Challenges on serving LLMs

Theofilos Papapanagiotou, Amazon

Agenda

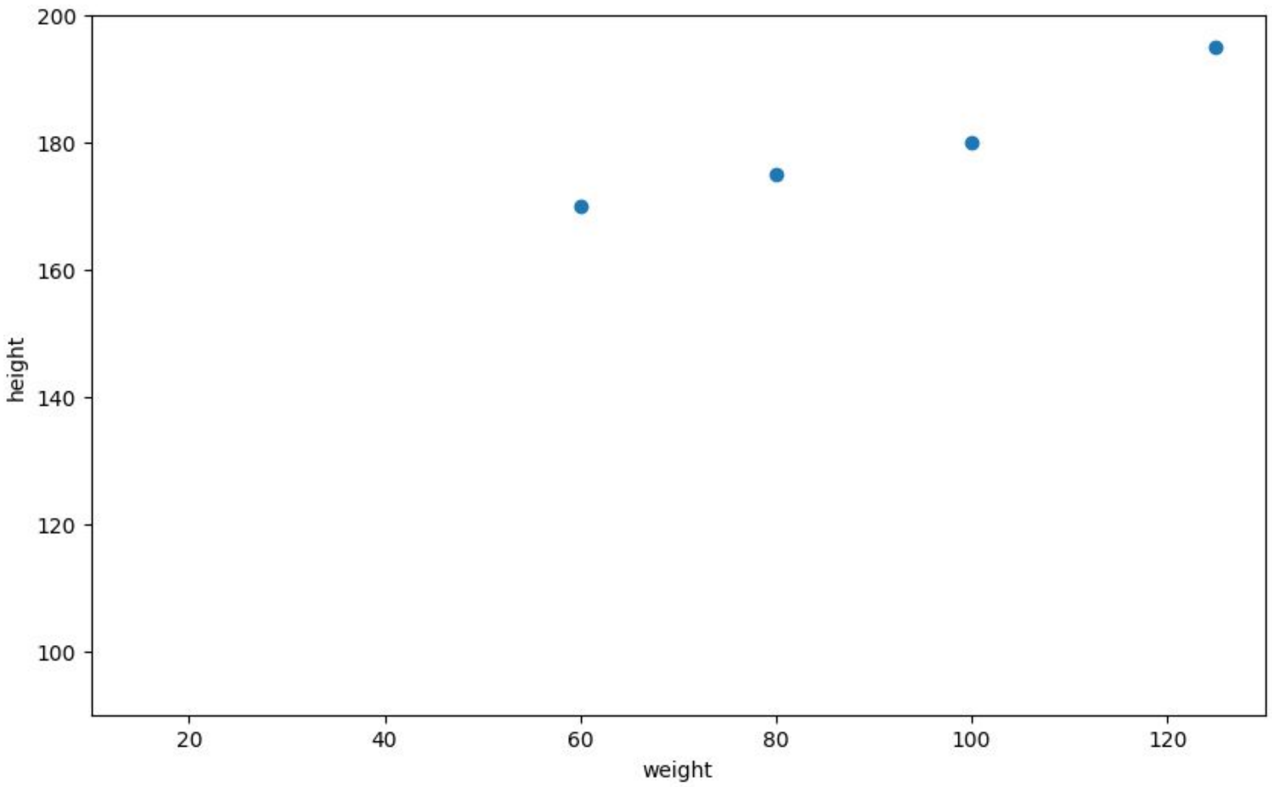
- Basics: Models and Parameters
- Saving formats and Model servers
- Deployment
- Model monitoring
- Scaling techniques

About me

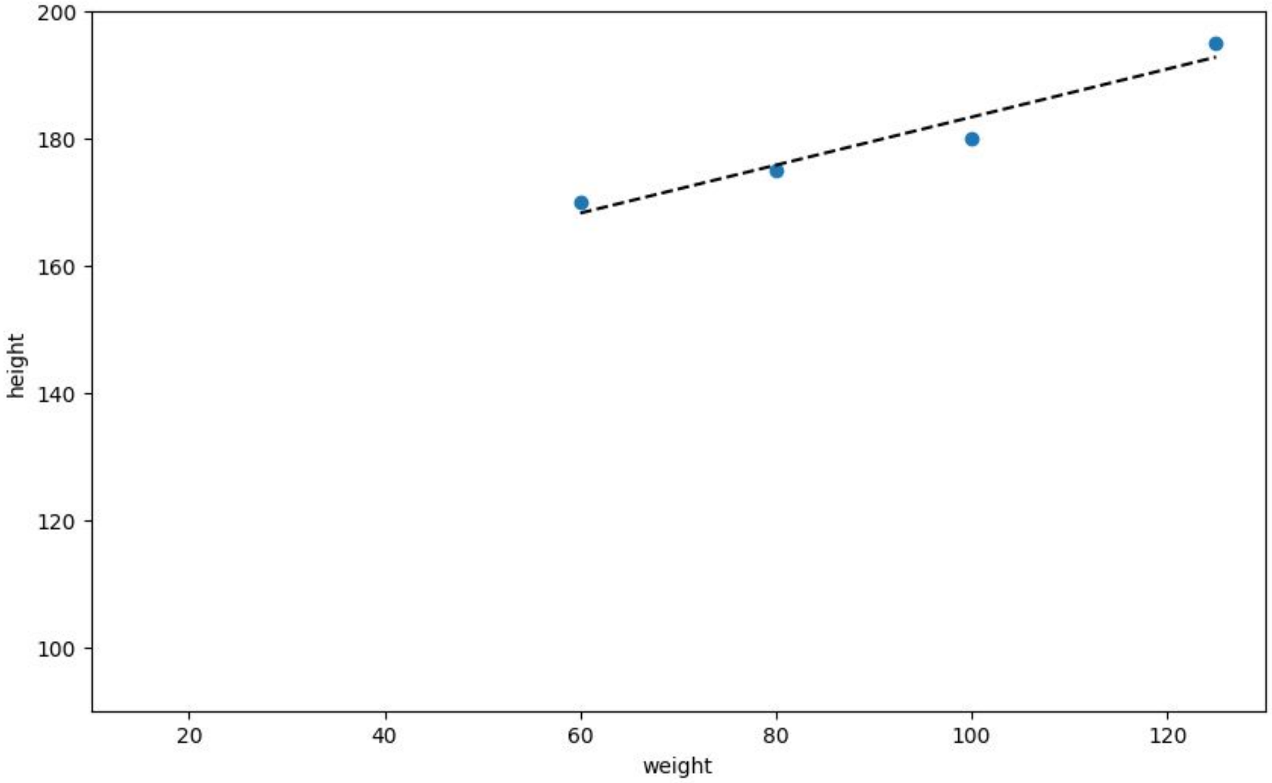
- 👨‍💻 ML & Kubernetes communities
- 🏢 Engineer in EU Operations, Amazon
- ❤️ Python/Go
- 🏠 Amsterdam

Basics: Models and parameters

```
x = [ 60.0, 80.0, 100.0, 125.0 ]
y = [ 170.0, 175.0, 180.0, 195.0 ]
```

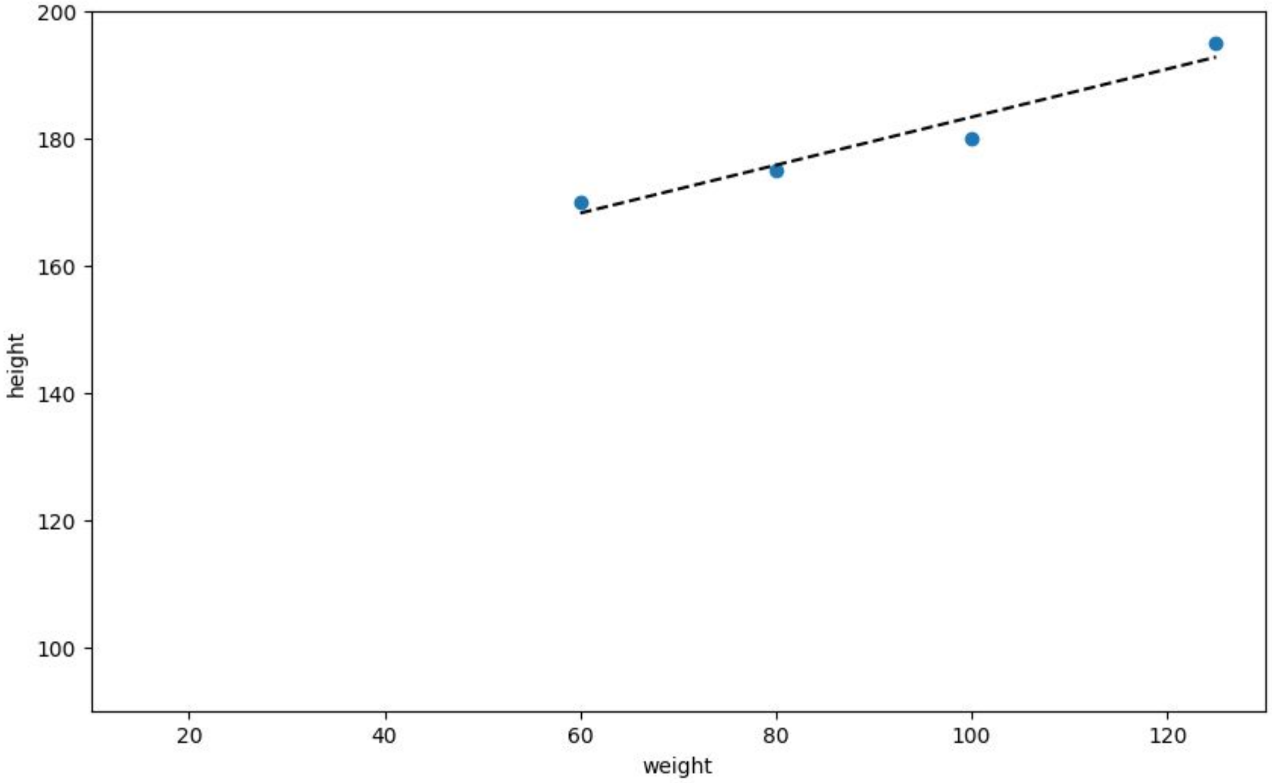


```python
from sklearn import linear_model
regr = linear_model.LinearRegression()
regr.fit(x, y)
```



```python
print(regr.coef_, regr.intercept_)
```

