大量の行動データに基づく リアルタイム予測基盤の実現

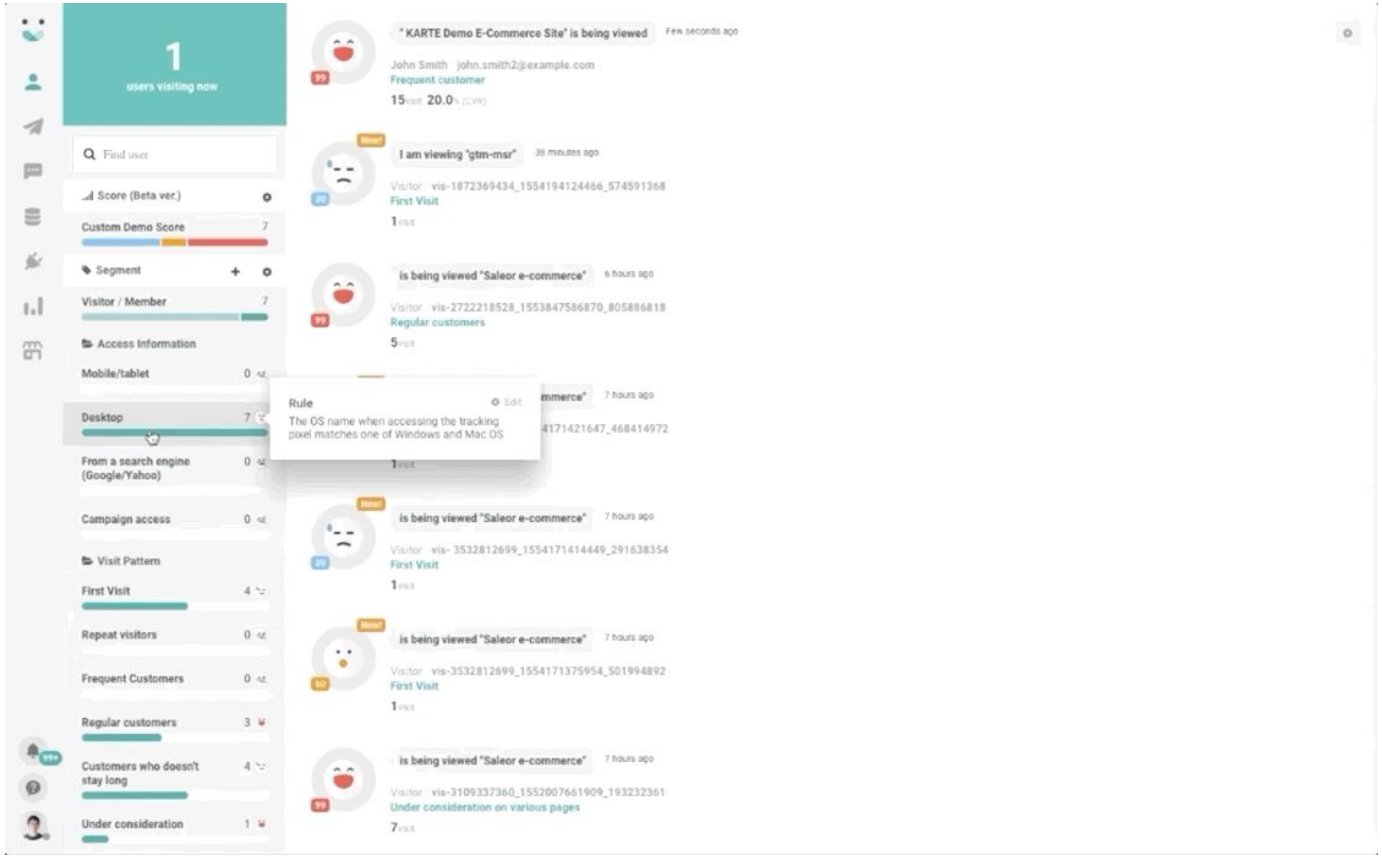
西村 優汰

株式会社プレイド
エンジニア

KARTEとは

KARTE

Real-time, Large-Scale, Global



KARTE 導入企業様（一部抜粋）

※2020年5月8日時点（ABC/50音順）

EC / 総合通販

AEON.com
Alpen Group
BAYCREW'S STORE
ENOTECA
GDO
KOMEHYO
LUXA
MIX.Tokyo
MIZUNO
PAL CLOSET ONLINE STORE
QVC
THE SUIT COMPANY
ZOZOTOWN
三井ショッピングパーク &mall
モノタロウ

金融 / 保険

Manulife マニュライフ生命
MIZUHO
Money Forward
PayPay
SBI証券
SAISON POINT MALL 永久不滅ポイント
Rakuten 楽天生命
WealthNavi
イーデザイン損保
三井ダイレクト損保 MS&AD INSURANCE GROUP

IT / サービス

AllAbout
bellFace
DeNA
freee
GMO RESEARCH
Lancers
NRI 未来創発
RECRUIT
SmartHR
Y!mobile

コンテンツ

DELISH KITCHEN
Rakuten TV
サンデーうぇぶり
ジャンプBOOKストア!
テレビ東京 ビジネス オンデマンド

メーカー

Canon キヤノンマーケティングジャパン株式会社
KIRIN
ネスレ Nestlé Good Food, Good Life
House
VOLVO

人材

BIZREACH
en
WANTEDLY
リクナビ

旅行 / 交通

HIS
JTB 感動のそばに、いつも。
JAL JAPAN AIRLINES
JapanTaxi

美容 / ヘルスケア

@cosme SHOPPING
エステティック TBC
KOSÉ
TIPNESS

不動産 / 住宅情報

FREEDOM
LIFULL HOME'S
リノベる。
三菱地所

教育 / 学習

学校法人 京都芸術大学 通信教育課程
スタディサプリ
LEC東京リーガルマインド

KARTE

CX(顧客体験)プラットフォーム

68. 3 億 UU
累計ユーザー数 ※1

1.55 兆円
年間解析流通金額 ※2

105,000 over
秒間トラッキング数 ※3

0.x 秒/解析
解析速度

180+ PB
月間解析データ量

8+ PB
蓄積データ量

※1 ローンチ～2020 年 2 月までの解析ユニークユーザー数の実績
※2 EC 領域における解析流通金額。2019 年 3 月～2020 年 2 月までの単年の実績
※3 秒間解析イベント数（閲覧、購入、クリックなど全計測イベントが対象2020 年 3 月の最大値）

データ解析の取り組み

02

Batch 予測と Realtime 予測

Batch 予測

- 使用するデータが数日、数週間で変化する場合

1週間後

ロイヤル
カスタマーになるか

Realtime 予測

- 使用するデータが数秒で変化する場合

20 秒後

問い合わせをするか

Batch 予測と Realtime 予測

Batch 予測

- 使用するデータが数日、数週間で変化する場合

1週間後

ロイヤル
カスタマーになるか

Realtime 予測

- 使用するデータが数秒で変化する場合

20 秒後

問い合わせをするか

Realtime 予測で実現したい例

問い合わせ予測

課題

電話とチャットの窓口を用意しているが、電話お問い合わせのオペレーションコストが高い

解決策

お問い合わせしそうなエンドユーザーに対して、チャットへのお問い合わせを促し、コスト削減

離脱予測

課題

会員登録でわからない点があり、離脱してしまう

解決策

離脱しそうなエンドユーザーに対して、ヘルプページなどを提案

購買予測

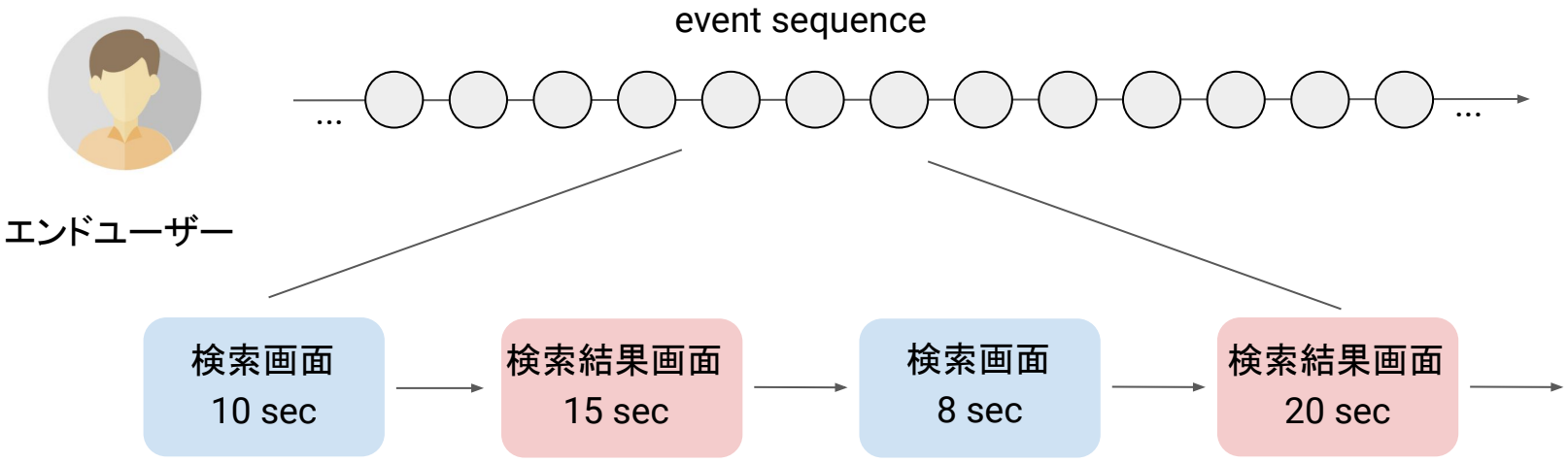
課題

クーポンがなくても購買するユーザーにもクーポンを配布しており、コストが増加している

解決策

購入を迷っているエンドユーザーにだけ、クーポンを配布することで ROI を最大化

Realtime 予測の詳細例 (行動の判別)

