FreakOut における AWS 上での機械学習活用事例

株式会社フリークアウト
西口 次郎 CTO
小浜 翔太郎 Software Engineer

FreakOut HD Inc.,

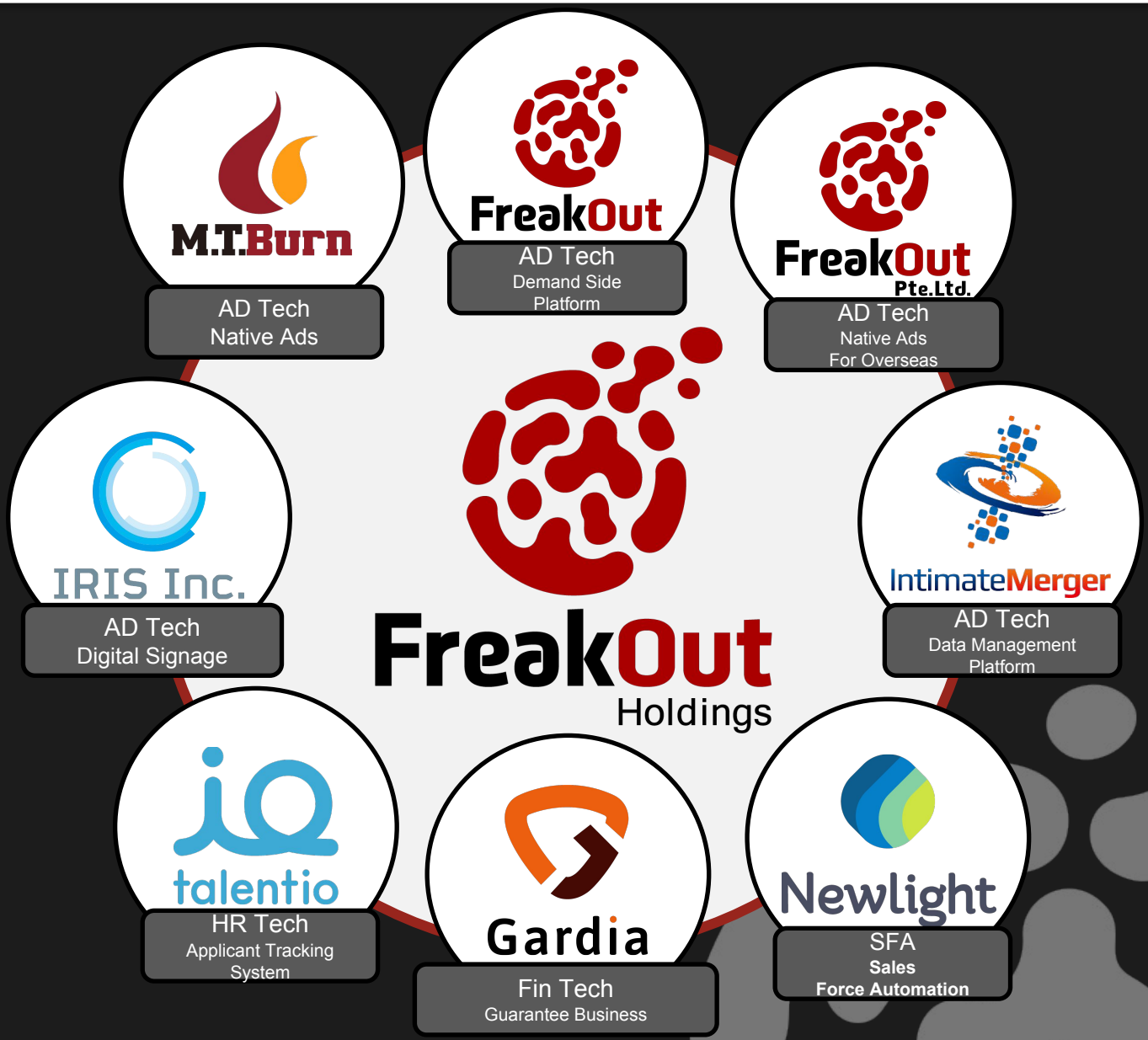
Disrupter by Learning Software

Founded : 2010/10/01

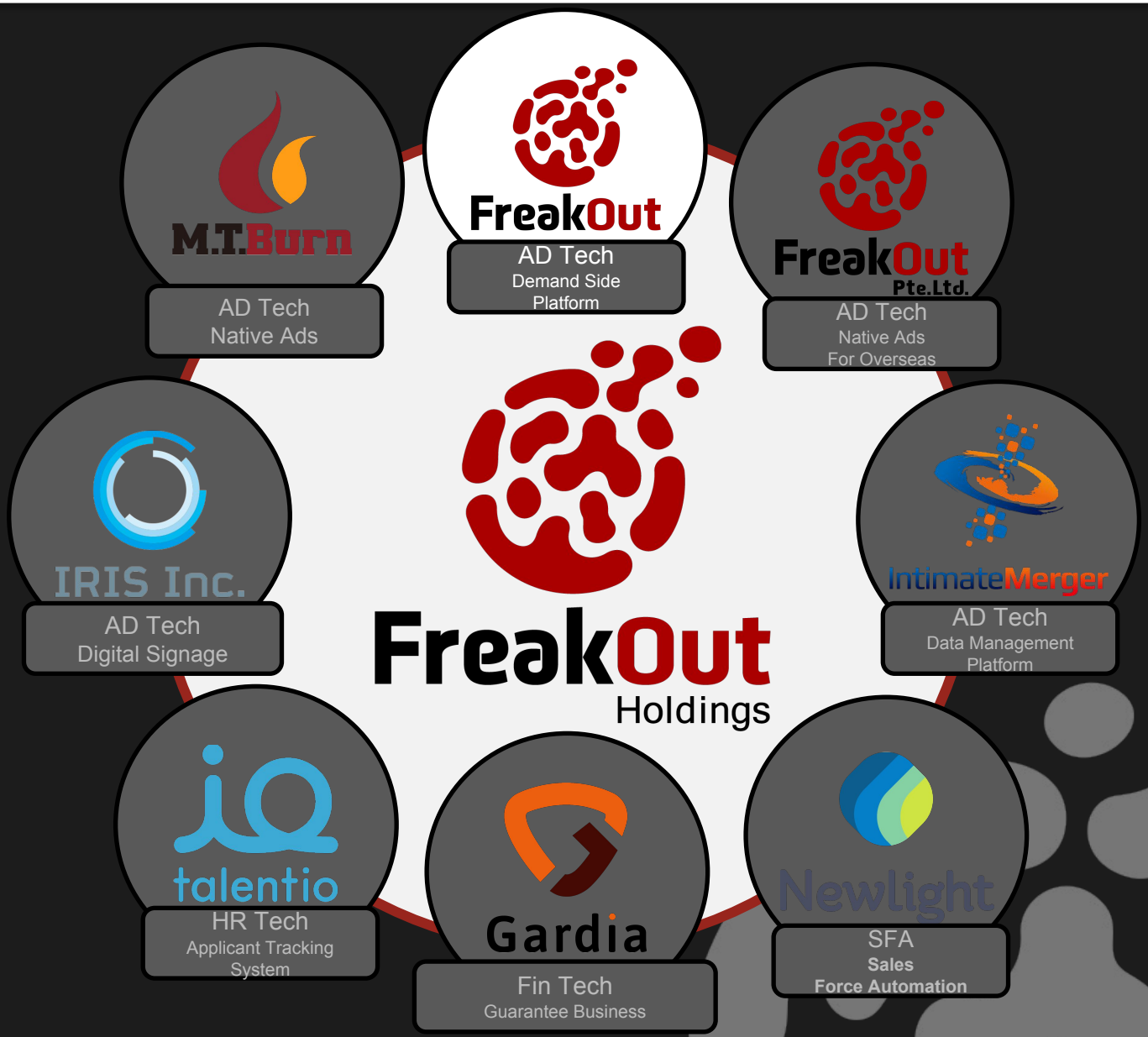
Tokyo | Osaka | Fukuoka
Singapore | Bankok | Jakarta | Istanbul | Taiwan | Manila |
Kuala Lumpur | Ha noi | Tehran | Gurugram | Shanghai