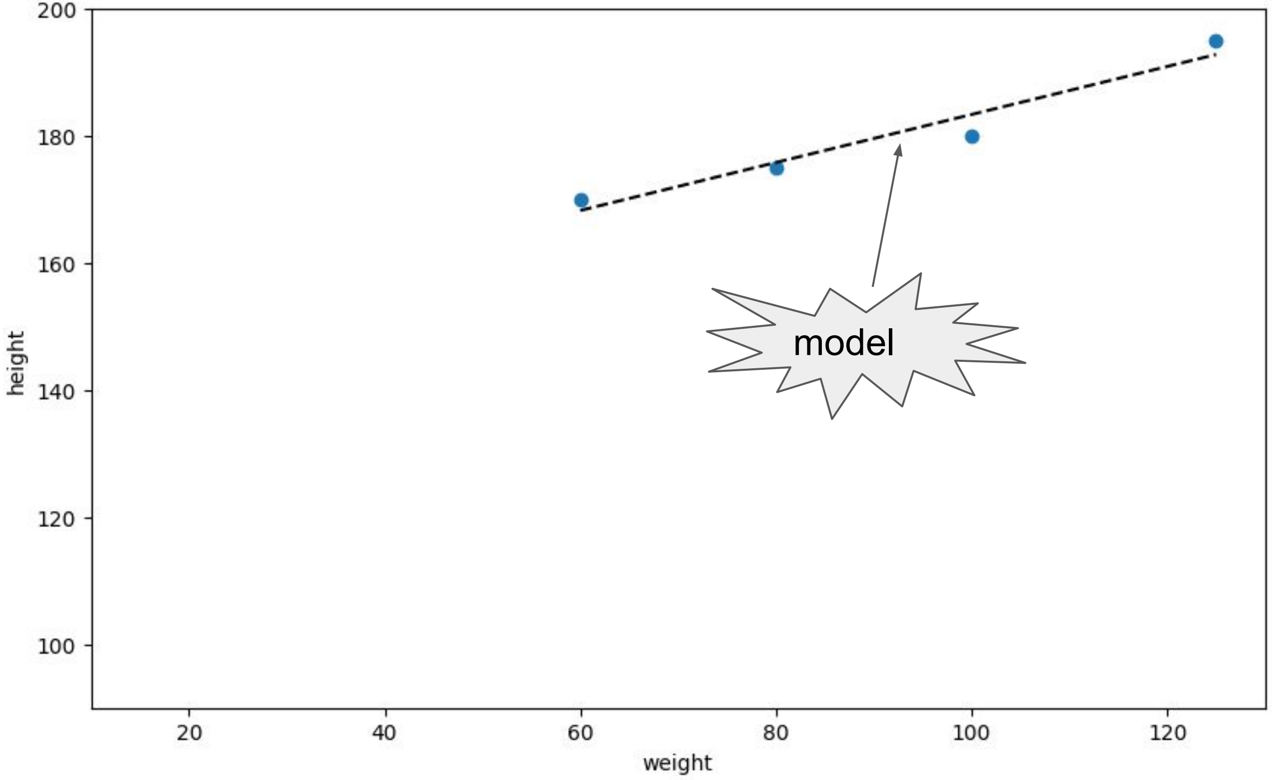
```
[0.37735849] 145.56603773584905
```



$y = \mathbf{0.377}\ x + \mathbf{145}$

```python
print(regr.coef_, regr.intercept_)
```

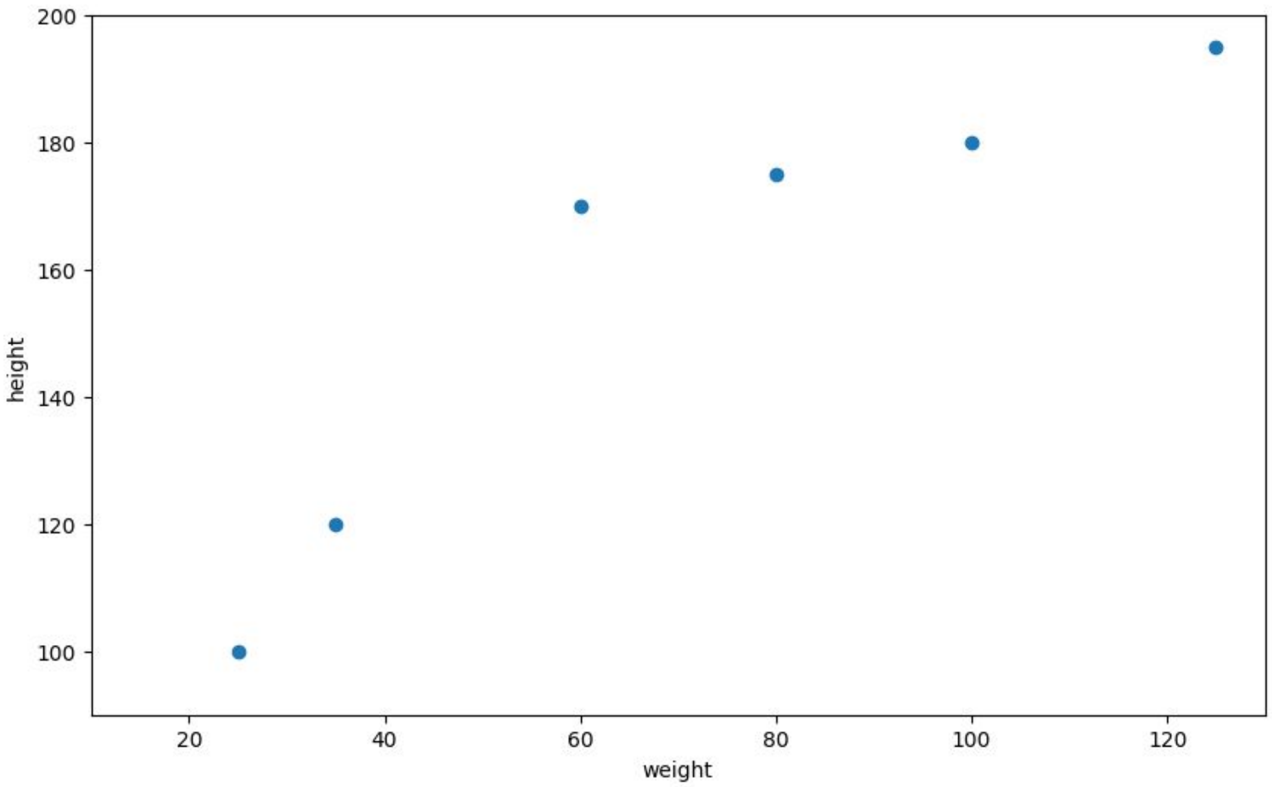
```
[0.37735849] 145.56603773584905
```

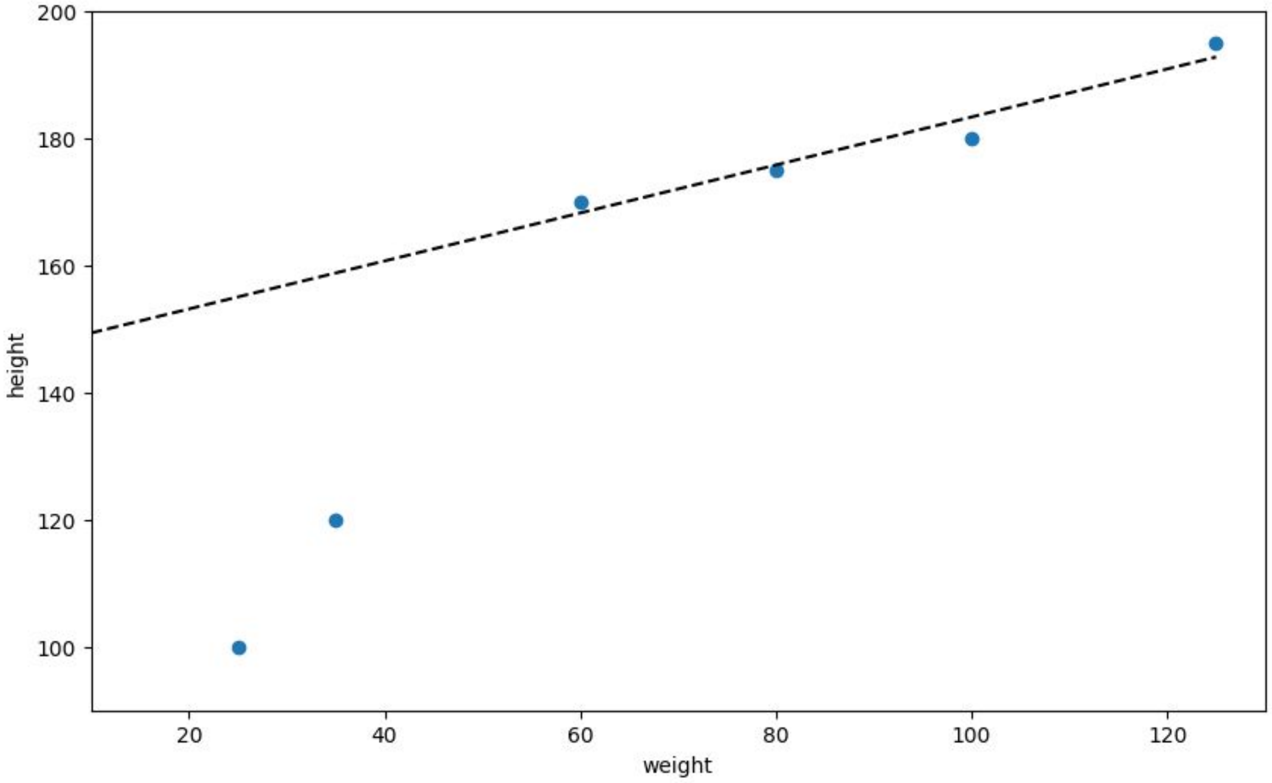


y = **0.377** x + **145**

parameters

```
x = [ 25, 35, 60.0, 80.0, 100.0, 125.0 ]
y = [ 100, 120, 170.0, 175.0, 180.0, 195.0 ]
```