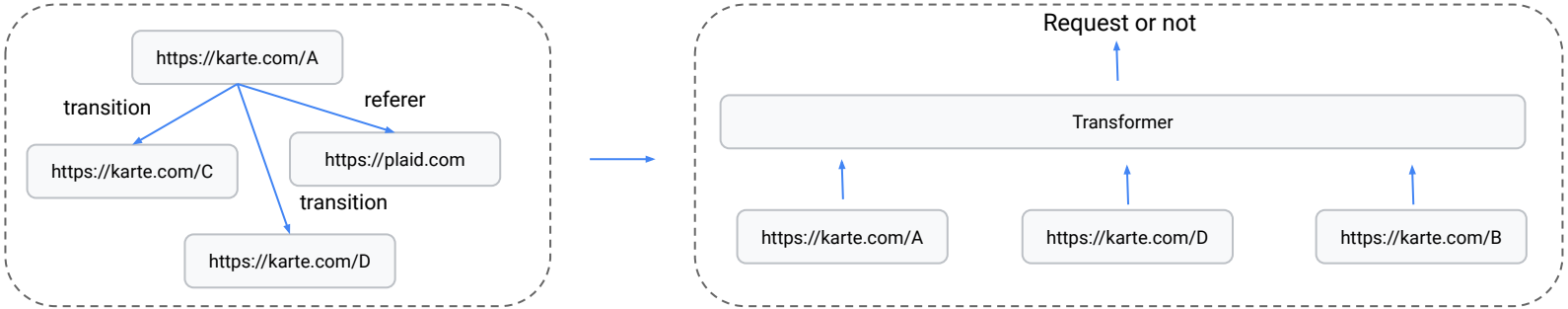


うまく検索できていないユーザー行動

上記と類似したユーザー行動をリアルタイムに判別

Graph と Sequence モデルの融合

- 様々な業界の Web Page をどう汎用的に embedding を作成するか
- 様々なエンドユーザーの時系列行動をどう汎用的にモデリングするか



web page embedding using graph

binary classification using transformer

リアルタイム
予測基盤

03

リアルタイム基盤で実現したいこと

リアルタイムデータの活用

リアルタイムに変化する
行動データを保持

適切な範囲の行動データを
リアルタイムに取出

様々なモデルをホストする

同じタスクのモデルでも
エンドユーザーによって
特徴量が違うモデル

同じ特徴量のモデルでも
お客様が違うモデル

実験と検証のサイクルを素早く回す

モデルの精度だけでなく、
Action まで踏まえたモデルの
価値検証が必要

リアルタイム基盤で実現したいこと

リアルタイムデータの活用

リアルタイムに変化する
行動データを保持

適切な範囲の行動データを
リアルタイムに取出

様々なモデルをホストする

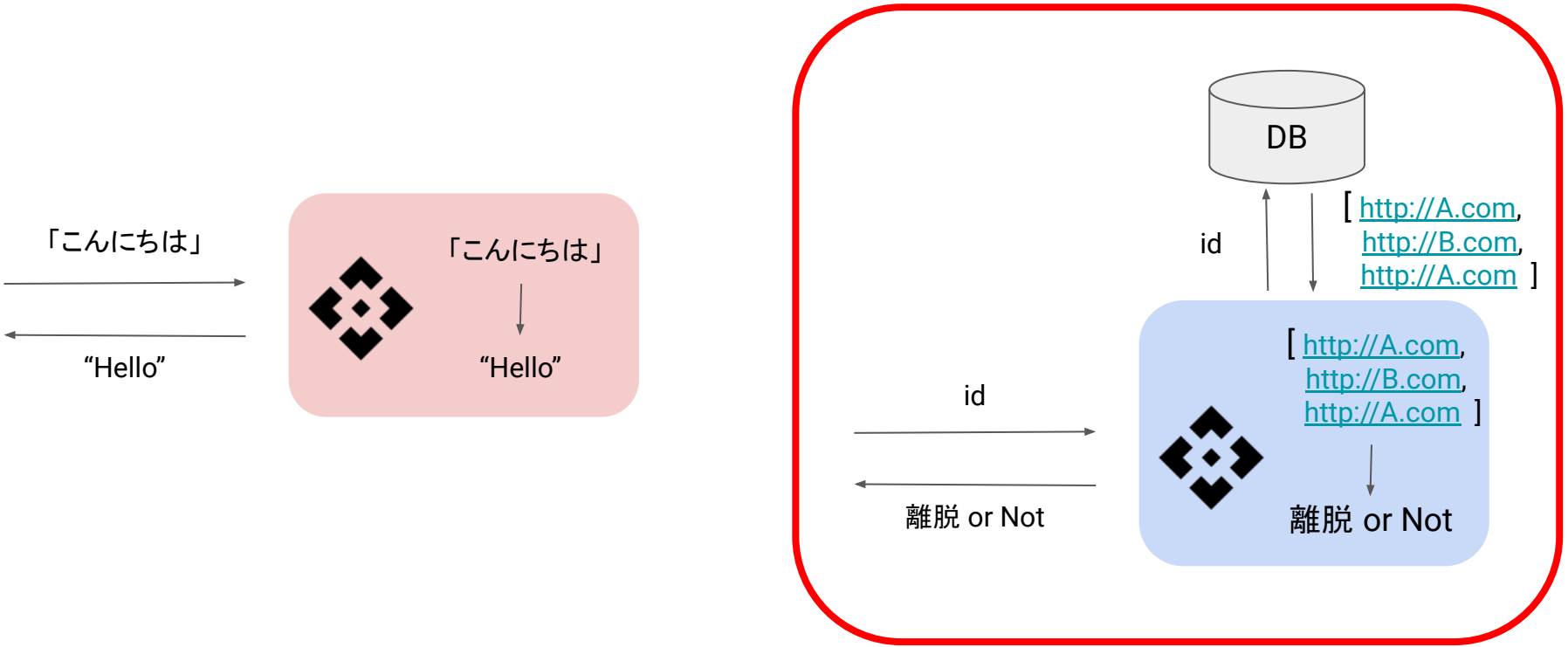
同じタスクのモデルでも
エンドユーザーによって
特徴量が違うモデル

同じ特徴量のモデルでも
お客様が違うモデル

実験と検証のサイクルを素早く回す

モデルの精度だけでなく、
Action まで踏まえたモデルの
価値検証が必要

stateless なモデルとの違い



リアルタイムに変化するデータ


```
{
  "event_name": "view",
  "sync_date": “2020-11-29 23:50:23.142 UTC",
  "keys": {
     "api_key": "xxxxxx",
     "user_id": "user_B",
     "session_id": "yyyy_662",
  },
  "page": "https://plaid.co.jp/",
  ...
}
```


user_id	date	event_name	...
user_A	2020-11-29 23:38:23.010 UTC	view	...
user_B	2020-11-29 23:40:23.042 UTC	buy	...
user_B	2020-11-29 23:50:23.142 UTC	view	...
...	...	...	....