- The First DSP in JAPAN
- Now, AI Based Software Company
- Team: 180 + 150
- Client: 15,000+
- IPO: 2014/06/24

FreakOut HD Inc.,



FreakOut Inc.,



FreakOut DSP/Red

FreakOut

適正な価格で広告主様にとって必要な広告表示機会だけを買い付け、広告効果を最大化するデマンドサイドプラットフォーム。
モバイル向けには、月間 3,600 億インプレッションに及ぶ業界最大級のモバイル広告枠在庫を保有。

Red for Publishers

FreakOut

デジタルメディア様のアドプラットフォームを構築し、
CPM 単価の高いプレミアム広告枠の開発と広告事業に必要な
販売・オペレーションまで全てを一気通貫で支援。

Poets

FreakOut

コンテンツ UI と親和性の高い広告フォーマットを活用した、
ユーザー体験を損なわずに広告体験を提供することができる
アドプラットフォーム。
情報接触の場がモバイル中心にシフトしていく中で、最適な広
告体験を提供。

ASE

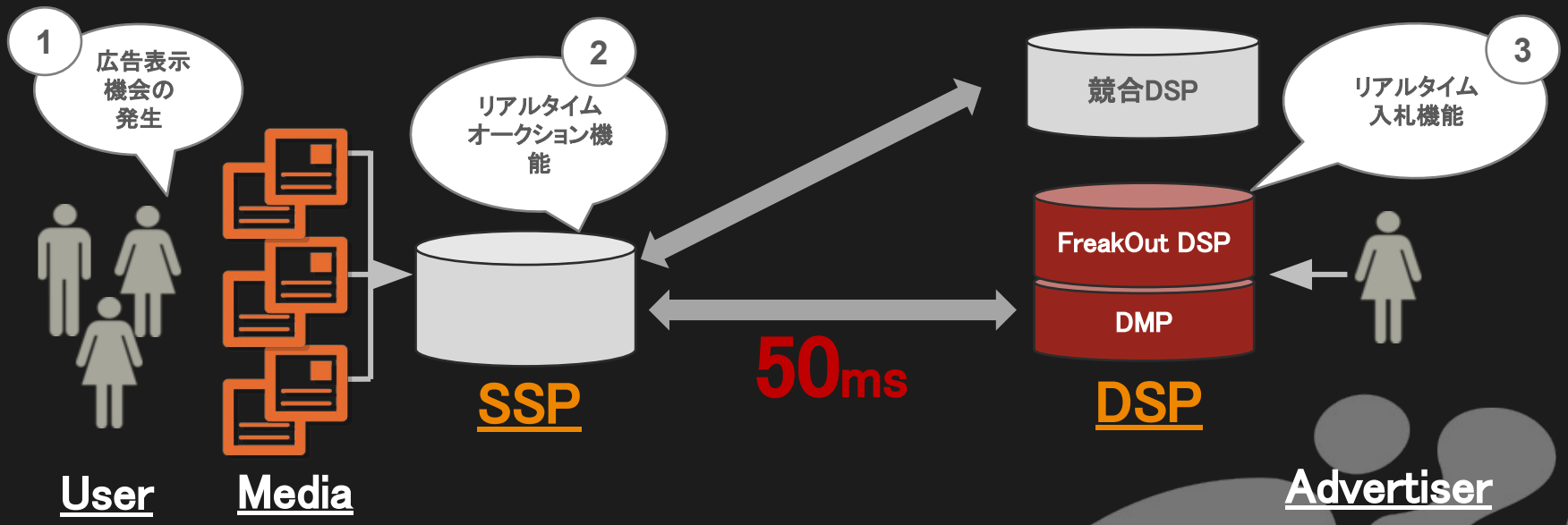
FreakOut

位置情報マーケティングプラットフォーム。
国内最大規模の位置情報データベースを保持。
流通・小売事業者の集客・顧客分析の効果及び利便性向上を
実現するマーケティングプラットフォーム。