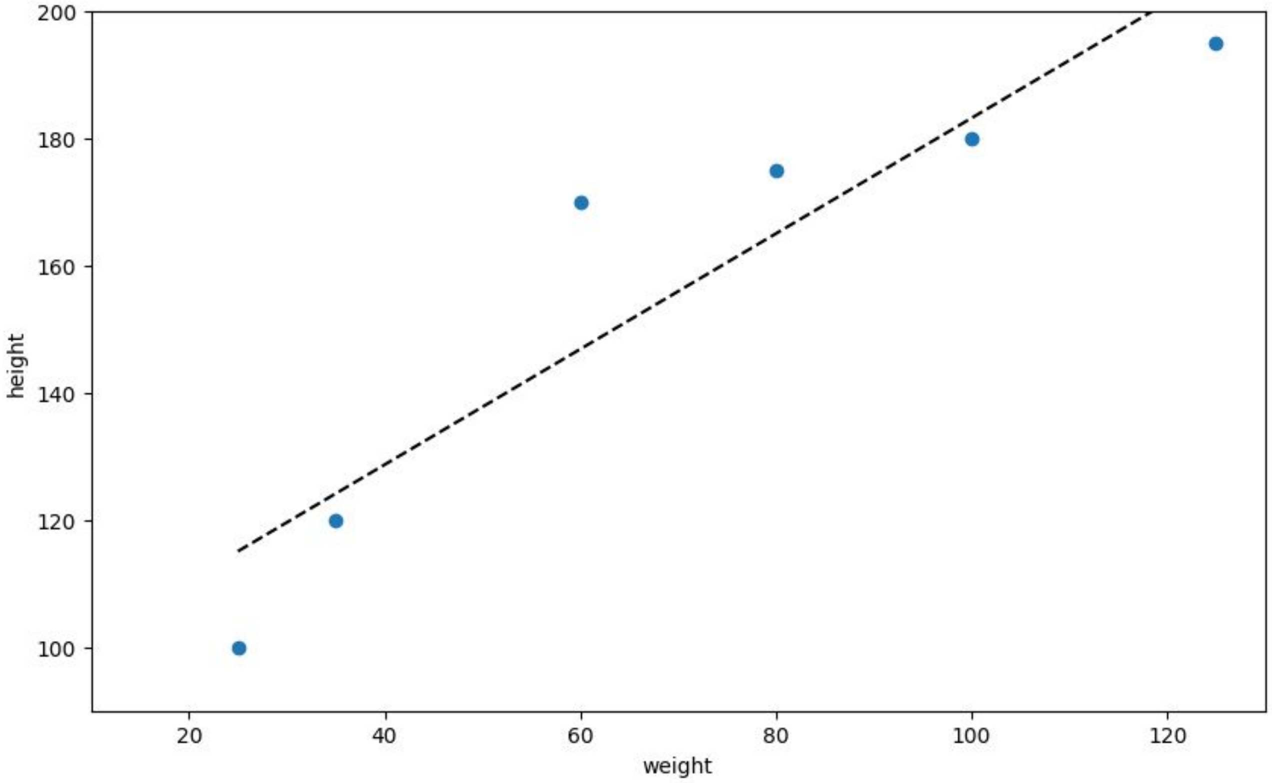


y = **0.377** x + **145**

retrain

```
from sklearn import linear_model
regr = linear_model.LinearRegression()
regr.fit(x, y)
```

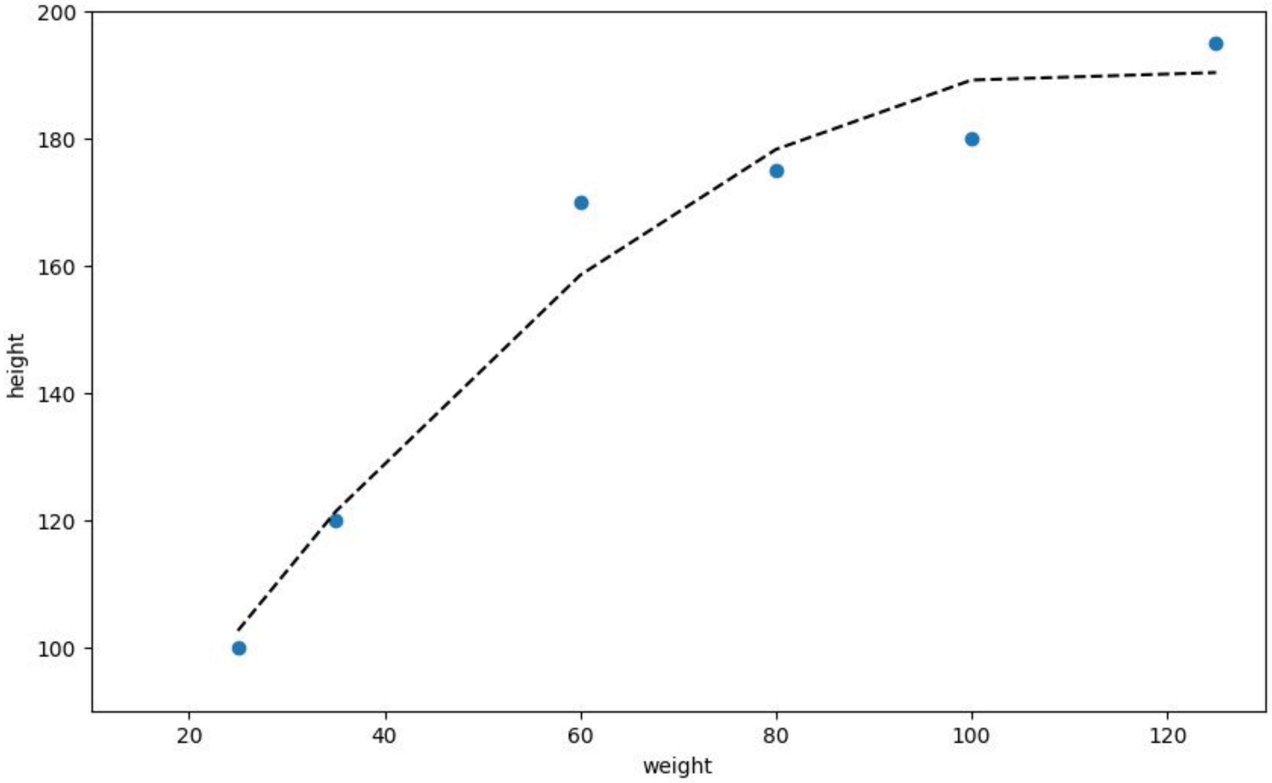
```
print(regr.coef_, regr.intercept_)

[0.90785755] 92.36009044657999
```



