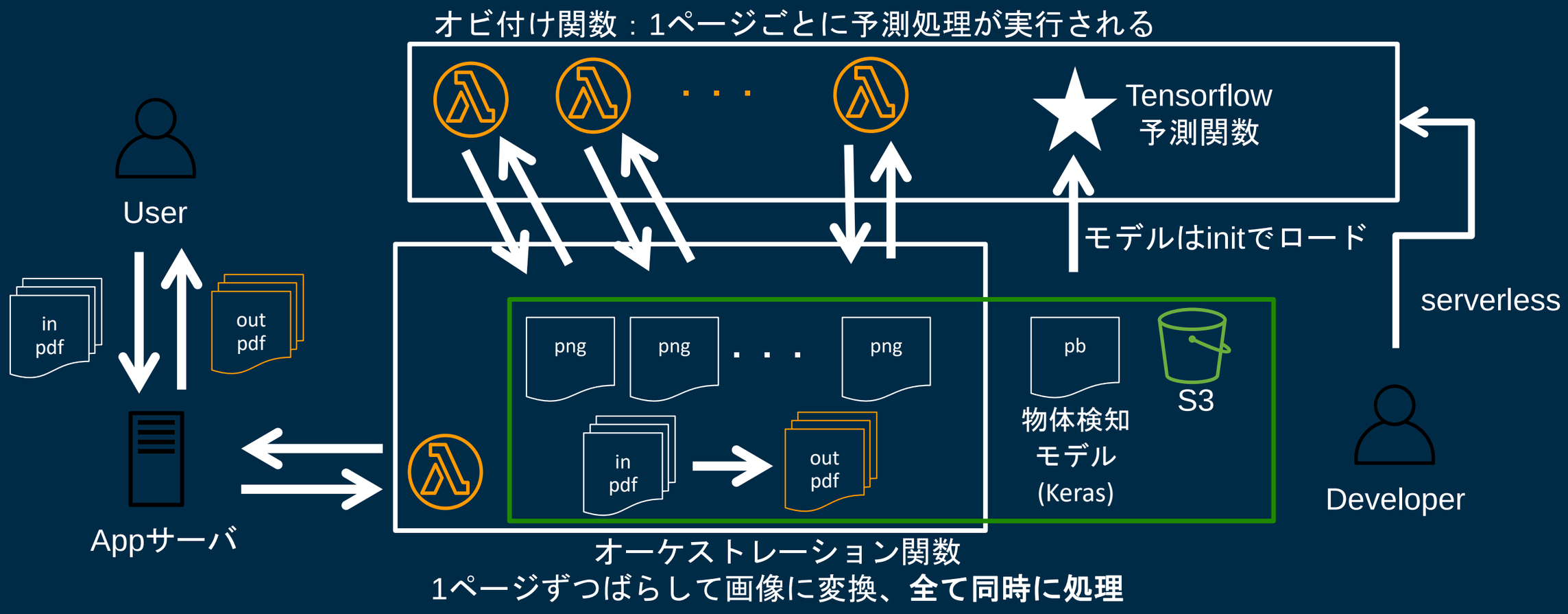
共通した処理はinitに記載すれば1度しか実行されない。

```
…
MODEL_FN = None #予測関数のグローバル変数
…
#初期化：モデルのダウンロード、予測関数の抽出
def init():
    global MODEL_FN
    MODEL_FN = NEW_MODEL.signatures['serving_default']
    …

#予測：S3画像のダウンロード、前処理、予測
def handler(event, context):
    global MODEL_FN
    …
    output_dict = MODEL_FN(encoded_image)
    …
```

アーキテクチャ（TOBE）

ラムダ関数に変更することで全て同時に処理できる



In Partnership with intel

aws DEV DAY
ONLINE JAPAN

検証結果

In Partnership with intel

検証条件

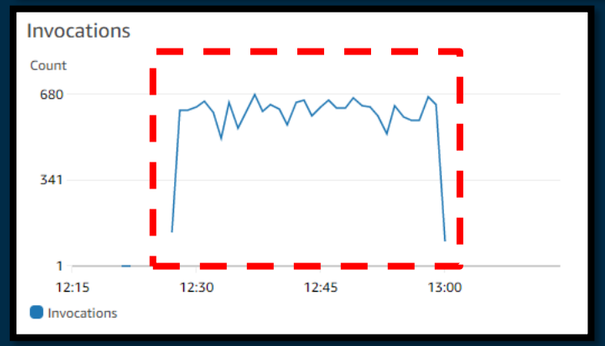
方法：Pythonでローカル PCから
予測関数をコール（非同期）

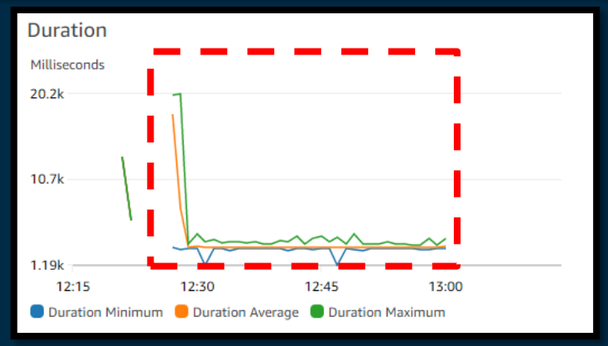
条件
実行回数：10000回
テスト時間：約33分
同時実行回数：約285回/分

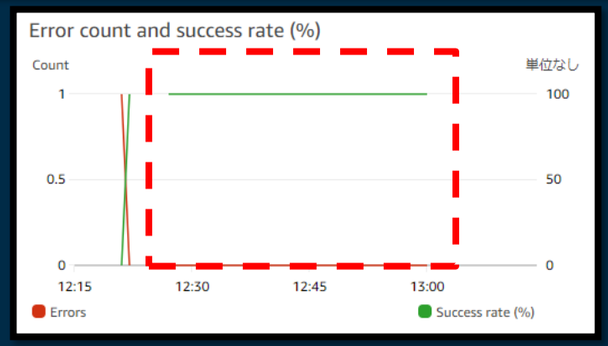
⇒実際の負荷を大きく超える
条件で検証を実施

検証結果

```
2020-10-12T12:33:33.361+09:00	START RequestId: 61119c60-40ff-41da-8759-e99d865d414b Version: $LATEST
2020-10-12T12:33:33.592+09:00	classify
2020-10-12T12:33:36.612+09:00	END RequestId: 61119c60-40ff-41da-8759-e99d865d414b
2020-10-12T12:33:36.612+09:00	REPORT RequestId: 61119c60-40ff-41da-8759-e99d865d414b Duration: 3246.95 ms Billed Duration: 3300 ms Memory Size: 3008 MB Max Memory Used: 2058 MB
```







- Max Used Memory：概ね2058MB
- エラー回数：0回
- 利用料金：約$1

⇒エラーなしでコストとレスポンスタイムの改善を実現

In Partnership with intel

aws DEV DAY
ONLINE JAPAN

まとめ

In Partnership with intel

まとめ

不規則な社内予測サービスはLambdaで実行する

Usecase

○：社内システム、不規則なアクセス

Ｘ：コンシューマ向け、規則的なアクセス

Good

- インフラ考慮不要
- スケールを待たない
- レスポンスタイムも早い
- 同時実行数も多い

Risk

- 環境構築の検証
- モデルサイズ上限（EFSを利用）