DSP

FreakOut

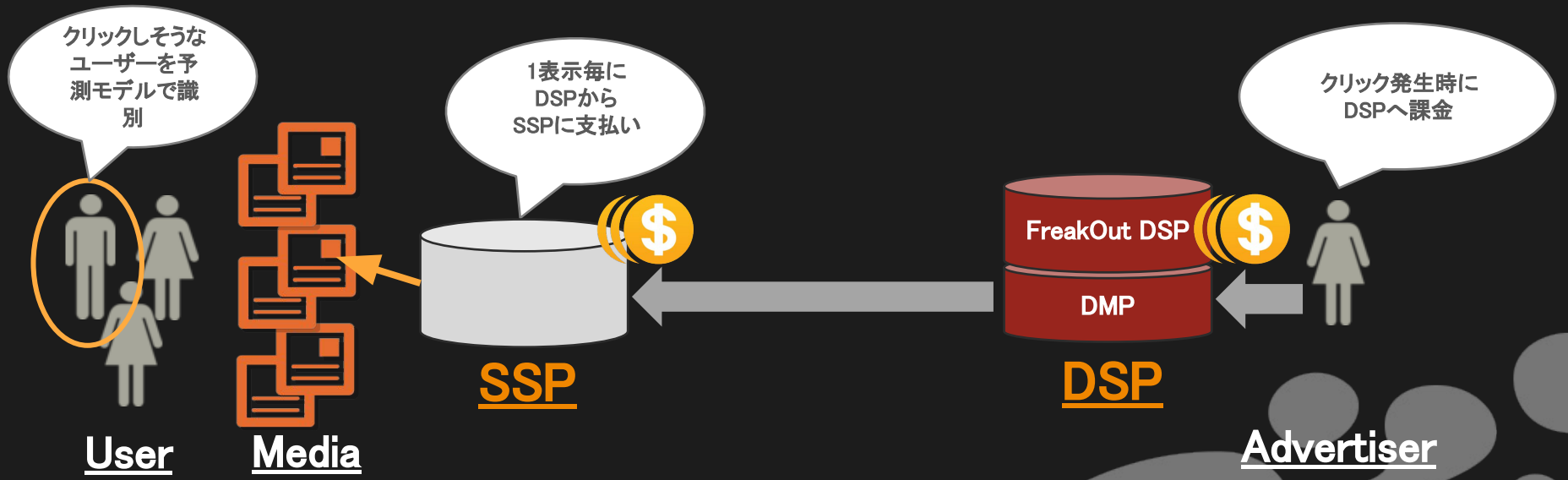
広告「枠」が「人」に見られた瞬間に取引
インプレッション（広告の閲覧行為）が入札制で自動売買



DSP

FreakOut

170億/day、ピーク時 29万/sec の
大規模なリクエストから価値あるリクエストを見つけ、
適切な入札価格を決定するために機械学習を利用



Partners

FreakOut

国内・国外の多くの SSP と接続済

Infrastructure (DSP)

- オンプレミス環境
    - その他のプロダクトではクラウドをメイン利用
- サーバ 700 台〜
- 5,200 億 Bid Request / month
- Server Side: Perl
- Analysis/ML: C++, Python, Scala, etc..

Log Analysis (DSP)

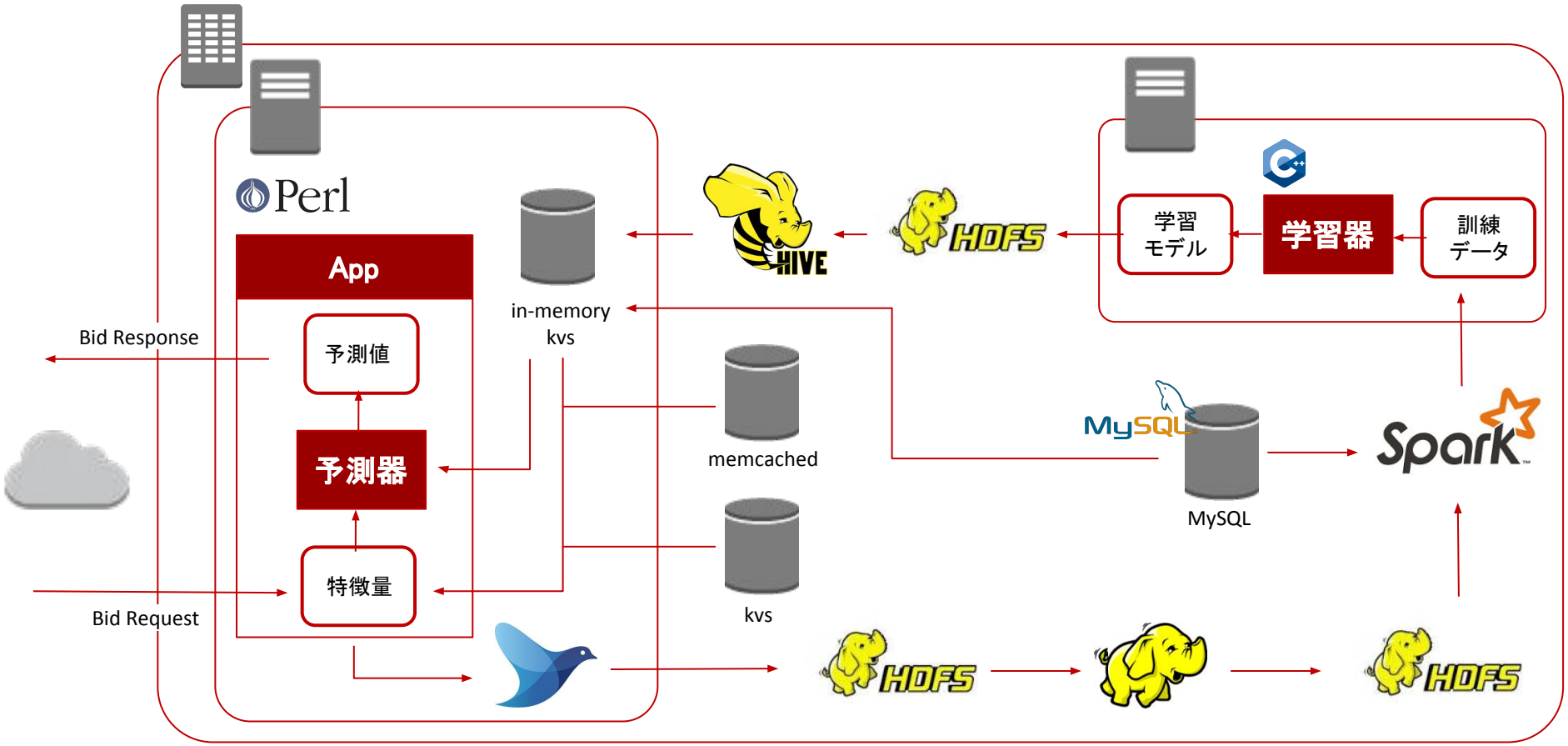
- Hadoop を利用 (Cloudera Distribution)
    - Hive, Spark, Presto, MapReduce, etc..
- Total 2PB ～
- AWS 環境に移行計画中

Machine Learning in FreakOut, Inc.