y = **0.9** x + **92**

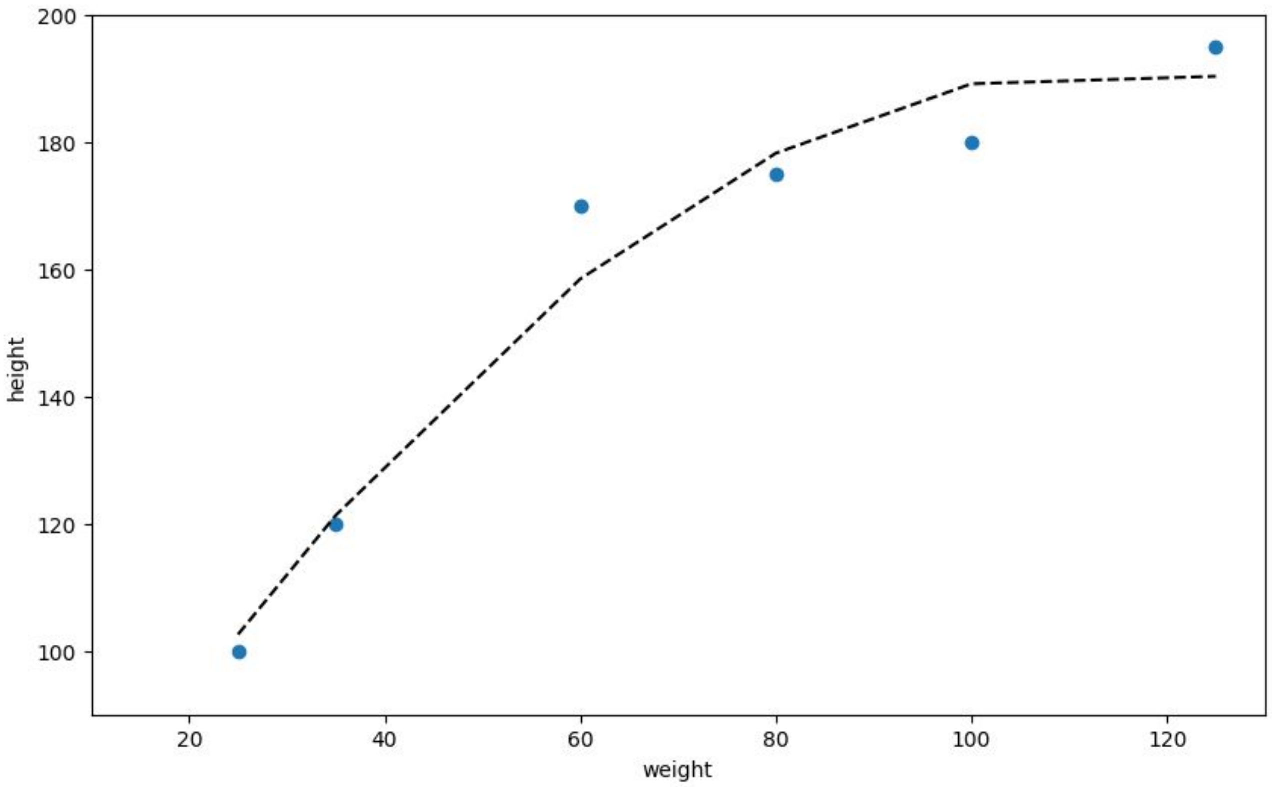
new parameters

polynomial

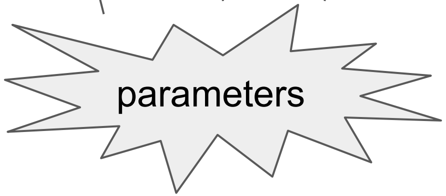


$$y = \mathbf{-0.01}\ x^2 + \mathbf{2.5}\ x + \mathbf{46}$$

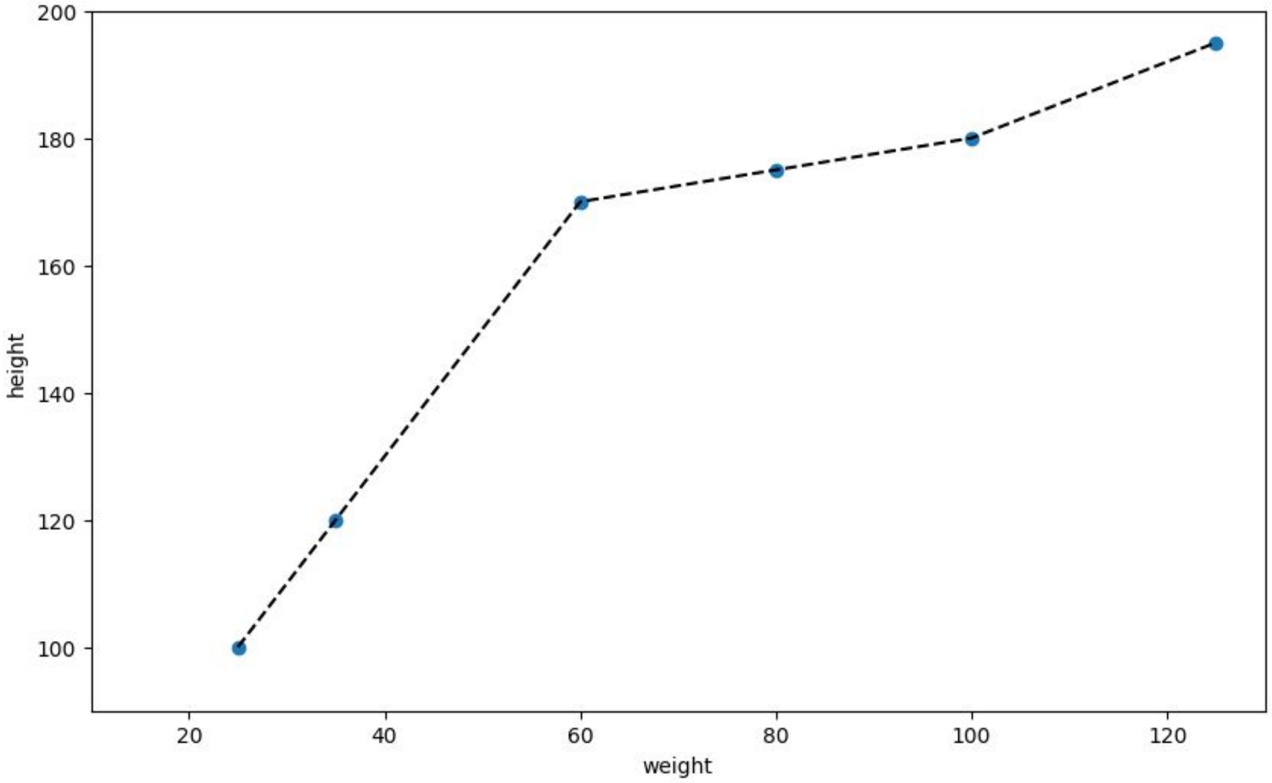
polynomial



$y = \mathbf{-0.01}\ x^2 + \mathbf{2.5}\ x + \mathbf{46}$



neural network



$y = w \cdot x$

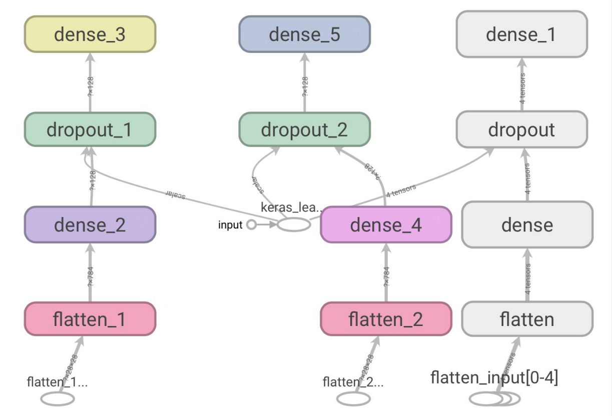
```
[array([[-3.57261984e-02,  1.89047917e-02,
        -8.90411427e-03,  2.21588407e-01,
        1.88532590e-02, -1.15048285e-01,
        2.47424537e-02, -9.41278783e-03,
        1.64560095e-01,  4.61095106e-02,
        4.49640934e-02, -3.63451478e-02,
        -4.00167897e-02, -8.76319520e-02,
        6.91530696e-02,  1.41112641e-01,
        -8.37367776e-03,  1.41093726e-01,
        -4.67398997e-02,  9.64951964e-02,
        -1.81029315e-01,  4.64469347e-02,
        -1.62661230e-01,  5.54083077e-02,
        -5.61493965e-03,  8.32387396e-02,
        -9.44921732e-03, -1.58248688e-01,
        -5.66456648e-02,  1.16658648e-03,
        -9.81299505e-02, -1.08349595e-02,
        -1.46522963e-01, -1.87401818e-01,
        -1.02077292e-01,  3.05457870e-02,
        1.53093451e-01,  8.96211493e-02,
        -4.18250981e-02, -1.63806557e-01,
        2.06004199e-01,  1.22082396e-01,
        -1.67390477e-01, -2.30[illegible]3685e-03,
        2.133[illegible]401e-0[illegible]  1.[illegible]703e-01,
        -7.8924[illegible]
```

parameters

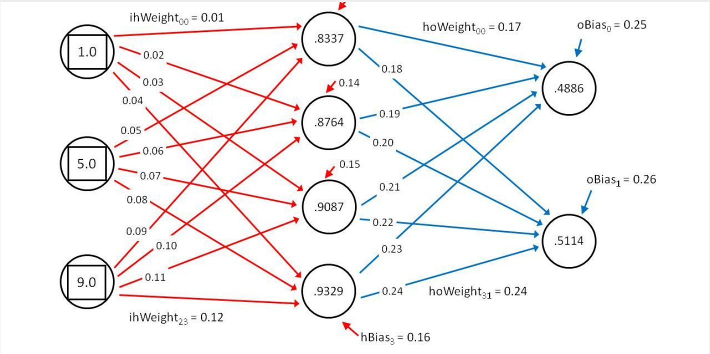
Saving formats and Model servers

Model

Model architecture (Graph)



Variables (weights & biases)



How do we save it in a file?

Pickle/Joblib

- Contain the class and the parameters of the model
- Custom model requires availability of the **class** with the algorithm
- Pickle vs Joblib:
  - Same methods (save/load)
  - Same structure
  - Joblib faster with numpy arrays

```
>>> import pickle
>>> pickle.load(open('mymodel.pkl','rb')).coefs_

[array([[-0.14196276, -0.02104562, -0.85522848, -3.51355396, -0.60434709],
        [-0.69744683, -0.9347486 , -0.26422217, -3.35199017,  0.06640954]]),
 array([[ 0.29164405, -0.14147894],
        [ 2.39665167, -0.6152434 ],
        [-0.51650256,  0.51452834],
        [ 4.0186541 , -0.31920293],
        [ 0.32903482,  0.64394475]]),
 array([[-4.53025854],
        [-0.86285329]])]
```

```
% strings mymodel.pkl|head -5
csklearn.neural_network._multilayer_perceptron
MLPClassifier
activationq
reluq
Solverq

% strings xgb.pkl|head -5
cxgboost.sklearn
XGBClassifier
n_estimatorsq
objectiveq
binary:logisticq
```

Tensorflow SavedModel

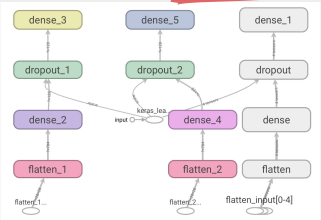
TensorFlow

MetaGraphDef

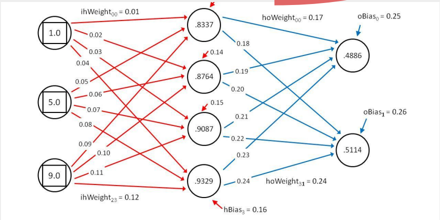
Metadata

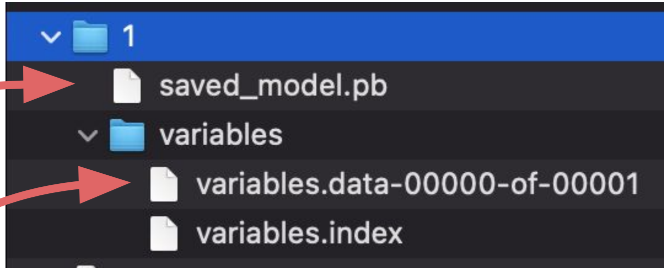
Assets
Tags
Version
SignatureDef

Model architecture (GraphDef)



Variables (weights & biases)