Web App

エンドユーザー

予測

予測結果に応じた Action

リアルタイム基盤で実現したいこと

リアルタイムデータの活用

リアルタイムに変化する
行動データを保持

適切な範囲の行動データを
リアルタイムに取出

様々なモデルをホストする

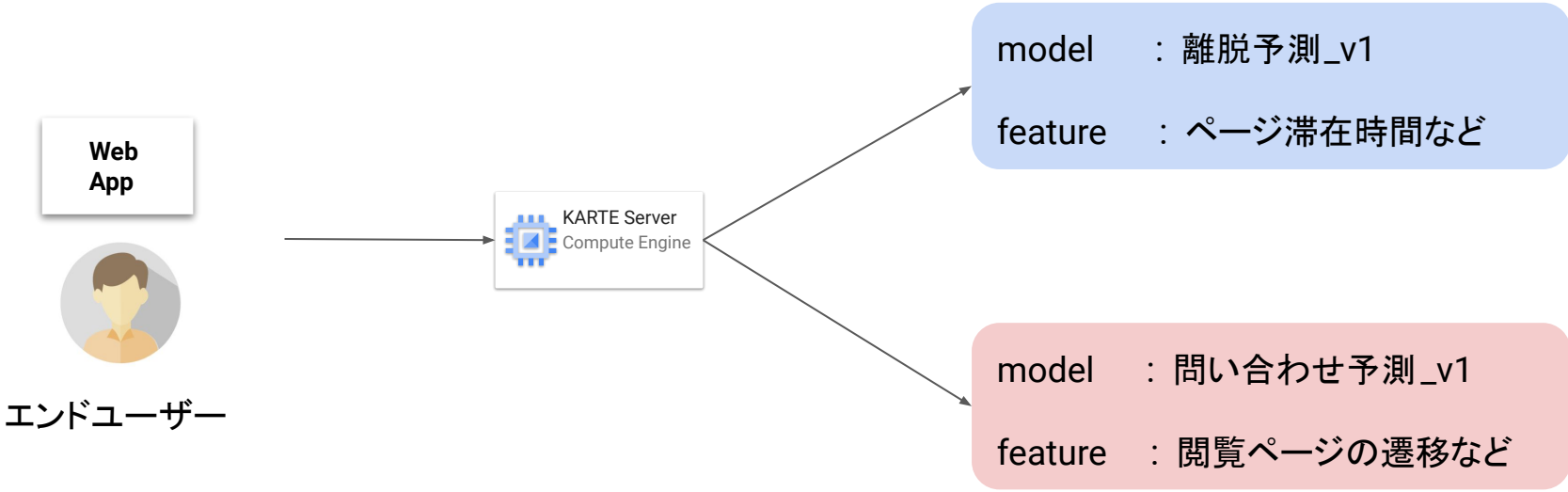
同じタスクのモデルでも
エンドユーザーによって
特徴量が違うモデル

同じ特徴量のモデルでも
お客様が違うモデル

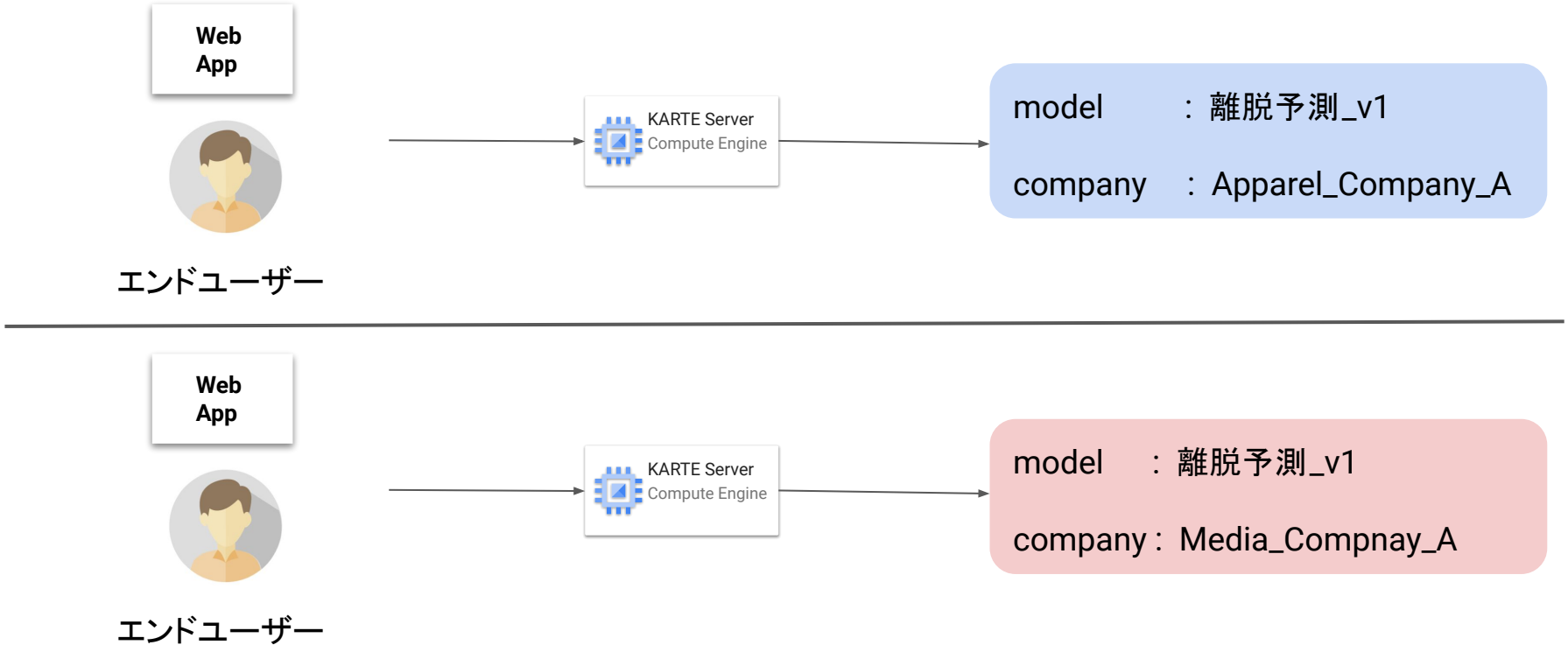
実験と検証のサイクルを素早く回す

モデルの精度だけでなく、
Action まで踏まえたモデルの
価値検証が必要

複数のモデルを同じユーザーに



同じモデルを複数のお客様に



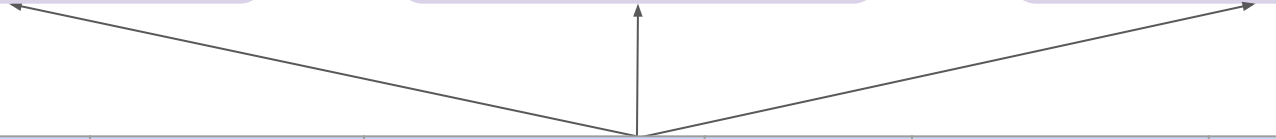
柔軟にデータを取り出せる仕組み

リアルタイム基盤でも生データからSQLで特徴量作成

model　：離脱予測
company：company_A
feature　：閲覧ページの遷移情報など

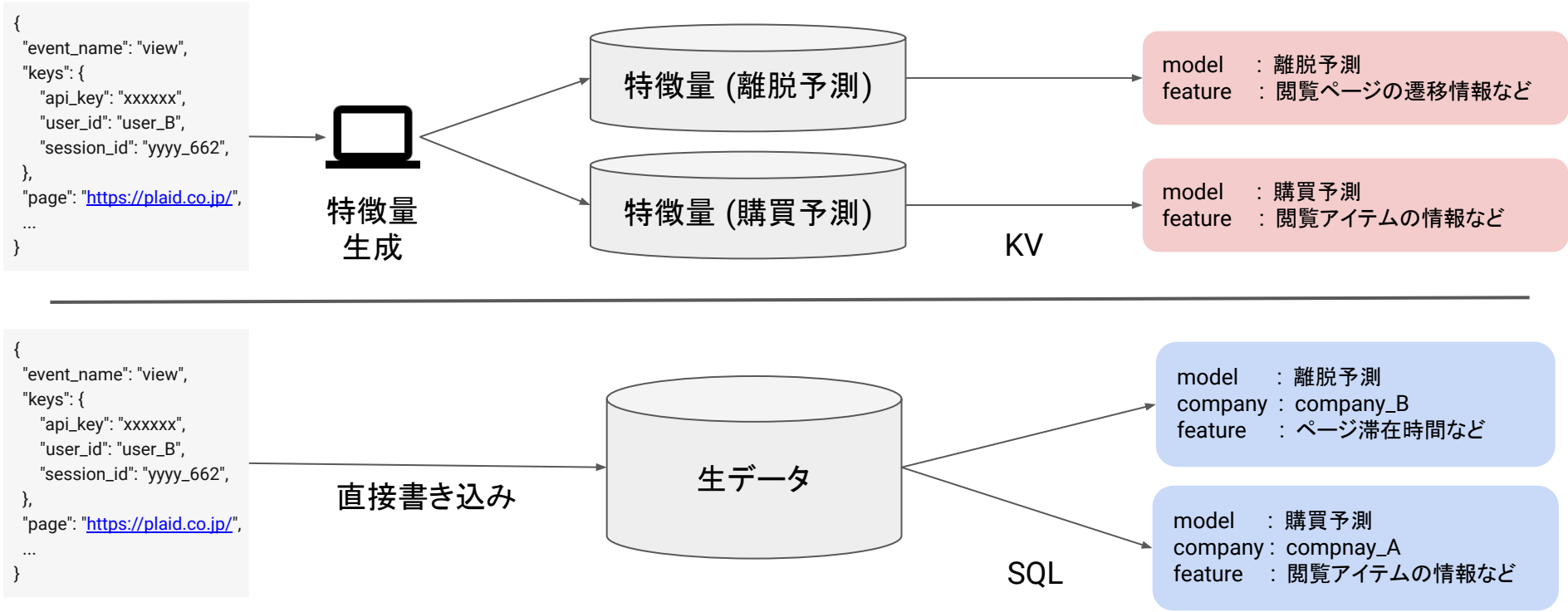
model　：離脱予測
company：company_B
feature　：ページ滞在時間など

model　：購買予測
company：compnay_A
feature　：閲覧アイテムの情報など



company_id	user_id	date	event_name	page	...
company_A	user_A	2020-11-29 23:38:23.010 UTC	view	http://A.com/item/item_A	...
company_A	user_A	2020-11-29 23:40:23.042 UTC	cart		...
company_A	user_A	2020-11-29 23:50:23.142 UTC	buy		...
...	...	...	...	...	...

