Serverless deep learning

Rustem Feyzkhanov
18 July 2018
Data science process

- **Business understanding**
  - Define objectives
  - Identify data sources

- **Data acquisition**
  - Ingest data
  - Explore data
  - Update data

- **Modeling**
  - Feature selection
  - Create model
  - Train model

- **Deployment**
  - Operationalize

- **Customer acceptance**
  - Testing and validation
  - Handoff
  - Re-train and re-score

Data science process

Challenges:
- starting fast
- being flexible
- integrating in current infrastructure

Takeaways from this talk

- How serverless deep learning works
- Serverless deep learning architecture
- Serverless deep learning use cases, do’s and don’ts
- Serverless deep learning simplicity of code
# Function as a service (FaaS)

<table>
<thead>
<tr>
<th>On premise</th>
<th>IaaS</th>
<th>PaaS</th>
<th>FaaS</th>
<th>SaaS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functions</td>
<td>Functions</td>
<td>Functions</td>
<td>Functions</td>
<td>Functions</td>
</tr>
<tr>
<td>Application</td>
<td>Application</td>
<td>Application</td>
<td>Application</td>
<td>Application</td>
</tr>
<tr>
<td>Runtime</td>
<td>Runtime</td>
<td>Runtime</td>
<td>Runtime</td>
<td>Runtime</td>
</tr>
<tr>
<td>Operating system</td>
<td>Operating system</td>
<td>Operating system</td>
<td>Operating system</td>
<td>Operating system</td>
</tr>
<tr>
<td>Virtualization</td>
<td>Virtualization</td>
<td>Virtualization</td>
<td>Virtualization</td>
<td>Virtualization</td>
</tr>
<tr>
<td>Networking</td>
<td>Networking</td>
<td>Networking</td>
<td>Networking</td>
<td>Networking</td>
</tr>
<tr>
<td>Storage</td>
<td>Storage</td>
<td>Storage</td>
<td>Storage</td>
<td>Storage</td>
</tr>
<tr>
<td>Hardware</td>
<td>Hardware</td>
<td>Hardware</td>
<td>Hardware</td>
<td>Hardware</td>
</tr>
</tbody>
</table>
Lambda function - AWS implementation of FaaS
How Lambda works

- Container pool
- Warm container
- Lambda configuration
  - Code
  - Libraries
  - Configuration (memory, max time)

Trigger
- S3
- API
- DynamoDB
- SQS, Kinesis

Response
Lambda triggers

- DynamoDB
- S3
- CloudWatch
- Lex
- SQS
- API gateway
# Lambda pros/cons/limits

<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
<th>Limits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easy to deploy (no docker)</td>
<td>Logging is not great</td>
<td>max 3 GB RAM</td>
</tr>
<tr>
<td>Easy to connect to triggers (API, S3, SQS, DynamoDB)</td>
<td>No local debug</td>
<td>max 500 MB disk</td>
</tr>
<tr>
<td>Easy to scale</td>
<td>Unpredictable warm containers</td>
<td>max 5 min execution time</td>
</tr>
<tr>
<td>Relatively cheap</td>
<td></td>
<td>CPU is proportional to provisioned memory</td>
</tr>
</tbody>
</table>
Google web search interest for different deep learning frameworks over time

Francois Chollet. “Deep Learning with Python MEAP.”
TensorFlow 1.*

• Keras in the core
• TF Boosted trees (!) + other ML algorithms
• Lots of other stuff:

  [https://github.com/tensorflow/tensorflow/blob/master/RELEASE.md](https://github.com/tensorflow/tensorflow/blob/master/RELEASE.md)
Why TF on Lambda?

- ~20000 runs for $1
- 1000 concurrent executions (up to 10000)
- Pay as you go model

=> perfect for early stage projects
Implementation Problem

Lambda limit - 50 MB

TensorFlow archive size - 43.1 MB

Numpy archive size - 16.5 MB

+ dependencies
Implementation Problem

Lambda limit - 50 MB

TensorFlow archive size - 43.1 MB

Numpy archive size - 16.5 MB

+ dependencies

250 MB unarchived

https://hackernoon.com/exploring-the-aws-lambda-deployment-limits-9a8384b0bec3
How to solve

1. Compress so files

Magic:

2. Delete .pyc files

3. Remove test folders, visualisation folders

Look up here: https://github.com/ryfeus/lambda-packs/blob/master/Tensorflow/buildPack.sh
How to import models

Use Keras or TensorFlow for weights import:

• Keras - h5 files
• TensorFlow - pb files
Usual AWS architecture for DL

Route 53 → ALB → ECS → ECR → EC2

ACG
Spot
Architecture for DL using Lambdas

Route 53 → API Gateway → Lambda → S3
Architecture for DL using Lambdas

SQS -> Lambda

S3

#oscon
Architecture for DL using Lambdas

SQS → λ → Lambda → S3

Step function
Architecture for DL using Lambdas

Step functions:
- allow modular approach
- enable to handle errors and special cases
- serverless functions => serverless pipelines
Where to get models

Train yourself

Keras:

https://github.com/fchollet/deep-learning-models

TensorFlow:

TensorFlow ZOO (https://github.com/tensorflow/models/tree/master/official)


Github projects (e.g. https://github.com/taehoonlee/tensornets)
Projects - Image recognition

API to recognize image using Inception-v3 - 0.00005$ / 1 image

https://github.com/ryfeus/lambda-packs/tree/master/Tensorflow
https://www.tensorflow.org/tutorials/image_recognition
Projects - accessible WEB

API to describe what happens on the picture - 0.0001$ / 1 image

Image

API Response

```
[
  {
    url: "https://hack4impact.org/assets/images/photos/mayors-awards.jpg",
    captions: [
      {
        prob: "0.005999",
        sentence: "a group of people standing next to each other ."
      },
      {
        prob: "0.002621",
        sentence: "a group of people posing for a picture ."
      },
      {
        prob: "0.001902",
        sentence: "a group of people posing for a picture"
      }
    ]
  }
]
```

How do you know if this is for you

- You want to deploy your model for pet project
- You want to make a simple MVP for your startup/project
- You have simple model and this architecture will reduce cost
- You have peak loads and it is hard to manage clusters
How do you know if this is NOT for you

- You have very complex model (a lot of data as input/high CPU)
- You need to have real-time response
Some new stuff

- LightGBM package - fast, distributed, high performance gradient boosting framework from Microsoft + Sklearn/Scipy/Numpy

https://github.com/ryfeus/lambda-packs/tree/master/LightGBM_sklearn_scipy_numpy

- Spacy package - natural language processing library

https://github.com/ryfeus/lambda-packs/tree/master/Spacy
Conclusions