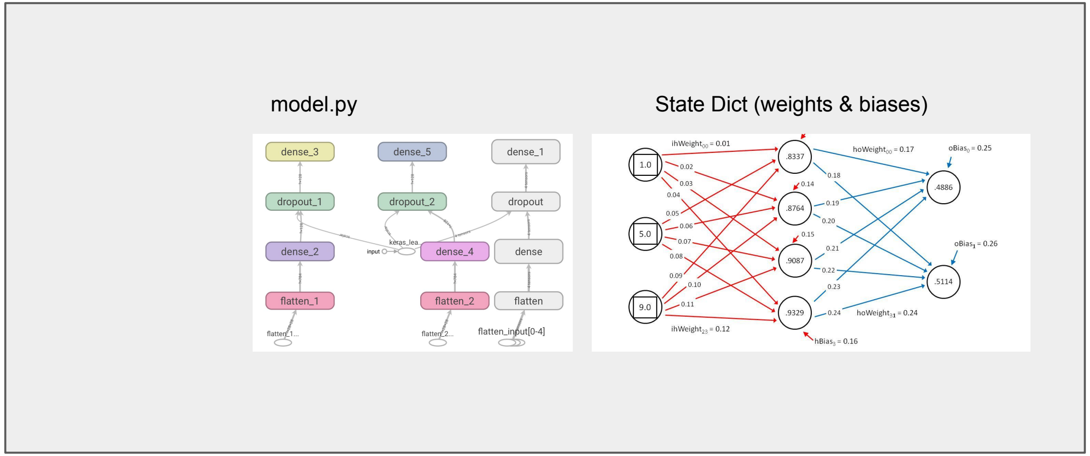


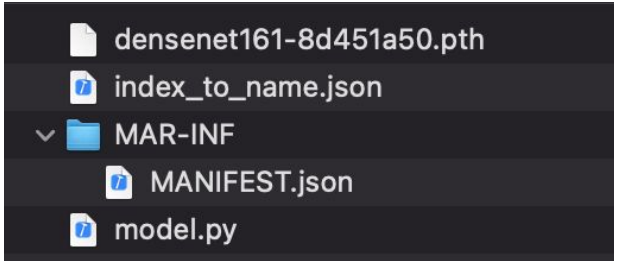



```
>>> import tensorflow as tf
>>> model=tf.keras.models.load_model('.')

>>> [i.name for i in model.weights]
['dense_4/kernel:0',
'dense_4/bias:0',
'dense_5/kernel:0',
'dense_5/bias:0'...

>>> model.weights[0][0].numpy()[:2]
array([0.06990073, 0.03880108], dtype=float32)
```

PyTorch Model Archive

PyTorch

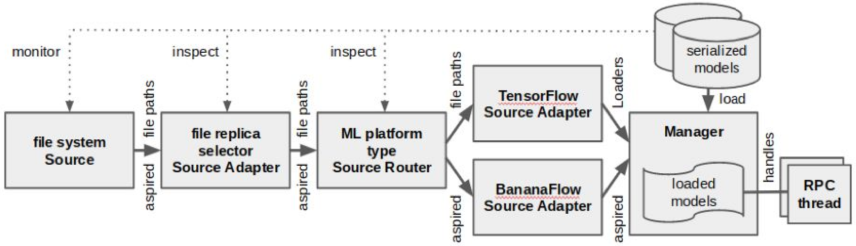





```
>>> import torch
>>> torch.load('densenet161-8d451a50.pth').keys()
```

```
odict_keys(
['features.conv0.weight',
'features.norm0.weight',
'features.denseblock1.denselayer1.norm.1.weight',
'features.denseblock1.denselayer1.norm.1.bias', ...
```

TensorFlow Serving

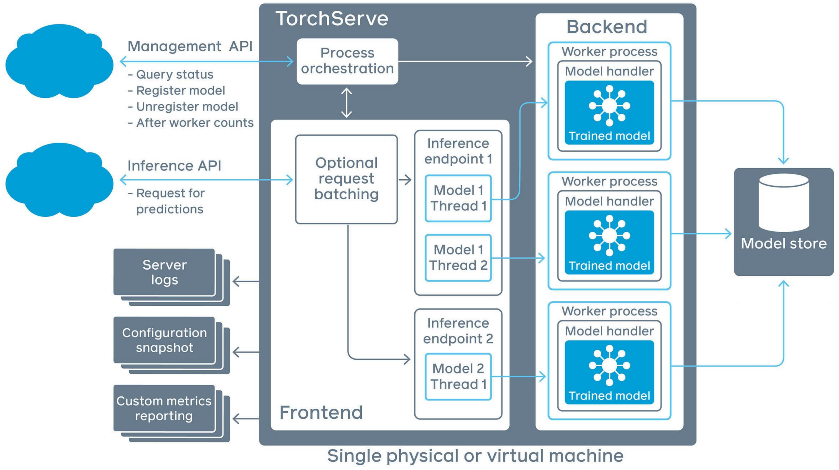



```
docker run -p 8501:8501 \
   -v /home/theofpa/models/mymnist:/mymnist -t \
   tensorflow/serving:2.3.0 --model_base_path=/mymnist --model_name=mymnist
```

```
curl -X POST http://localhost:8501/v1/models/mymnist:predict -d@digit.json
{
    "predictions": [
              [2.16072715e-10,
              1.42498227e-08,
              8.15775447e-09,
              0.00080721511,
              2.23614914e-19,
              0.999192774,
              6.97247078e-12,
              9.37563e-09,
              3.14906656e-10,
              5.07091258e-08]
    ]
}
```

https://arxiv.org/abs/1712.06139 (2017)
https://github.com/tensorflow/serving

TorchServe



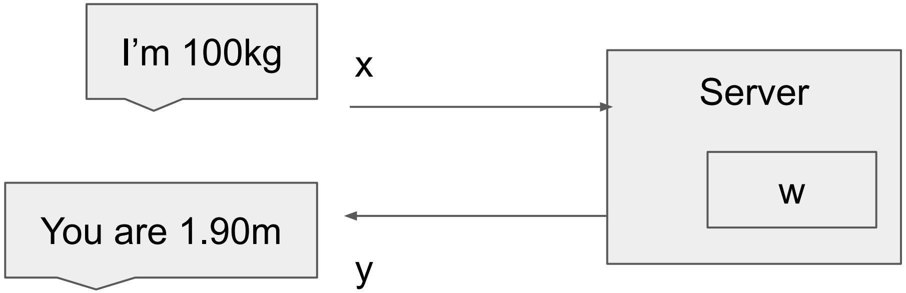

```
docker run -p 8080:8080 \
  -v /home/theofpa/models/pytorch-densenet:/models pytorch/torchserve torchserve \
  --model-store /models --models densenet161.mar
```

```
curl http://127.0.0.1:8080/predictions/densenet161 -T kitten_small.jpg
{
  "tabby": 0.5078840851783752,
  "lynx": 0.18985284864902496,
  "tiger_cat": 0.16152925789356232,
  "tiger": 0.05462226644158363,
  "Egyptian_cat": 0.04894305393099785
}
```

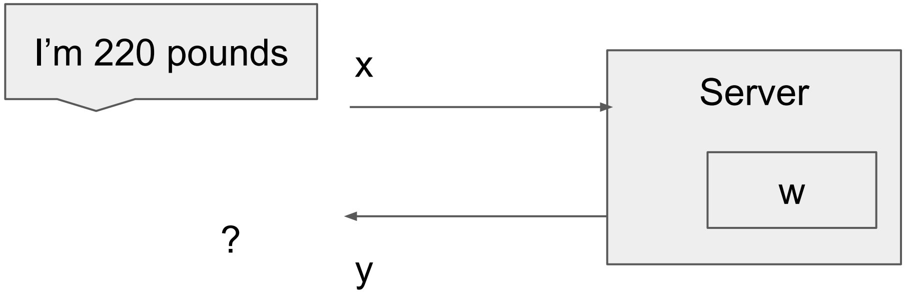
https://github.com/pytorch/serve (2020)