一般的なリアルタイム基盤との違い



リアルタイム基盤で実現したいこと

リアルタイムデータの活用

リアルタイムに変化する
行動データを保持

適切な範囲の行動データを
リアルタイムに取出

様々なモデルをホストする

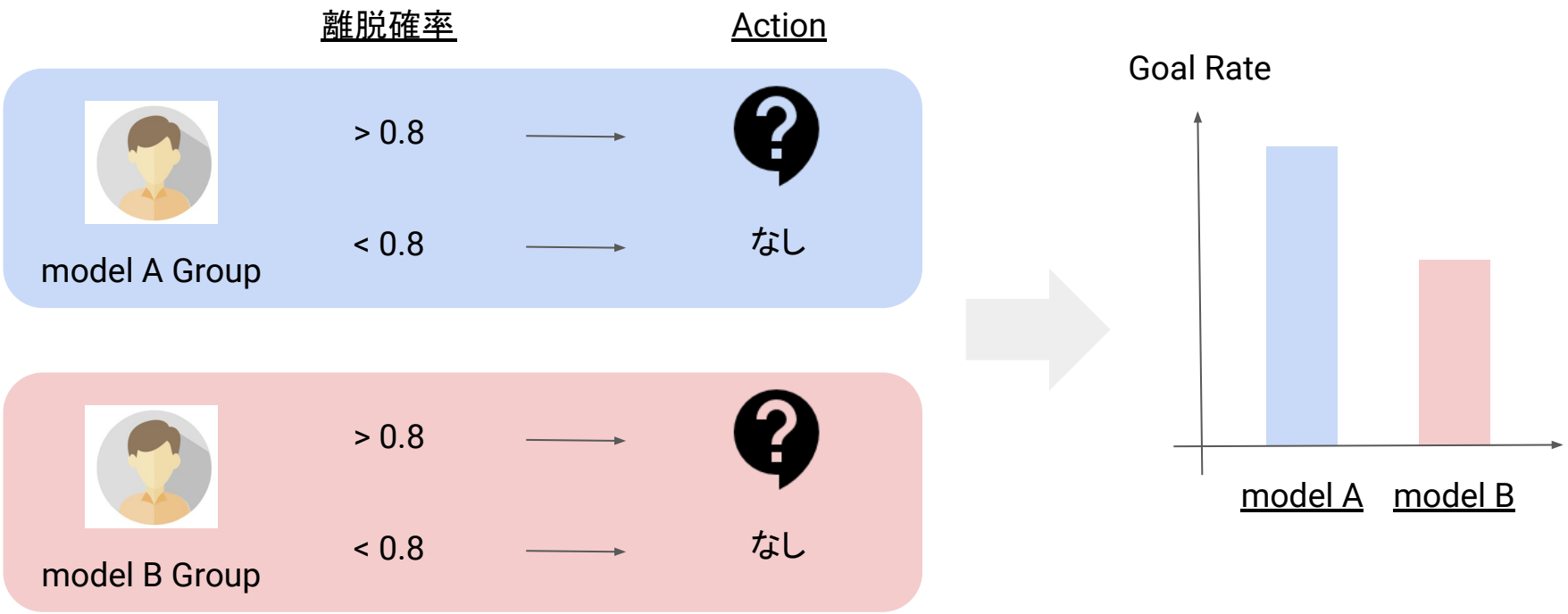
同じタスクのモデルでも
エンドユーザーによって
特徴量が違うモデル

同じ特徴量のモデルでも
お客様が違うモデル

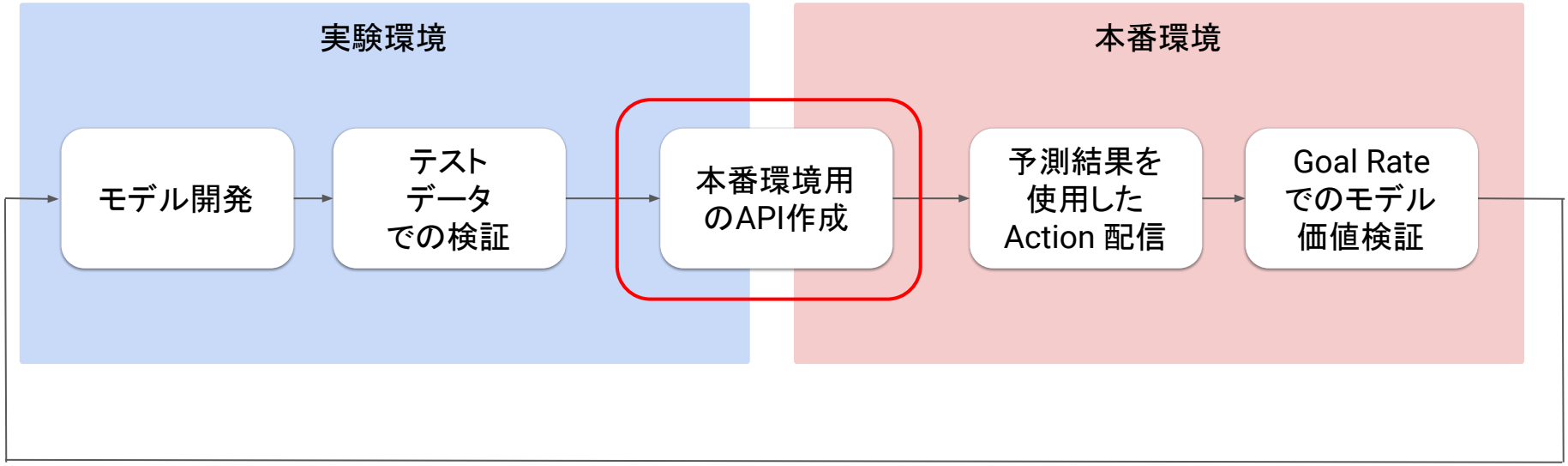
実験と検証のサイクルを素早く回す

モデルの精度だけでなく、
Action まで踏まえたモデルの
価値検証が必要

Action を含めたモデルの価値検証

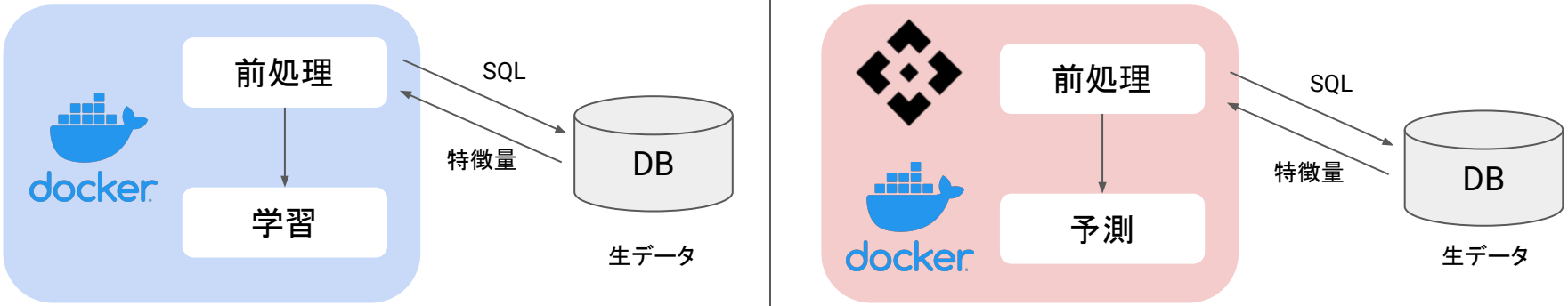


モデルの検証サイクル

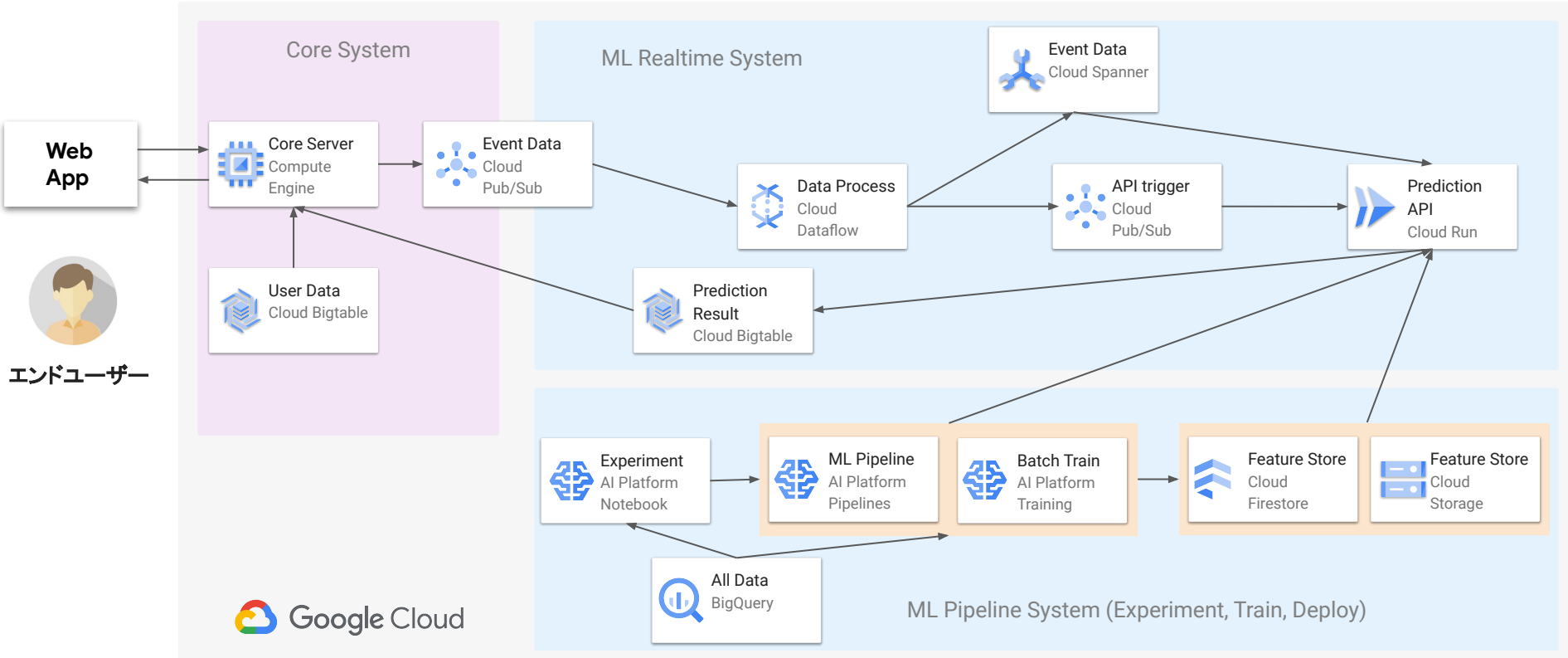

実験環境
本番環境
モデル開発
テスト
データ
での検証
本番環境用
のAPI作成
予測結果を
使用した
Action 配信
Goal Rate
でのモデル
価値検証

実験と本番の環境をなるべく同じに

実験コードをあまり変更せず、本番環境に

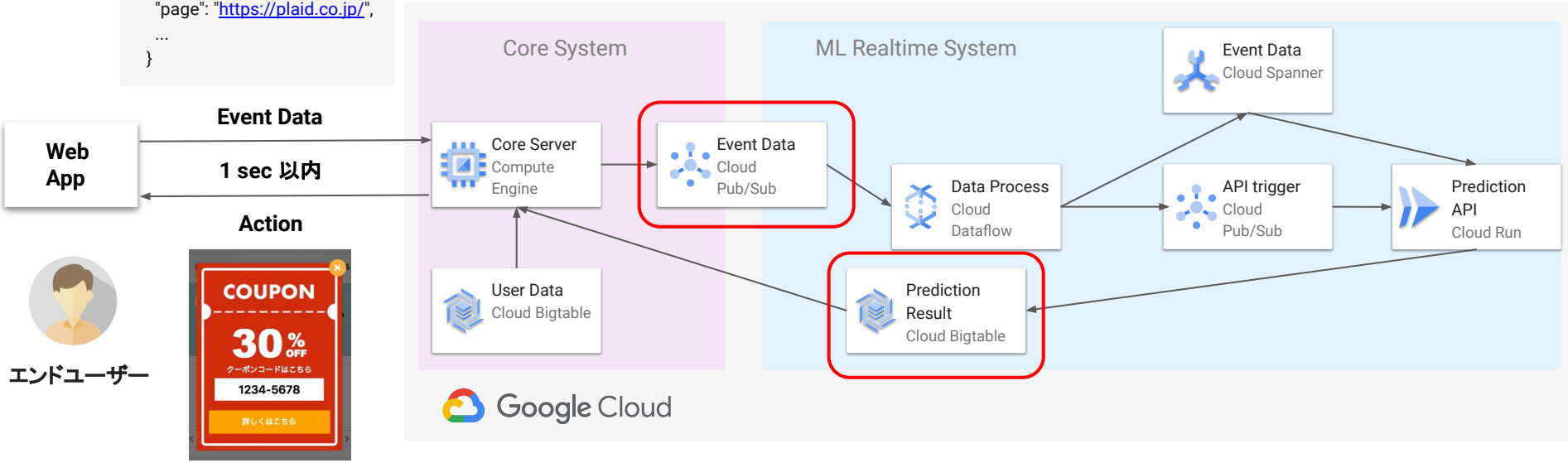


全体構成


Core System
ML Realtime System
Web App
Core Server