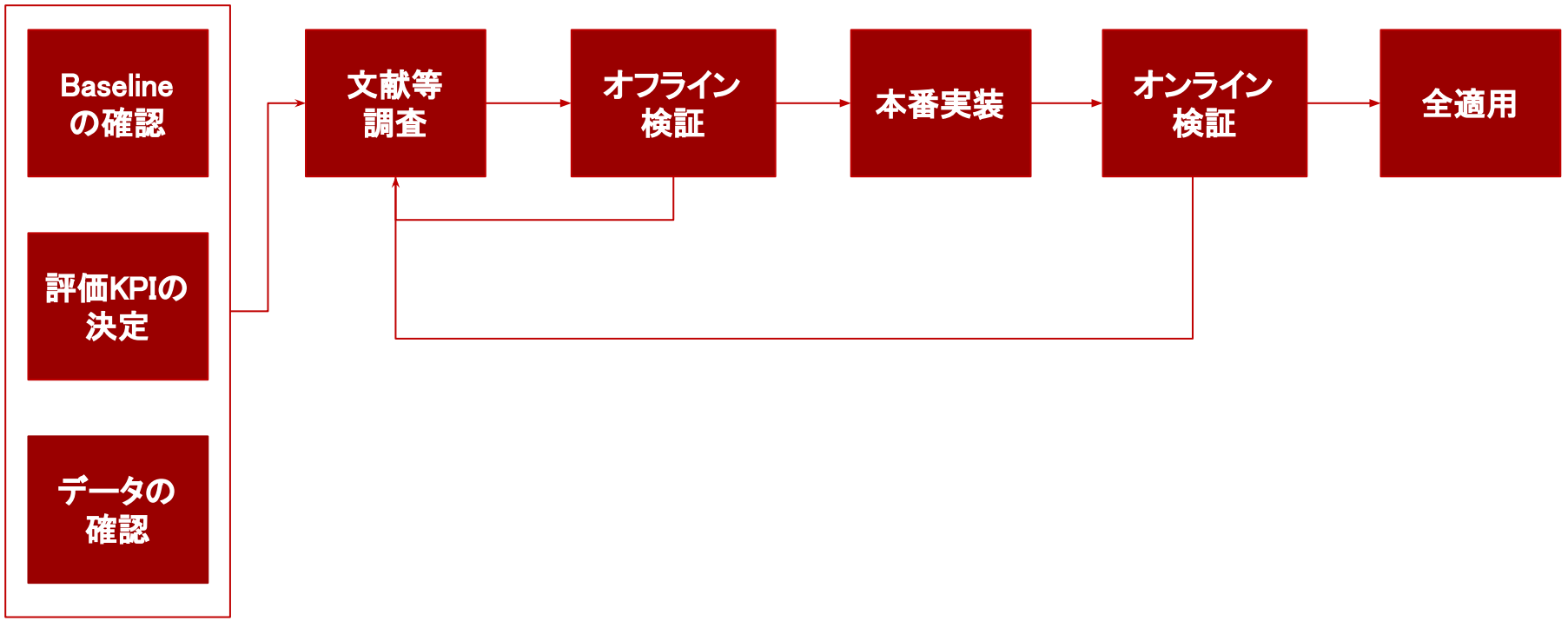
適切な入札価格を決定するために **CTR/CVR予測** を利用

CTR = Click Through Rate
CVR = Conversion Rate

学習モデルはバッチで更新し、予測はアプリケーション内で行う



CTR/CVR 予測モデル改善の流れ



オフライン検証の環境の変化

R&D チームの人数の増加と共に、オフライン検証の環境も変化

2014年

オンプレミスの環境に分析サーバを用意

2016年

EC2 の利用を始める Vagrant AWS Provider を使い
開発で使い慣れているコマンドで
インスタンスの操作ができるようにした

2018年

Amazon SageMaker により大規模で
並列にオフライン検証が行えるようになった

オフライン検証におけるAmazon SageMaker の利用

FreakOut

主要な3つのコンポーネントから構成される

Authoring
前処理用の Notebook instance を簡単に立ち上げられる

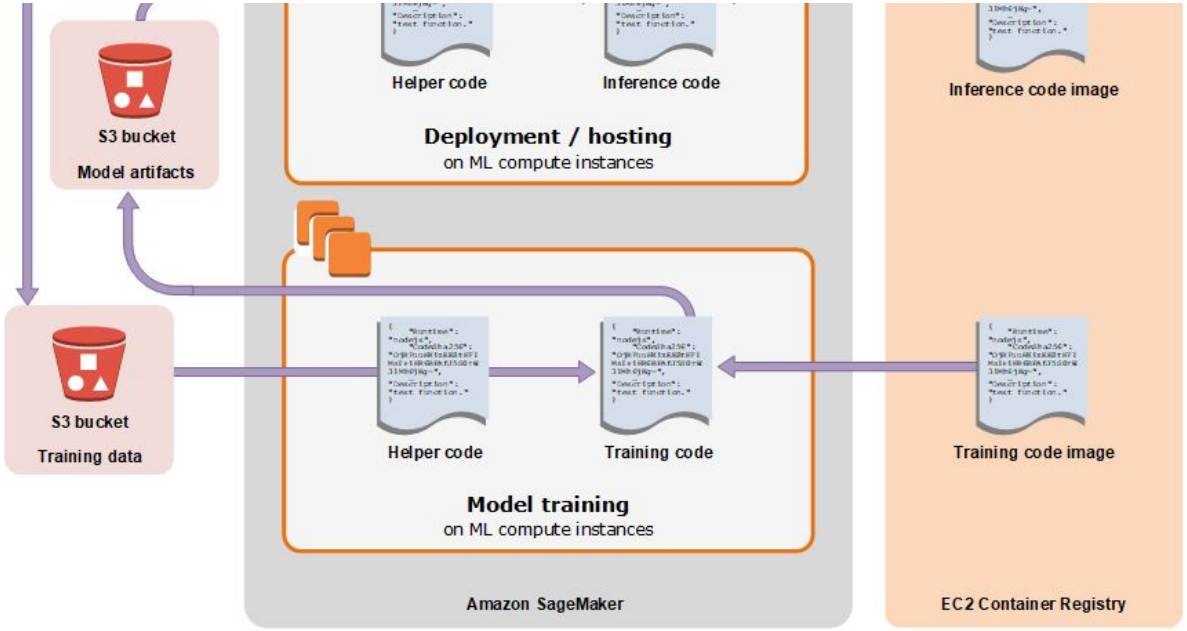
Model Training
モデル学習 Job を簡単に実行

Model Hosting
学習したモデルを利用したエンドポイントを簡単に作成

現在は **Model Training** のところを主に利用している

主に以下の4つを指定するとモデル学習 Job が実行できる

- 学習アルゴリズムが入った Docker Image の ECR パス
- 訓練データなどの入力用 S3 パス
- 学習に利用するハイパーパラメータ
- 学習済みモデルなどの出力用 S3 パス



https://docs.aws.amazon.com/sagemaker/latest/dg/how-it-works-training.html

FreakOut

有名な学習アルゴリズムは Built-in で Image が用意されている

- XGBoost Algorithm
- Factorization Machines
- Image Classification Algorithm (ResNet)
- Sequence2Sequence

...