Deployment for scale

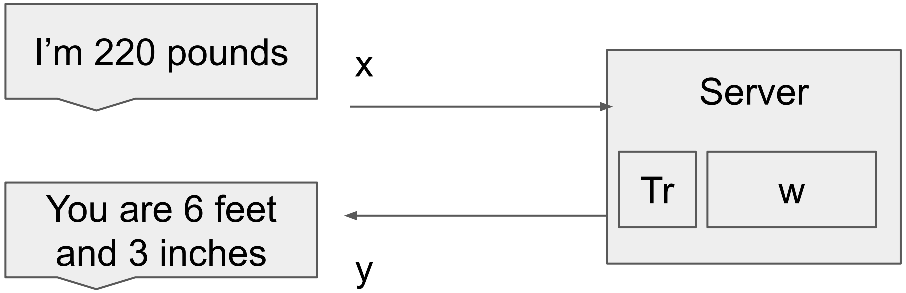
Inference



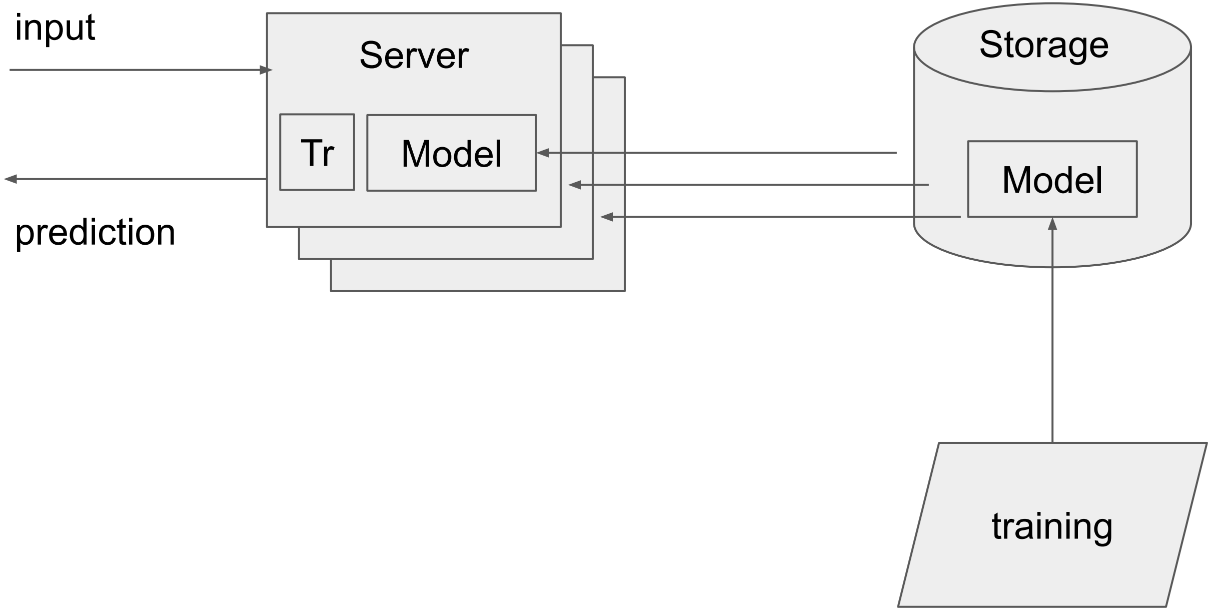
Inference



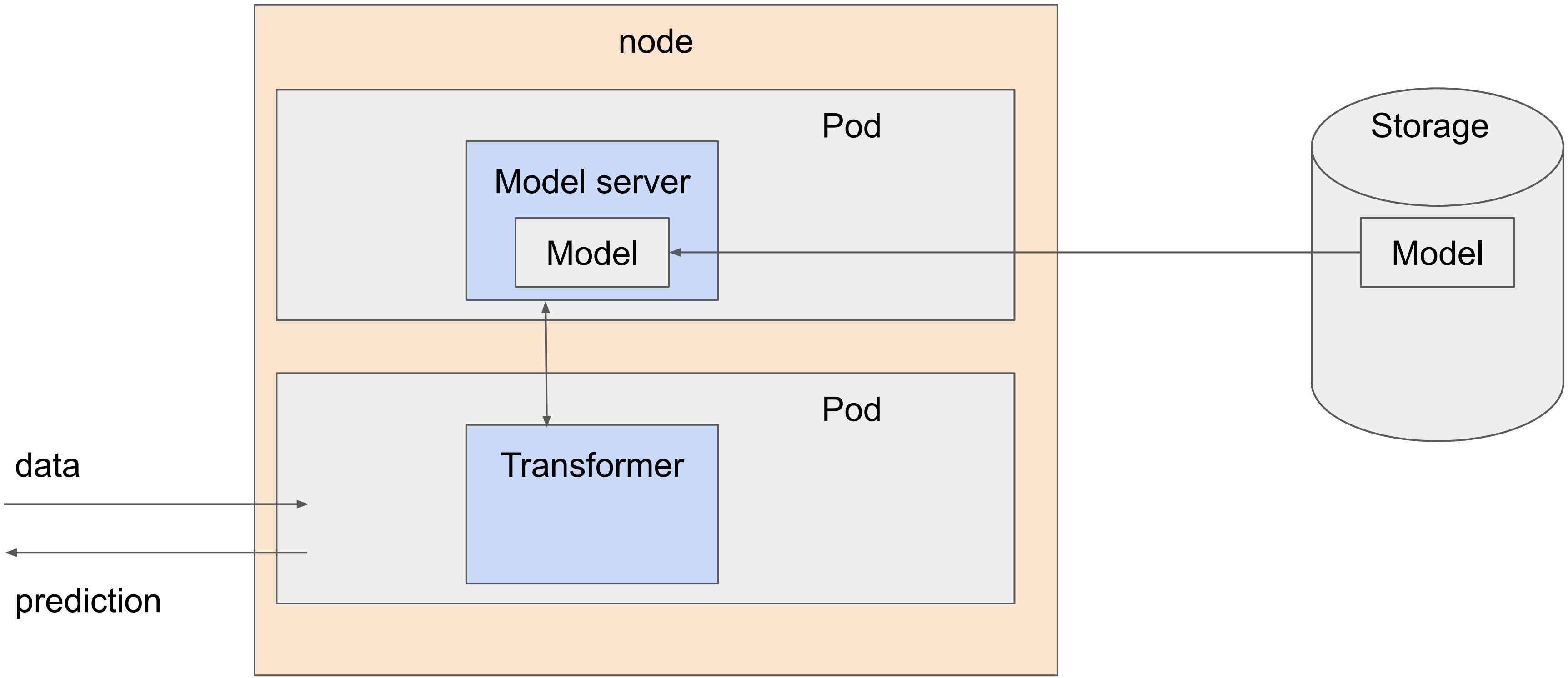
Inference



Deployment

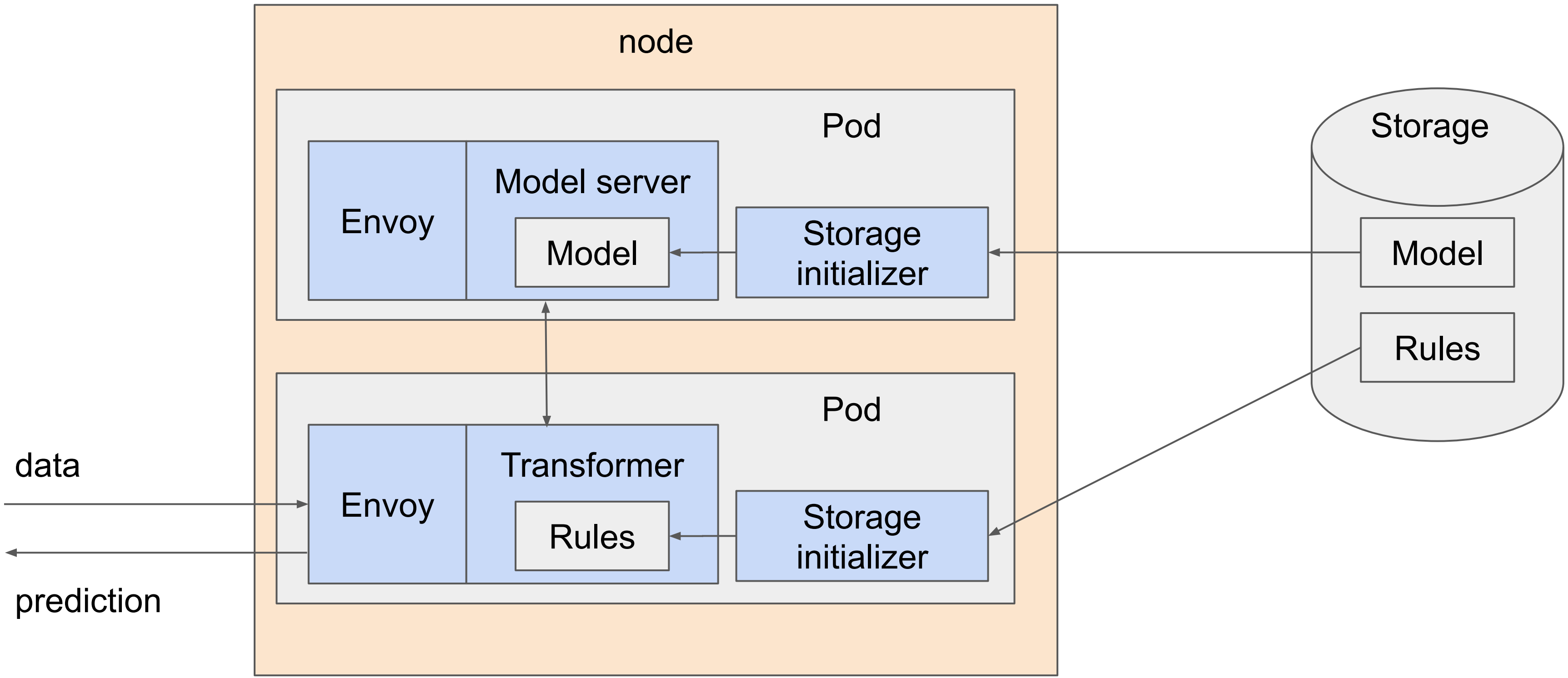


Deployment

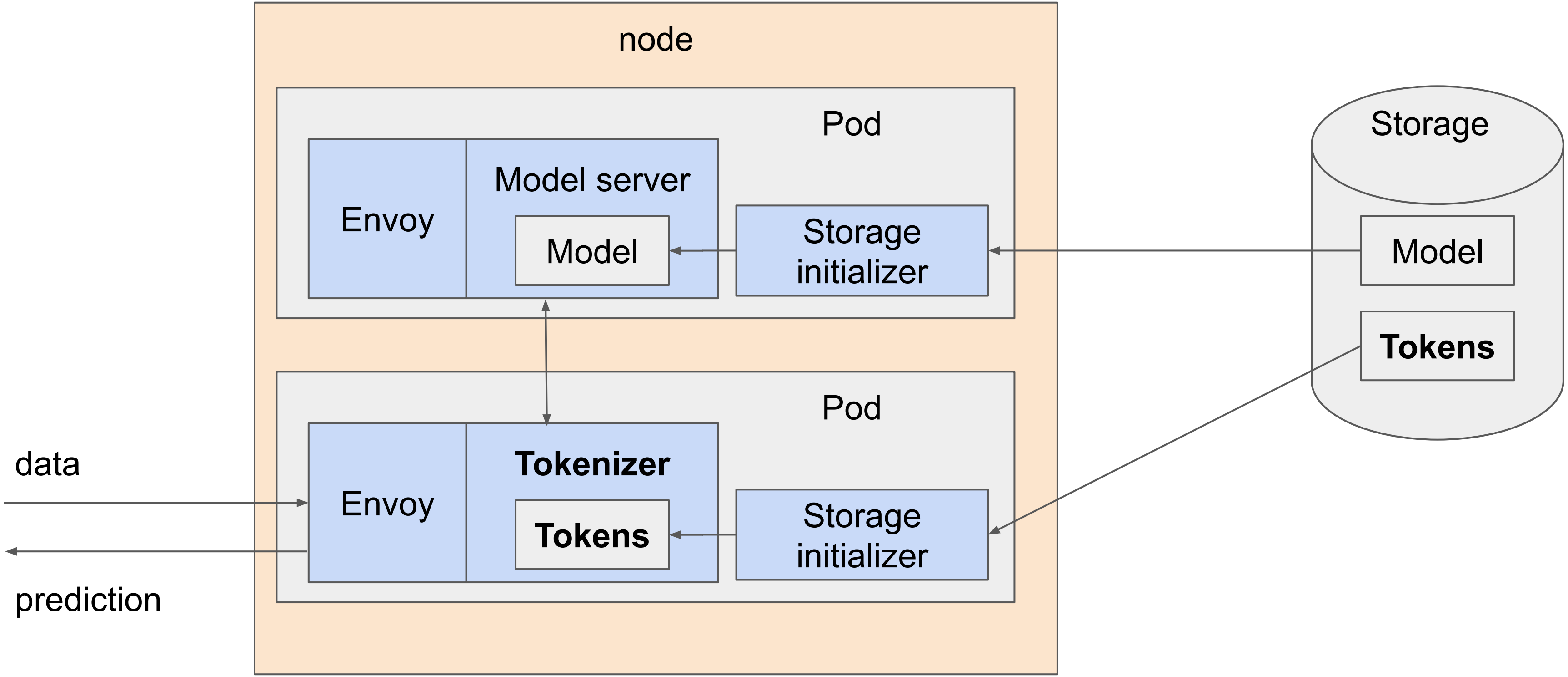


node
Pod
Model server
Model
Pod
Transformer
Storage
Model
data
prediction

Deployment

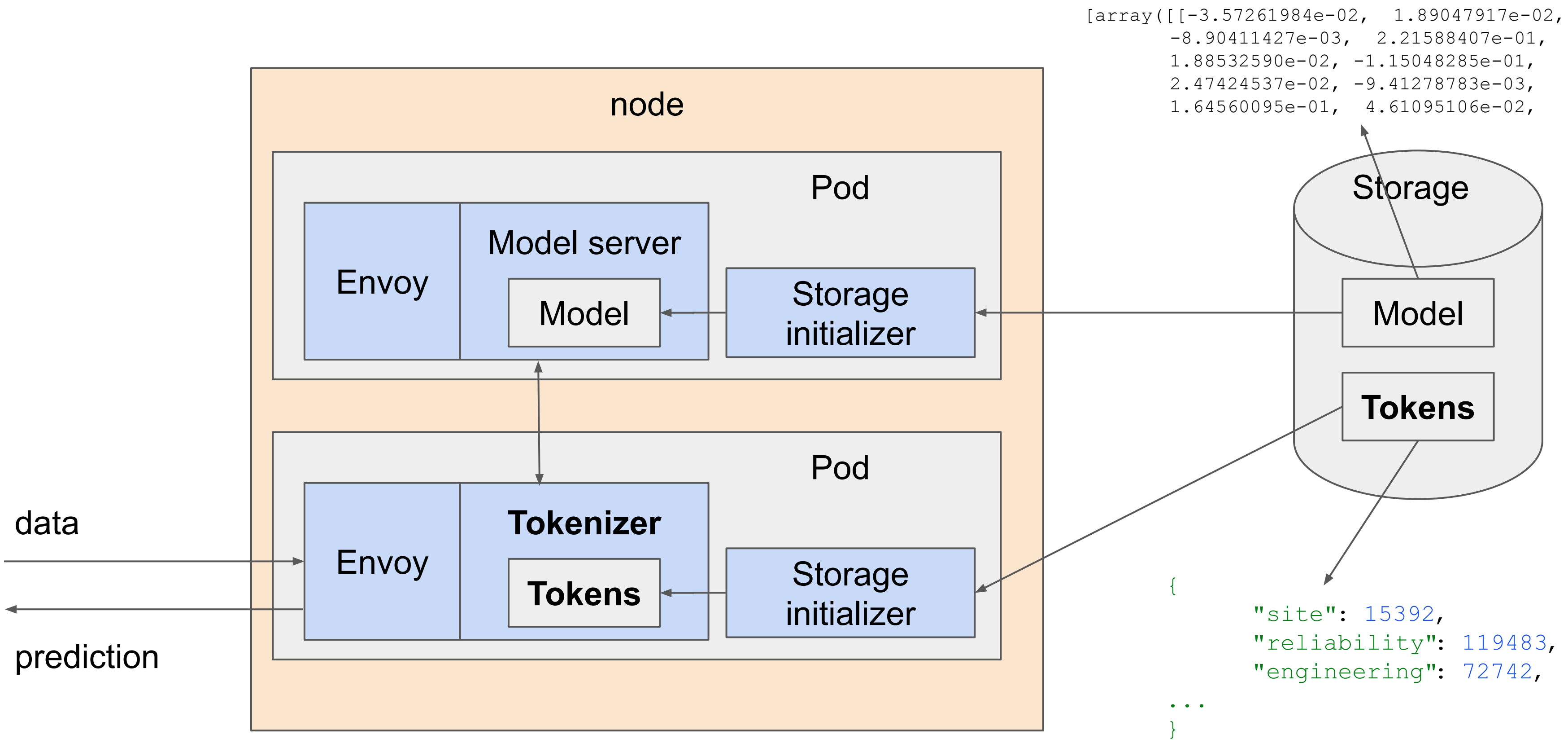

node
Pod
Envoy
Model server
Model
Storage initializer
Pod
Envoy
Transformer
Rules
Storage initializer
Storage
Model
Rules
data
prediction

Deployment



node
Pod
Envoy
Model server
Model
Storage initializer
Pod
Envoy
Tokenizer
Tokens
Storage initializer
Storage
Model
Tokens
data
prediction

Deployment


[array([[-3.57261984e-02, 1.89047917e-02,
-8.90411427e-03, 2.21588407e-01,