Compute Engine
Event Data
Cloud Pub/Sub
Event Data
Cloud Spanner
Data Process
Cloud Dataflow
API trigger
Cloud Pub/Sub
Prediction API
Cloud Run
User Data
Cloud Bigtable
Prediction Result
Cloud Bigtable
エンドユーザー
Experiment
AI Platform Notebook
ML Pipeline
AI Platform Pipelines
Batch Train
AI Platform Training
Feature Store
Cloud Firestore
Feature Store
Cloud Storage
All Data
BigQuery
Google Cloud
ML Pipeline System (Experiment, Train, Deploy)

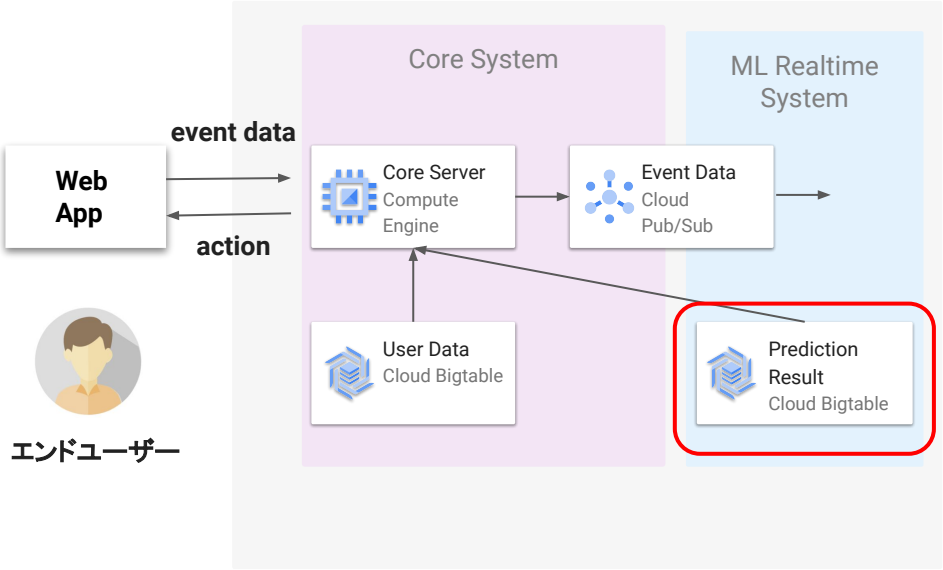
Core System (Not ML) との連携


```
{
  "event_name": "view",
  "keys": {
     "api_key": "xxxxxx",
     "user_id": "user_B",
  },
  "page": "https://plaid.co.jp/",
  ...
}
```