Built-in のアルゴリズムだけでなく

Docker image を用意することで任意の学習アルゴリズム

を SageMaker 上で動かすことができる

Model Training with Custom Image

- Job を submit すると以下が実行される

```
docker run image train
```

- 訓練データなどの入力用 S3 パスは、FILE mode の場合
  以下のパスにマウントされる

```
/opt/ml/input/data/channel_name
```

- ハイパーパラメータは、以下のパスに JSON 形式で置かれる

```
/opt/ml/input/config/hyperparameters.json
```

- 学習済みモデルなどは、以下のパスに置くと、S3に保存される

```
/opt/ml/model/
```

Model Training with Custom Image

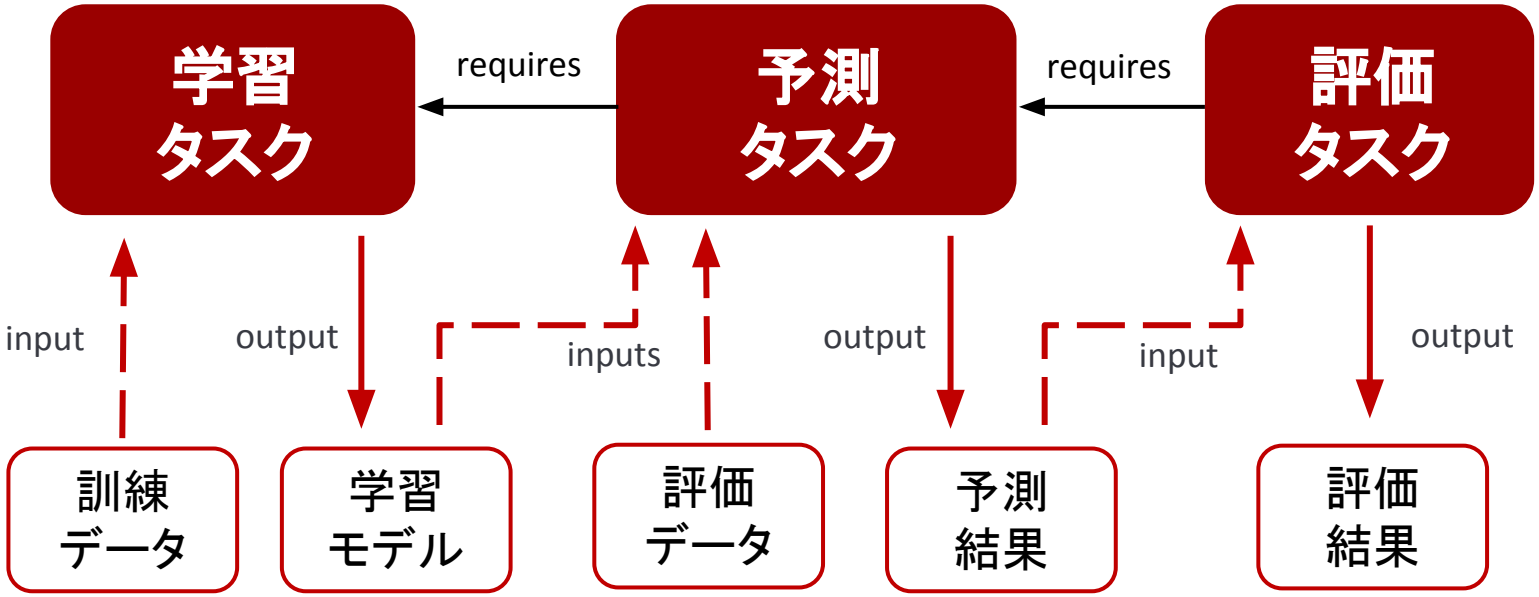
`docker run image train` で以下を実行する
イメージをオフライン検証用に用意した

- マウントされた訓練・評価データの連結と移動
- hyperparameter.json を luigi.cfg に設定
- 学習・評価を行う luigi ワークフローの実行
- モデル・評価結果をS3に保存されるように移動

Luigi Train, Prediction and Evaluation Workflow

FreakOut

Luigi は Spotify を中心に開発している**ワークフロー管理ツール**

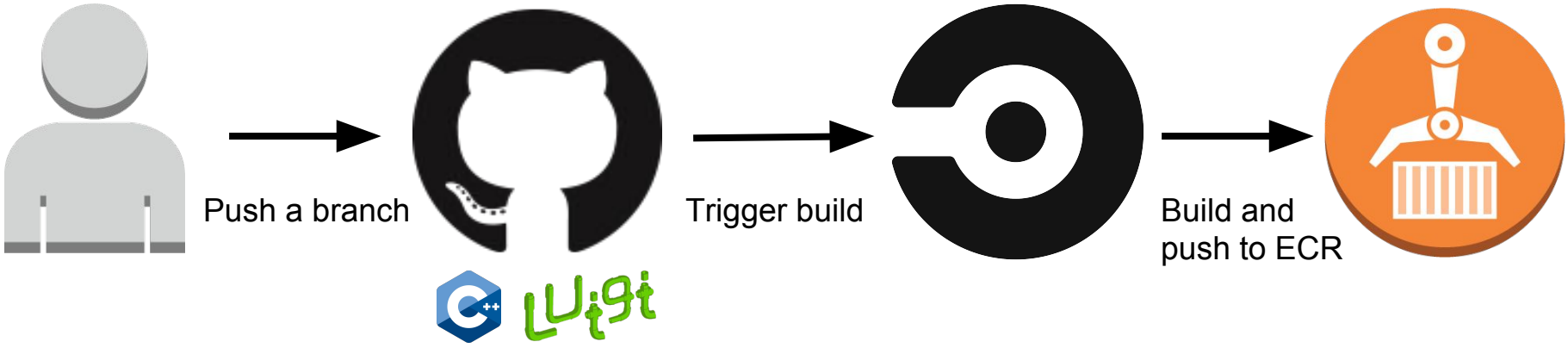


luigi.DictParameter を使うと hyperparameters.json が
JSON のまま luigi.cfg 経由で**タスクに簡単に渡すことが可能**

Luigi Train, Prediction and Evaluation Workflow

https://github.com/shotarok/vw-luigi

Docker image は CircleCI で build し ECR で管理



SageMaker を用いたオフライン検証手順

SageMaker を使うことで効率的にオフライン検証が可能に

1. 訓練・評価データを抽出し HDFS 上に保存する
2. オンプレ HDFS 上のデータを S3 に保存する
3. Notebook instance を立ち上げ Job を submit
4. Notebook instance で結果を確認し比較する