1.88532590e-02, -1.15048285e-01,
2.47424537e-02, -9.41278783e-03,
1.64560095e-01, 4.61095106e-02,
node
Pod
Envoy
Model server
Model
Storage initializer
Storage
Model
Tokens
Pod
Envoy
Tokenizer
Tokens
Storage initializer
data
prediction
{
"site": 15392,
"reliability": 119483,
"engineering": 72742,
...
}

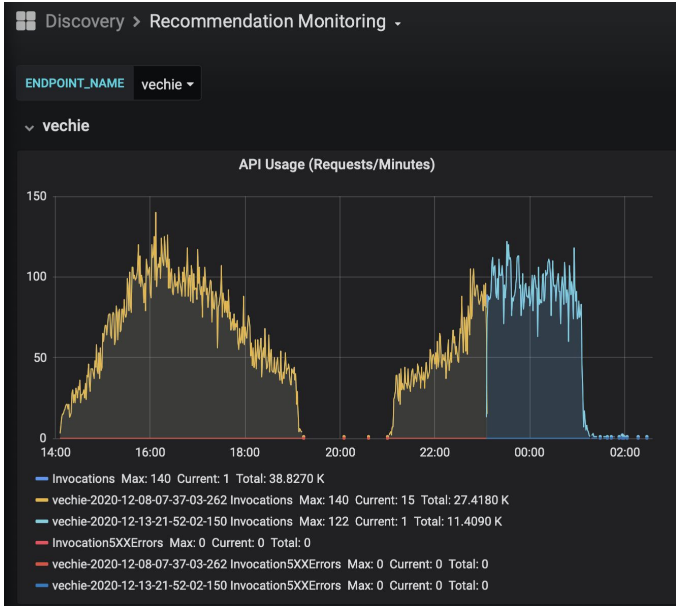
Model monitoring

Topics on monitoring during model serving

- Service metrics
- Model server metrics
- Payload logging
- Data monitoring, validation, drift detection
- Explainers

Service metrics

- Grafana/prometheus
- Metrics
  - Latency
  - Success rate
  - # invocations
- Dimensions
  - By model
  - By model version



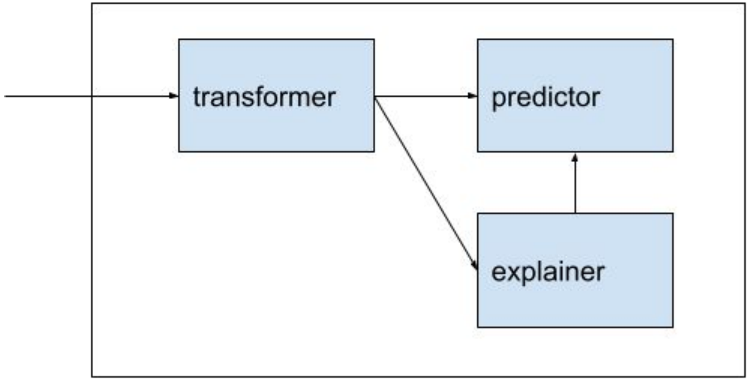
Service metrics

- Jaeger
  - Transactions across µs
- Component latency
  - Transformer
  - Predictor