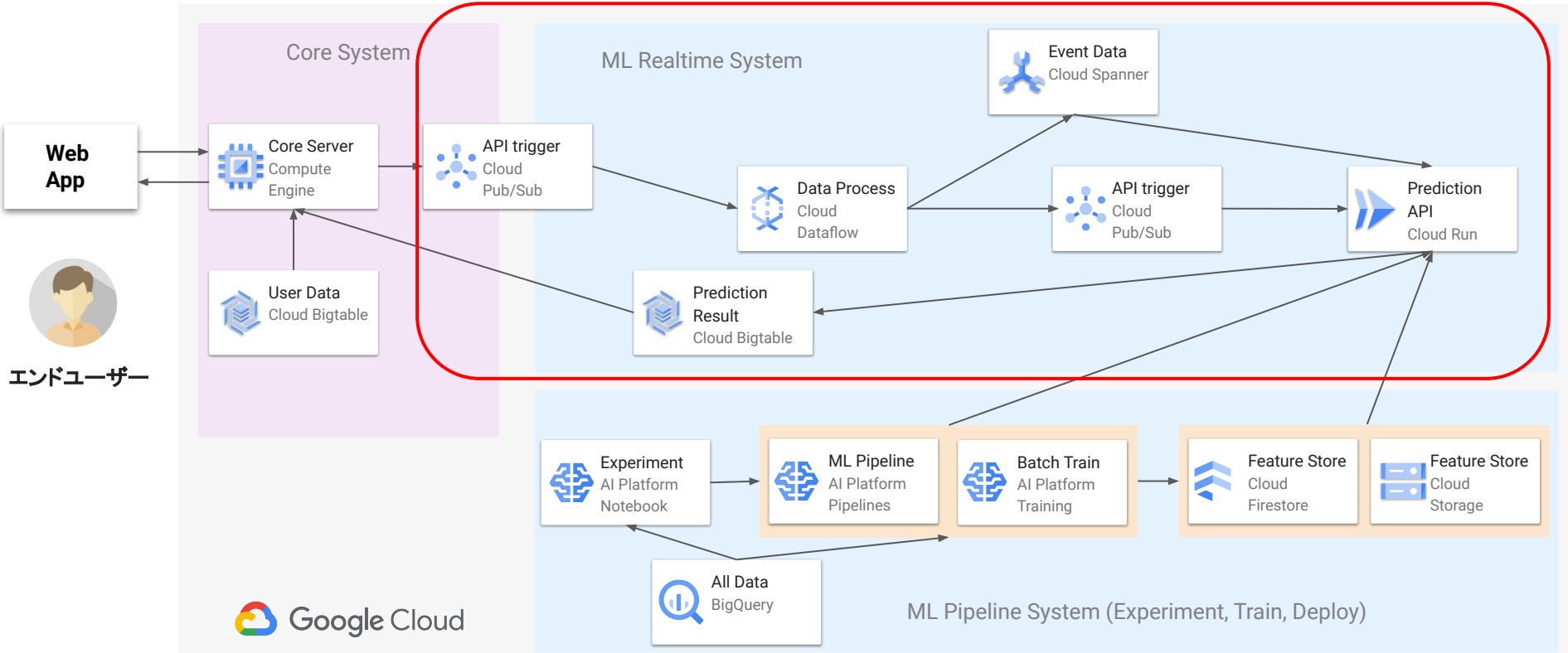



Core と ML の latency の差を吸収

- 予測終了を待たずに action を返す
- 最新の予測結果を DB から読み込む
- 過去の予測結果が使われる可能性
- ML の latency と Core の latency に要件の違いがある



ML Realtime System について



ML Realtime System でのデータの流れ

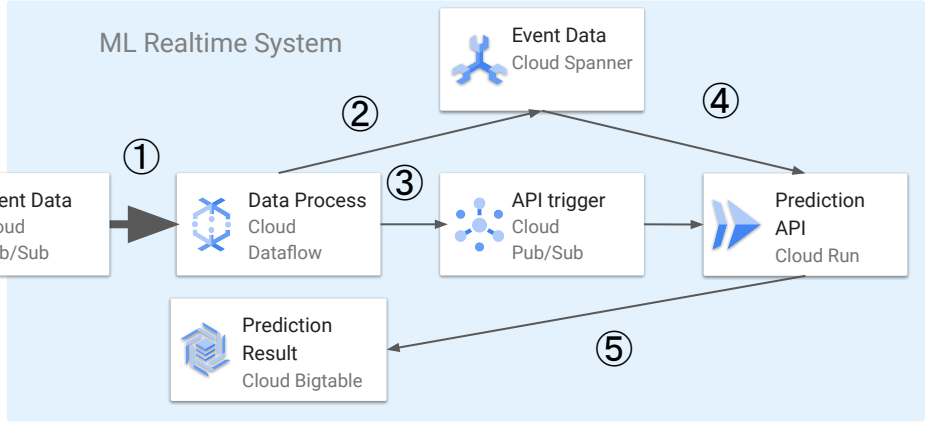
① Core System から event data を受け取る

② event data を DB に保存

③ 特定 event のみを抽出して、Prediction API に リクエスト送信

④ DB から生データを取得し、予測

⑤ 予測結果を DB に書き込み




```
{
  "event_name": "view",
  "sync_date": “2020-11-29 23:50:23.142 UTC",
  "keys": {
     "api_key": "xxxxxx",
     "user_id": "user_B",
     "session_id": "yyyy_662",
  },
  "page": "https://plaid.co.jp/",
  ...
}
```