1. 訓練・評価データを抽出し HDFS 上に保存

FreakOut

データ量の増加や抽出条件の複雑化や堅牢性・保守性
を高めるため、データ抽出方法は以下のように改善してきた

Phase 1: HiveQL

- 2014 Spring ~

要因	評価
Efficiency	😢
Testablility	😢
Maintainablility	😄

Phase 2: naive な Spark App

- 2017 Spring ~

要因	評価
Efficiency	😄
Testablility	😄
Maintainablility	😢

Phase 3: 汎用化した Spark App

- 2018 Spring ~

要因	評価
Efficiency	😄
Testablility	😄
Maintainablility	😄 (ときに 😢)

FreakOut

訓練・評価データを **Spark** を利用して抽出する

抽出に使う Hive
テーブルの条件

抽出する特徴量の設定

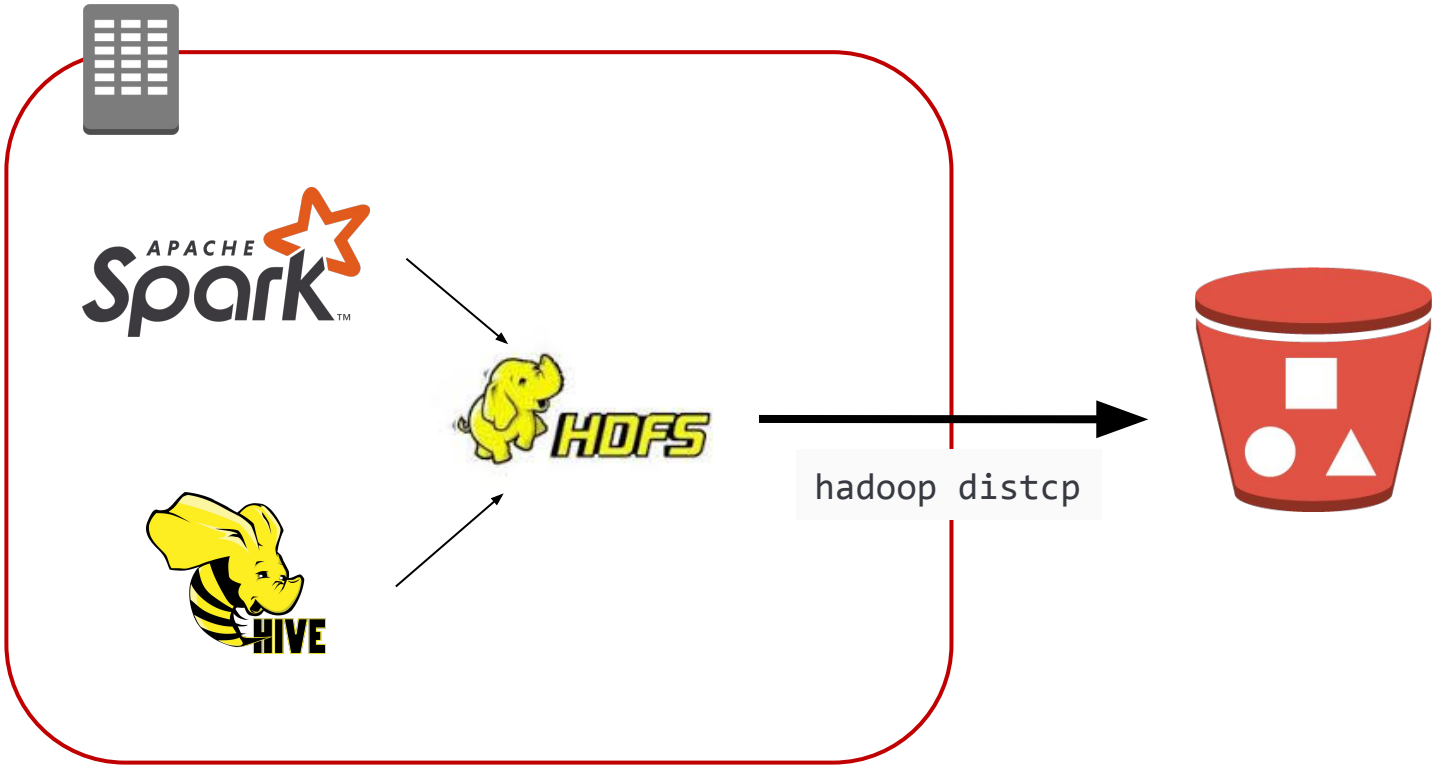
データに対して行う
フィルターの設定

抽出条件を書いた YAML から
データ抽出する Application を用意

```
sample_type: imp-click
logs_tables:
    ad:
        last_hour: “2017-10-15T12:00:00”
        duration_hour: 24
        conditions:
            -  “billing_type = 1”
            -  “is_application = 1”

    …

features:
    - ad_id
    - hour
    - spot_id

    …

remove_anormaly_cvr_cpns: true
performance_indicator_type: ‘cpa’

    …
```

2. オンプレ HDFS 上のデータを S3 に保存する

`hadoop distcp` を利用して抽出したデータを S3 上に移動



訓練・評価データは、別のパスに置いて job を submit する際
それぞれ **train/test** Channel として指定できるようにする

3. Notebook instance を立ち上げ job を submit

FreakOut

```
import boto3

import sagemaker as sage

sess = sage.Session()
```

train/test Channel の指定

```
input_config = [{
    'ChannelName': 'train',
     **sage.session.s3_input(f's3://{bucket}/sagemaker_test/train/').config
}, {'ChannelName': 'test',
     **sage.session.s3_input(f's3://{bucket}/sagemaker_test/test/').config
}]
```

Job の submit

```
sess.train(image
           , input_mode
           , input_config
           , role
           , job_name
           , output_config
           , resource_config
           , hyperparameters
           , stop_condition)
```

4. Notebook instance で結果を確認し比較する

FreakOut

sagemaker.Session.log_for_job
で job の進行を確認可能

```
In [0]: sess.logs_for_job('test-sagemaker-fout-fm-201804271647-mlc518xlarge-71', wai
* 2 present dependencies were encountered:
    - 2 ExternalInputText(path=/tmp/work/test.txt,/tmp/work/train.txt)
* 3 ran successfully:
    - 1 EvalTask(...)
    - 1 LearnTask(...)
    - 1 PredictTask(...)

This progress looks :) because there were no failed tasks or missing externa

===== Luigi Execution Summary =====

Copy result and model files to a persistent dir
Print evaluation results
the average negative log likelihood on all samples is: 0.12495198084074835
the average negative log likelihood on positive samples is: 2.79229466931399
the average negative log likelihood on negative samples is: 0.03807588903755
the auPR is: 0.14016833419421373
the auROC is: 0.7573931517071275
===== Job Complete =====
Billable seconds: 513
```

S3 に保存した評価結果を
取得し比較を行う

```
In [7]: import tarfile
def print_result(localpath):
  tar = tarfile.open(localpath, "r:gz")
  for member in tar.getmembers():
    if member.name != 'result.txt':
      continue
    with tar.extractfile(member) as f:
      print("".join(map(lambda s: s.decode("utf-8"), f.readlines())))

s3client = sess.boto_session.client("s3")
job_name = 'test-sagemaker-fout-fm-201804271647-mlc518xlarge-71'
s3path = s3path_tmpl.format(job_name)
localpath = '/tmp/{}.tar.gz'.format(job_name)
s3client.download_file(bucket, s3path, localpath)
print(job_name)
print_result(localpath)

test-sagemaker-fout-fm-201804271647-mlc518xlarge-71
the average negative log likelihood on all samples is: 0.12495198084074835
the average negative log likelihood on positive samples is: 2.7922946693139945
the average negative log likelihood on negative samples is: 0.03807588903755077
the auPR is: 0.14016833419421373
the auROC is: 0.7573931517071275
```

オフライン検証の具体例1

グリッドサーチによるハイパーパラメータの最適化を行うためのオフライン検証

```
import sagemaker
import numpy as np

sess = sagemaker.session.Session(boto_session=boto_session)

test_cases = [
    (a, b) for a in np.arange(0.0, 1.0, 0.1) for b in np.arange(0.0, 1.0, 0.1)
]

for ftrla, ftrlb in test_cases:
    this_job_name =  f'{job_name}-a{ftrla}-b{ftrlb}'.replace(".", "")
    sess.train(image
               , input_mode
               , input_config
               , role
               , this_job_name
               , output_config
               , resource_config
               , {**hyperparameters, 'ftrla': ftrla, 'ftrlb': ftrlb}
               , stop_condition)
```

ハイパーパラメータの一部だけ上書き

FreakOut

特徴量を変更したデータに対するオフライン検証

特徴量を削減したデータに対してJob を submit する

```
def get_input_config(x):
    return [{
        'ChannelName': 'train',
        **sage.session.s3_input(f'{s3_base}/{exp_name}/{x}/train').config
    }, {'ChannelName': 'test',
        **sage.session.s3_input(f'{s3_base}/{exp_name}/{x}/test').config
    }]

def get_output_config(x):
    return {'S3OutputPath': f"{s3_base}/{exp_name}/{x}/out/"}
```

with User-Agent

```
condition = 'w-ua'

input_conf = get_input_config(condition)
output_conf = get_output_config(condition)

for i in range(n_trials):
    job_name = f"{job_exp_name}-{condition}-t{i:02}"
    sess.train(image, input_mode, input_conf, role, job_name, output_conf,
               resource_config, hyperparameters(n_features), stop_condition)
```

without User-Agent

```
condition = 'wo-ua'

input_conf = get_input_config(condition)
output_conf = get_output_config(condition)

for i in range(n_trials):
    job_name = f"{job_exp_name}-{condition}-t{i:02}"
    sess.train(image, input_mode, input_conf, role, job_name, output_conf,
               resource_config, hyperparameters(n_features - 1), stop_condition)
```

オフライン検証の具体例3

マシンタイプと学習アルゴリズムで使用する並列スレッド数を
変更して学習時間とコストを比較するオフライン検証

・マシンタイプ
・スレッド数
だけ上書きしている

```
sess = sagemaker.session.Session(boto_session=boto_session)

test_cases = [
   ('ml.c5.2xlarge', '7'),
   ('ml.c5.4xlarge', '15'),
   ('ml.c5.9xlarge', '35'),
   ('ml.c5.18xlarge', '71')
]

for instance_type, thread_num in test_cases:
  this_job_name = "{}-{}-{}".format(job_name, instance_type, thread_num).replace(".", "")
  print(this_job_name)
  sess.train(image
            , input_mode
            , input_config
            , role
            , this_job_name
            , output_config
            , {**resource_config, 'InstanceType': instance_type}
            , {**hyperparameters, 'thnum': thread_num}
            , stop_condition)
```

```
INFO:sagemaker:Creating training-job with name: test-sagemaker-fout-fm-201804271647-mlc52xlarge-7
```

```
test-sagemaker-fout-fm-201804271647-mlc52xlarge-7
```

```
INFO:sagemaker:Creating training-job with name: test-sagemaker-fout-fm-201804271647-mlc54xlarge-15
INFO:sagemaker:Creating training-job with name: test-sagemaker-fout-fm-201804271647-mlc59xlarge-35
```

```
test-sagemaker-fout-fm-201804271647-mlc54xlarge-15
test-sagemaker-fout-fm-201804271647-mlc59xlarge-35
```

```
INFO:sagemaker:Creating training-job with name: test-sagemaker-fout-fm-201804271647-mlc518xlarge-71
```

```
test-sagemaker-fout-fm-201804271647-mlc518xlarge-71
```