event data 例

ML Realtime System での今回の焦点

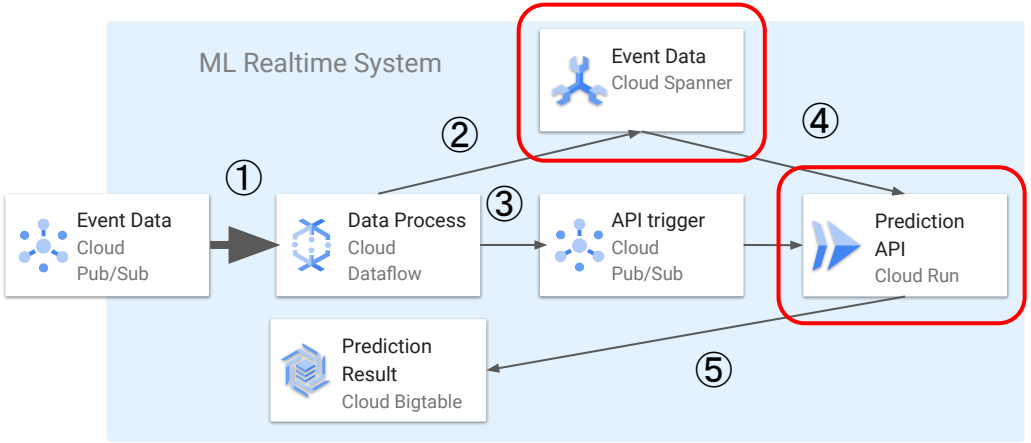
① Core System から event data を受け取る

② event data を DB に保存

③ 特定 event のみを抽出して、Prediction API に リクエスト送信

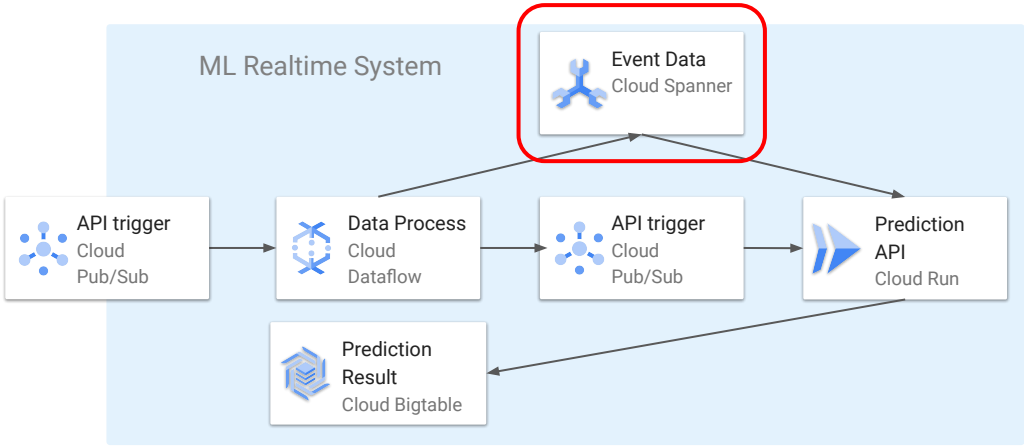
④ DB から生データを取得し、予測

⑤ 予測結果を DB に書き込み

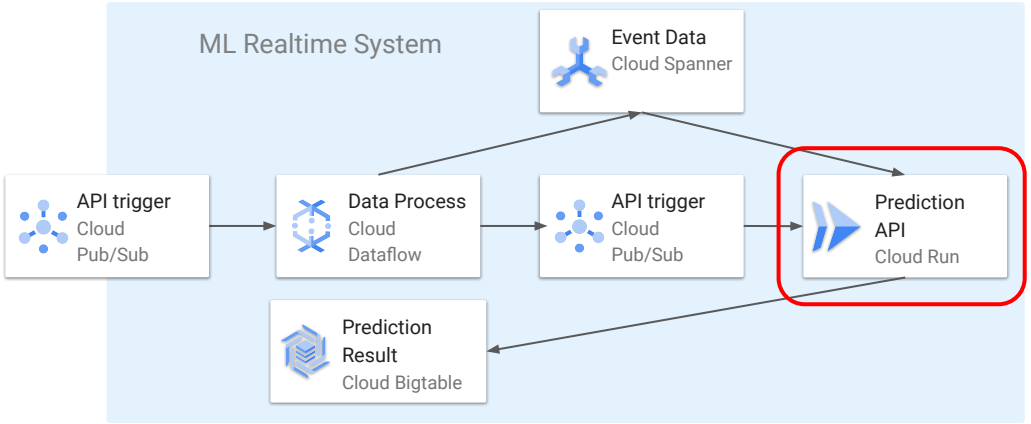


Cloud Spanner を選んだ理由

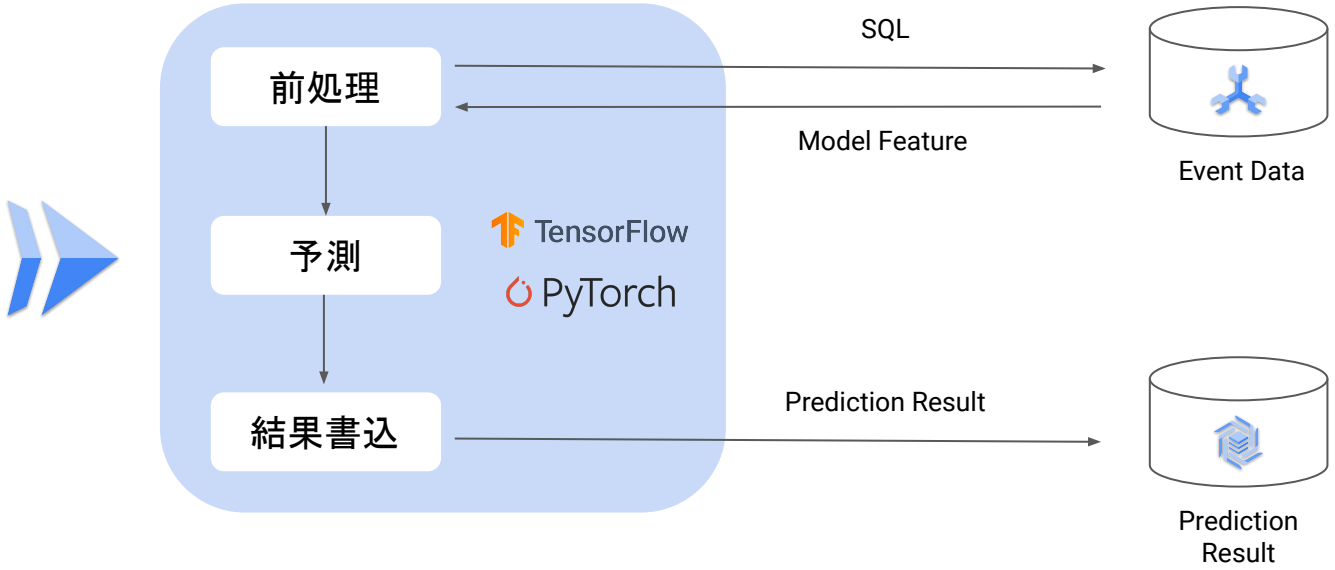
- データを柔軟に取り出せる
  - SQL でデータを取り出せる
- インフラ管理を減らしたい
  - フルマネージド
- 大量リクエストに耐えられる
  - スケーラビリティ



Prediction API について

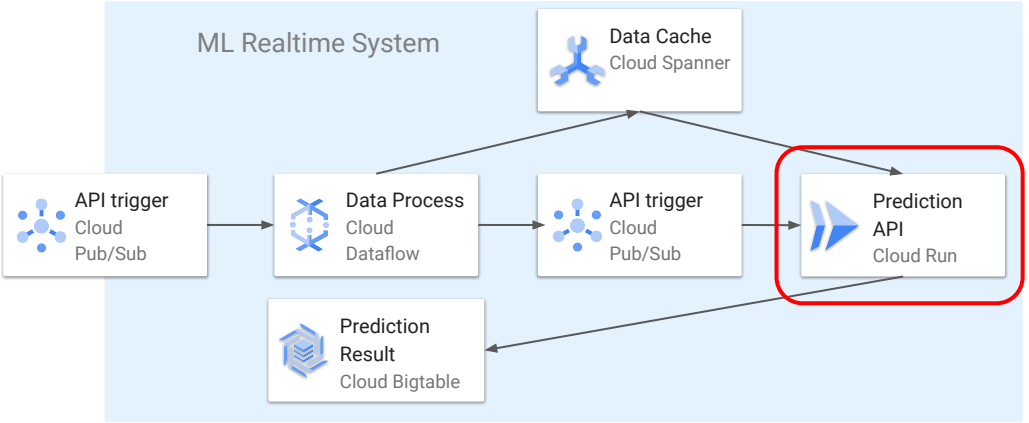


Cloud Run での処理内容



Cloud Run を選んだ理由

- 自由な環境で自由なコードをデプロイ可能
  - Docker が使用可能
- インフラ管理を減らしたい
  - フルマネージド
- 大量リクエストに対するスケーラビリティ



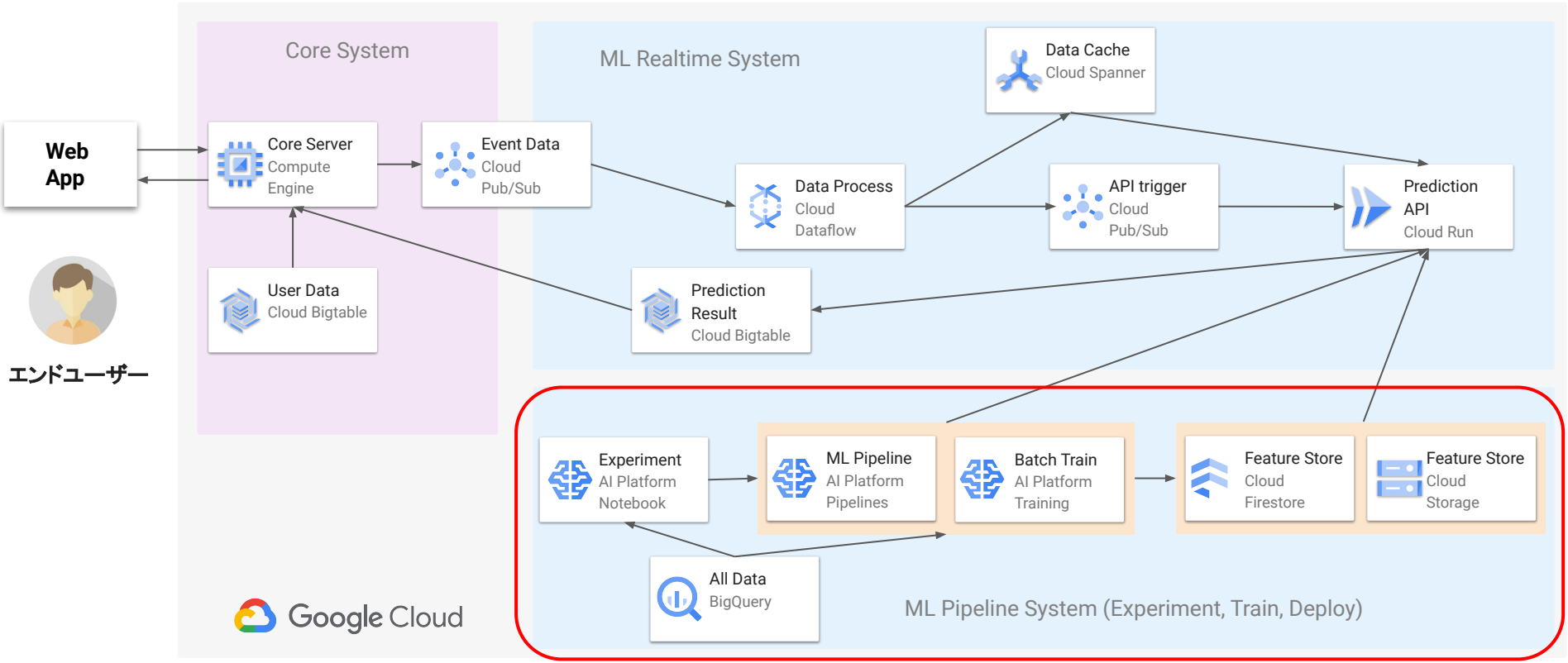
今後の開発予定

- 様々な用途の大規模リアルタイム予測に適した DB の検討
  - Spanner or Bigtable or etc...
- モデル精度の監視
  - モデル精度の変化時にすぐモデル訓練、デプロイが始まる仕組み
- モデル管理
  - 様々な用途、様々なお客様に使用する膨大な数のモデル管理
- モデルを自由にデプロイ
  - 弊社のエンジニアだけでなく、お客様が作成したモデルもホスト可能に