FreakOut

プロダクション環境での利用

- 現在進行中. 学習モデルの取得元を S3 に切り替える想定
- バッチごとに使用するリソースを決めることができ、
  学習時間を柔軟に調整できるようになる

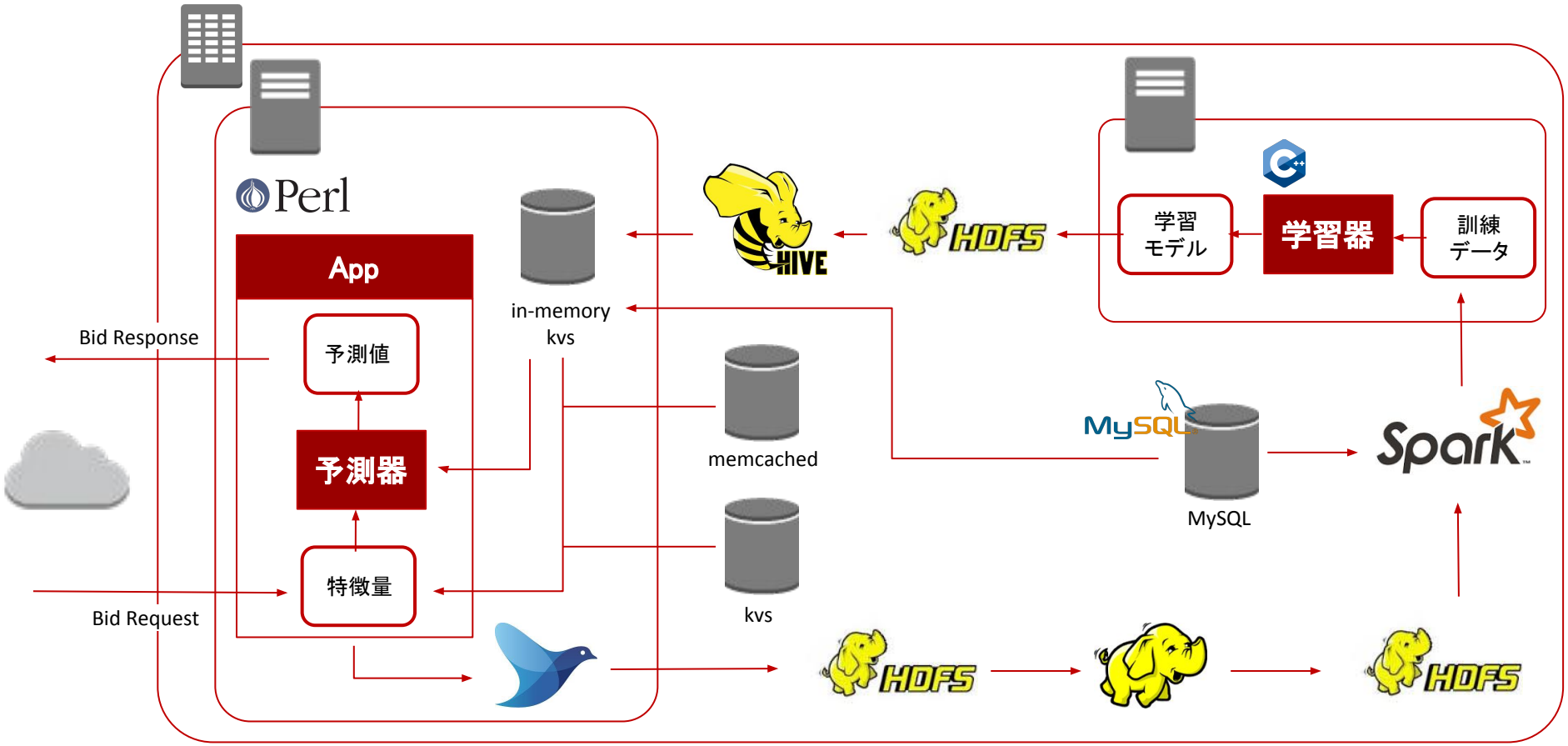
学習・評価結果の可視化

- 評価結果比較用のリーダーボードを用意
- 学習曲線の可視化を CloudWatchLogs で行う

実験スクリプトの永続化の簡単な方法

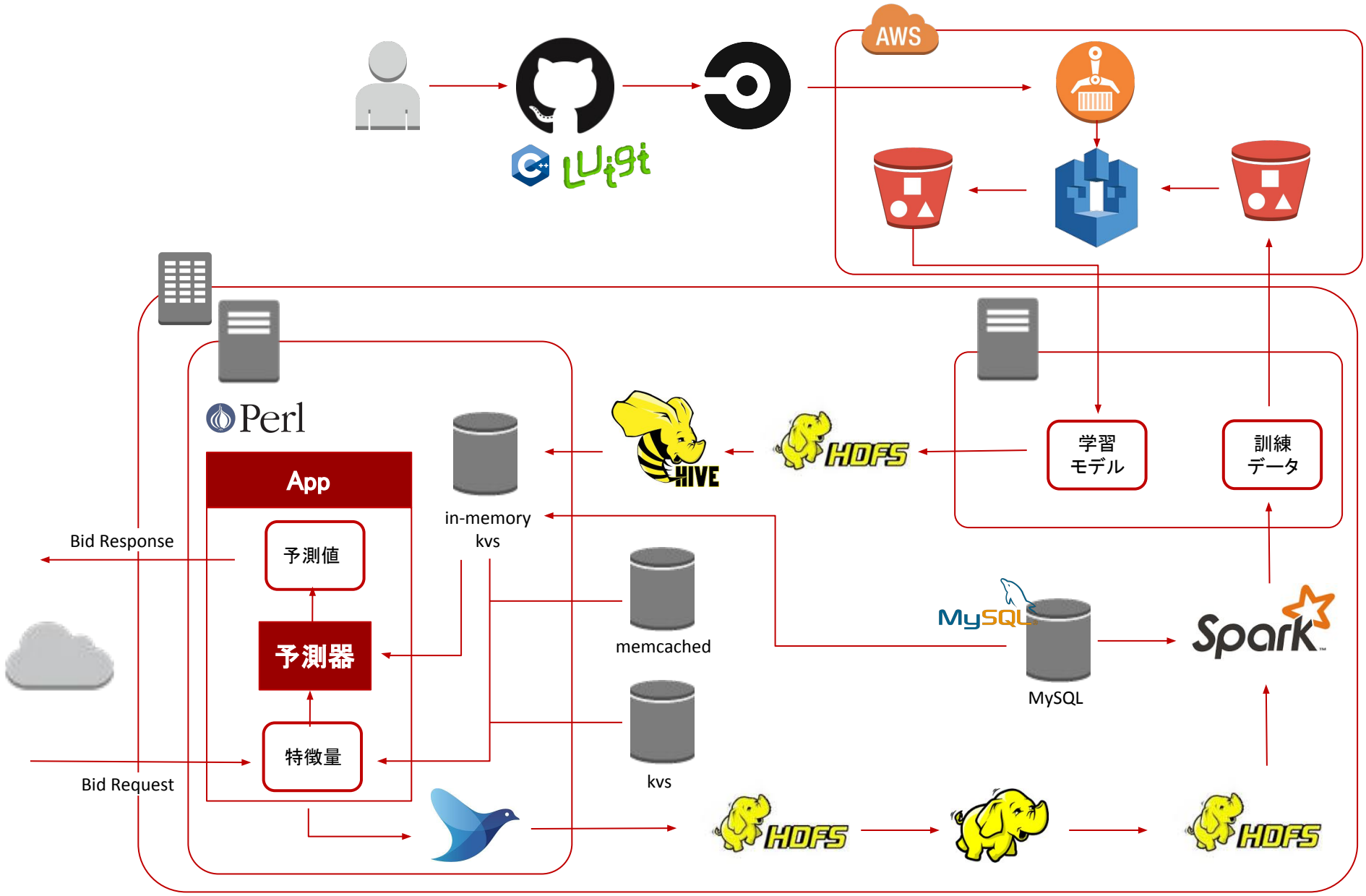
- 今は手動で .ipynb をダウンロードして GitHub に Upload

As-is Production Architecture | Future Work



To-be Production Architecture | Future Work

FreakOut



Summary

- 機械学習モデルの精度向上の仮説検証に SageMaker を利用した
- SageMaker を利用することで複数人で大規模に並列で検証が可能になった
- また Job 単位でリソースを柔軟に調整できることで、効率的に検証が行えるようになった

FreakOut

Give People Work That Requires A Person.