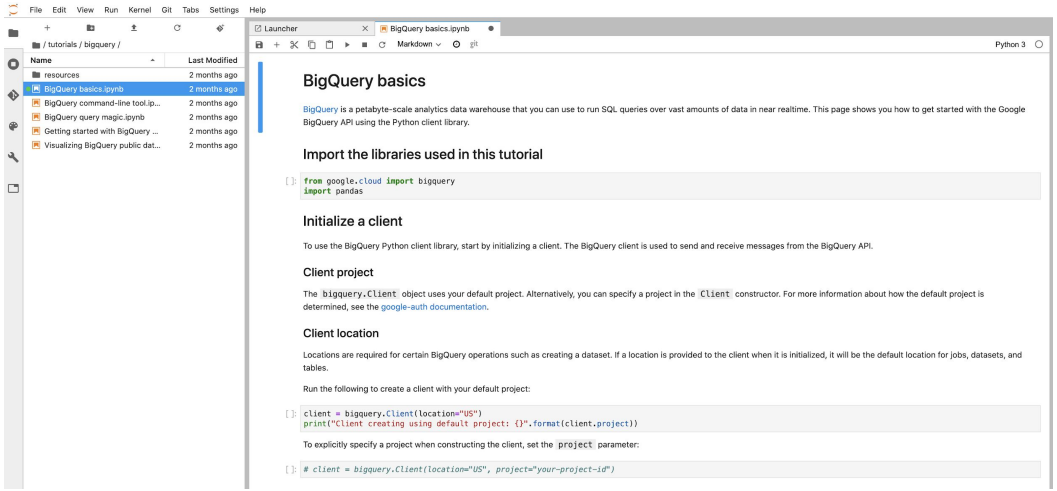
Pipeline System

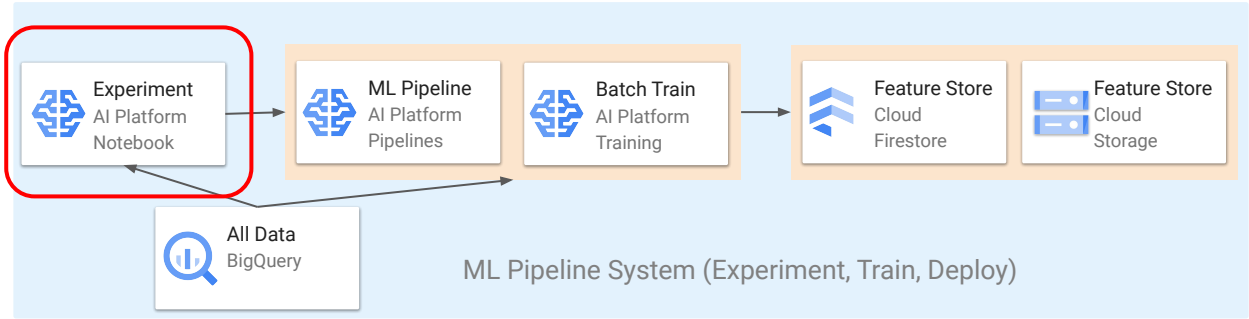
04

ML Pipeline System について

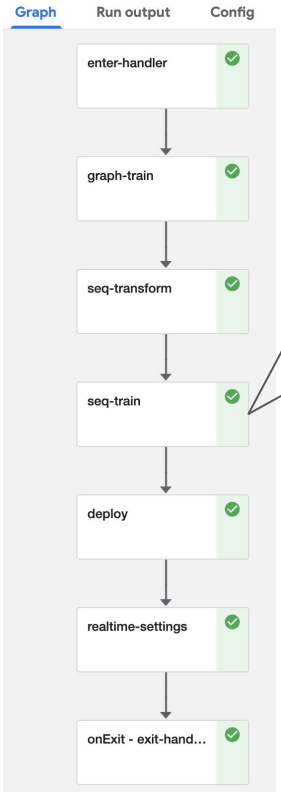


Experiment


File Edit View Run Kernel Git Tabs Settings Help
/ tutorials / bigquery /
Name
Last Modified
resources
BigQuery basics.ipynb
BigQuery command-line tool.ip...
BigQuery query magic.ipynb
Getting started with BigQuery ...
Visualizing BigQuery public dat...
2 months ago
Launcher
BigQuery basics.ipynb
Markdown
Python 3
BigQuery basics
BigQuery is a petabyte-scale analytics data warehouse that you can use to run SQL queries over vast amounts of data in near realtime. This page shows you how to get started with the Google BigQuery API using the Python client library.
Import the libraries used in this tutorial
from google.cloud import bigquery
import pandas
Initialize a client
To use the BigQuery Python client library, start by initializing a client. The BigQuery client is used to send and receive messages from the BigQuery API.
Client project
The bigquery.Client object uses your default project. Alternatively, you can specify a project in the Client constructor. For more information about how the default project is determined, see the google-auth documentation.
Client location
Locations are required for certain BigQuery operations such as creating a dataset. If a location is provided to the client when it is initialized, it will be the default location for jobs, datasets, and tables.
Run the following to create a client with your default project:
client = bigquery.Client(location="US")
print("Client creating using default project: {}".format(client.project))
To explicitly specify a project when constructing the client, set the project parameter:
client = bigquery.Client(location="US", project="your-project-id")


Experiment
AI Platform
Notebook
ML Pipeline
AI Platform
Pipelines
Batch Train
AI Platform
Training
Feature Store
Cloud
Firestore
Feature Store
Cloud
Storage
All Data
BigQuery
ML Pipeline System (Experiment, Train, Deploy)

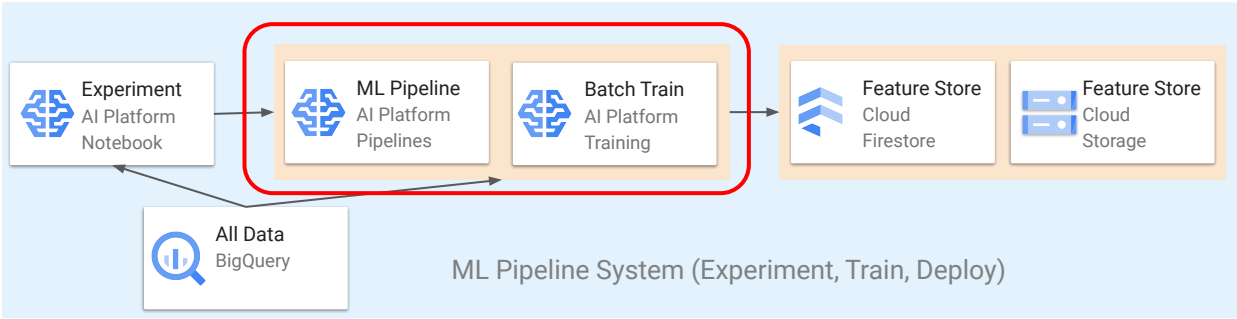
Training



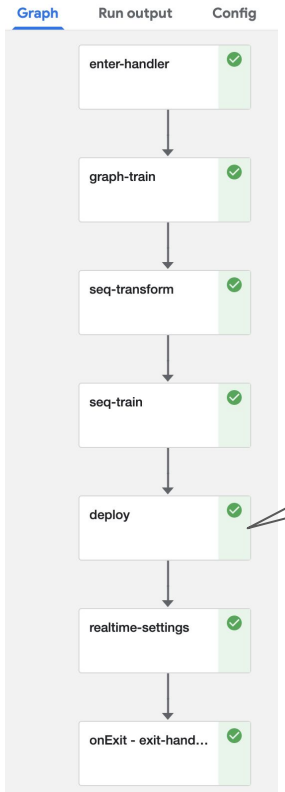
Batch Train
AI Platform
Training

- 柔軟なインスタンスサイズの指定
- Job 実行時間分だけの課金

```
$ gcloud ai-plarform jobs submit training
  --master-machine-type=”n1-highmem-8”,
  --master-accelerator=”type=nvidia-tesla-k80,count-2”
```



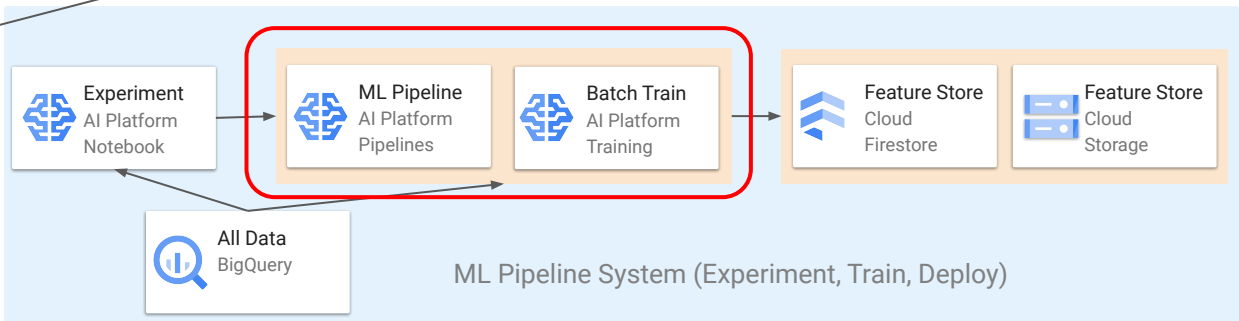
Deploy



- cloud-sdk の image を使用
- コマンド 1 つでデプロイ可能

Prediction API
Cloud Run

```
$ gcloud run deploy model_A
  --image=gcr.io/hoge/hoge:latest
```



まとめ

まとめ

- 以下の要件を考慮してリアルタイム予測基盤を作成
  - リアルタイムデータの活用
  - 様々なモデルをホストする
  - 実験と検証のサイクルを素早く回す

- Cloud Spanner を 時系列データ用の DB として使うことでスケーラビリティをもちながら柔軟にデータを取り出せる

- Cloud Run を前処理から予測までを行う API として活用

- AI Platform 製品を柔軟に使用した Pipeline 基盤の活用

Thank you