JAEGER

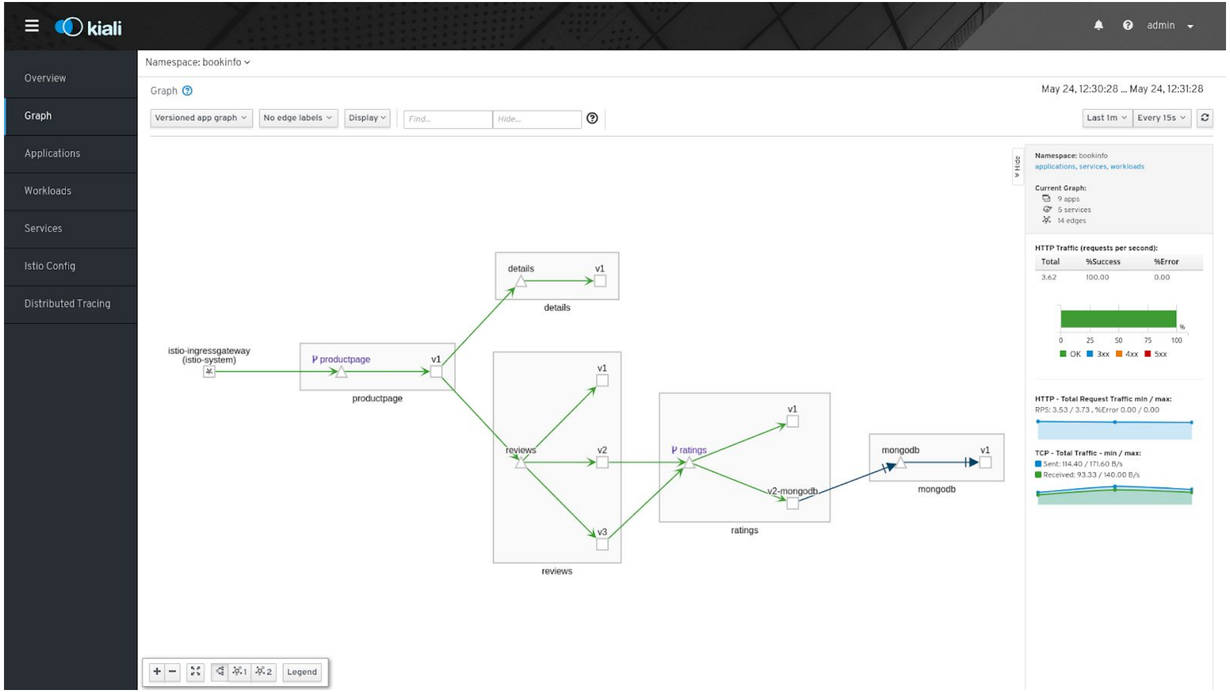
OpenTelemetry

Model serving components



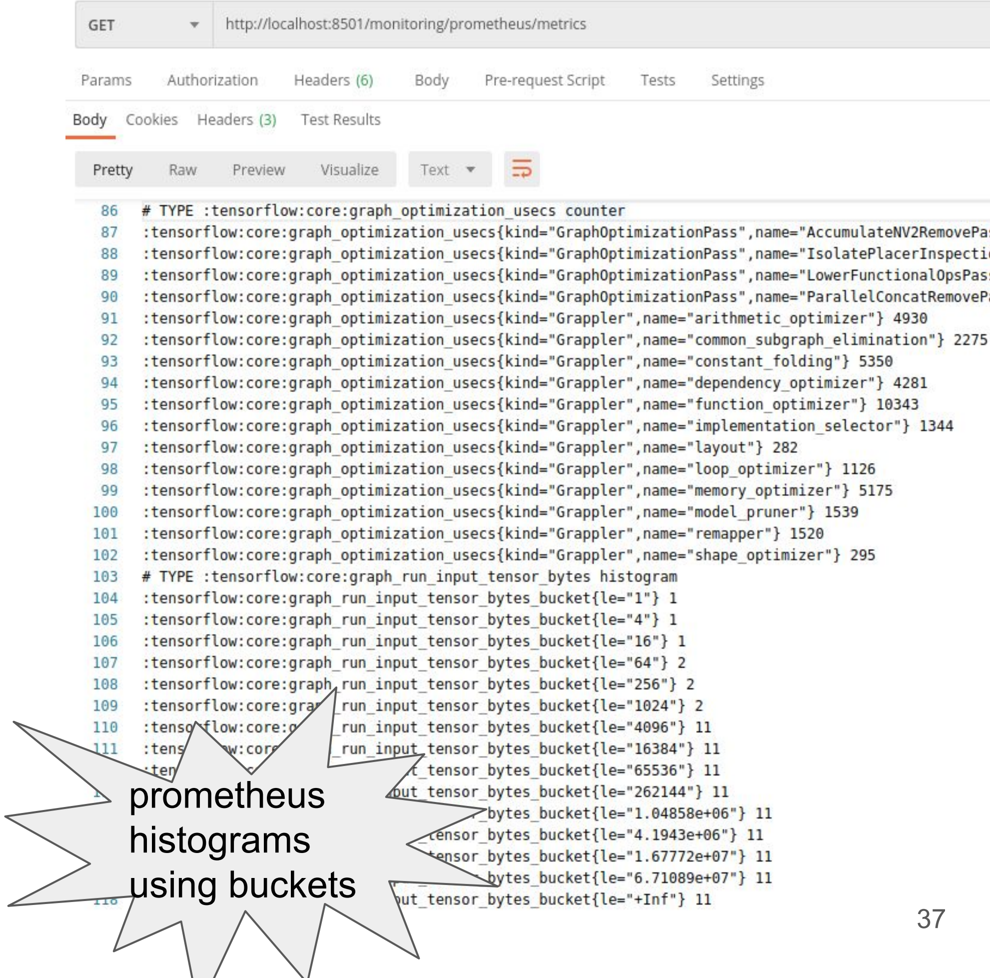
Service metrics

- Kiali
  - Routing flow
- Multi-model
  - % of traffic per model

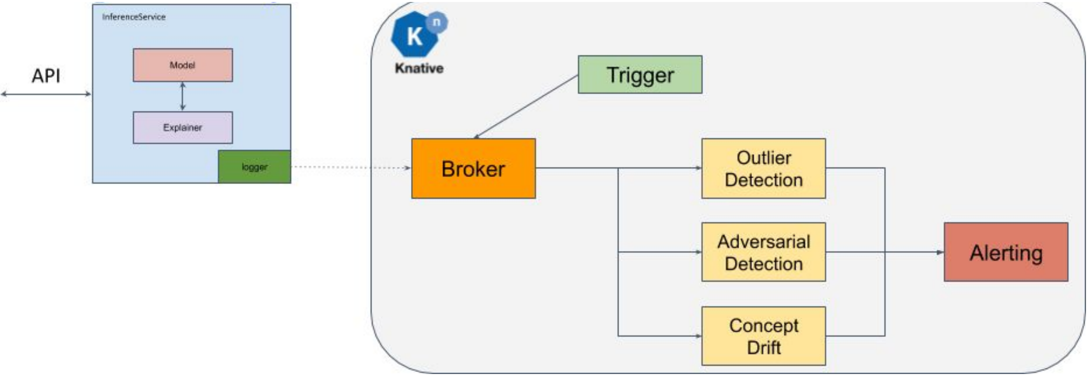


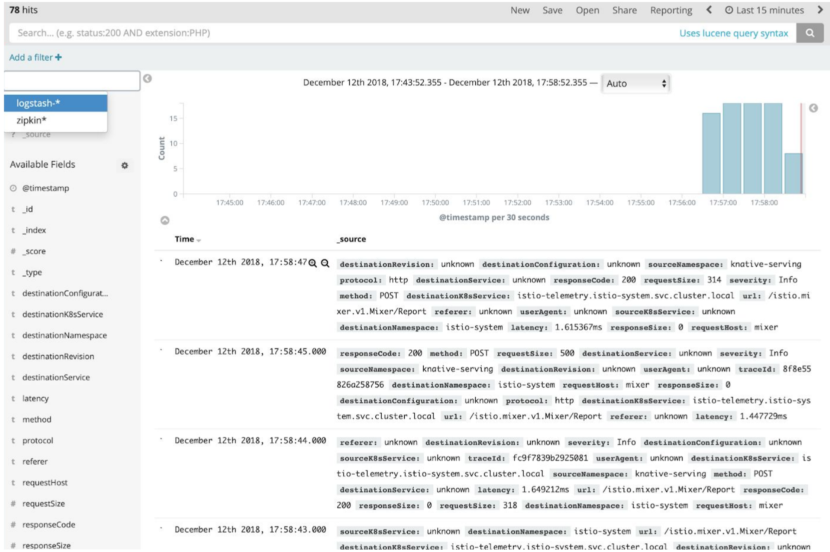
Model server metrics

- Model load latency (init/restore graph)
- Graph optimization, grappler
- Graph run time, graph runs
- Warmup latency



Payload Logging



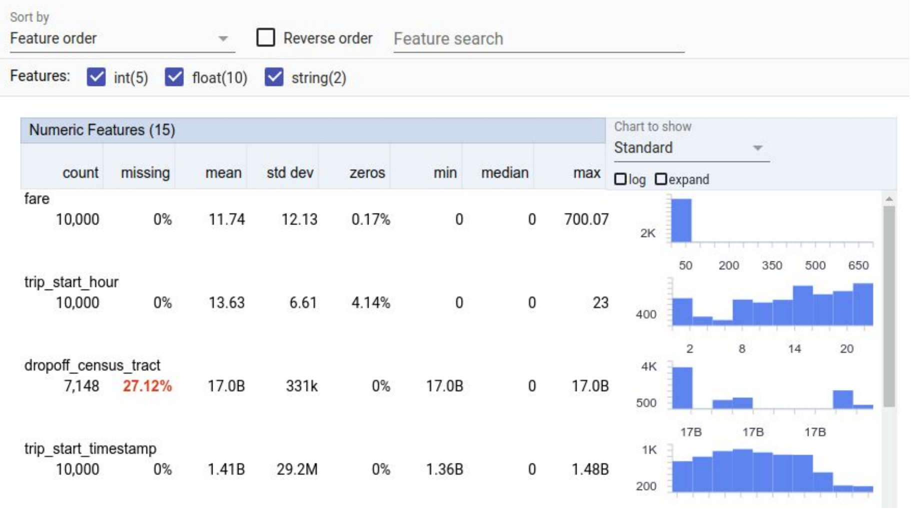


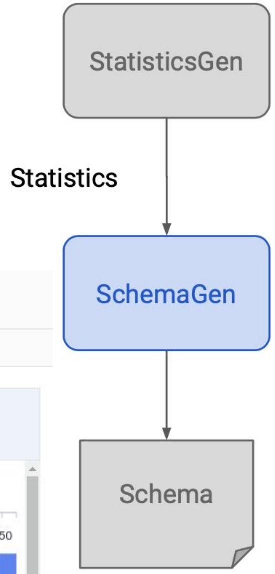
Data monitoring - drift detection

- Descriptive statistics
- Schema
- Data anomalies

```
feature {
  name: "fare"
  value_count {
        min: 0
        max: 700
  }
  type: INT
  presence {
        min_fraction: 1.0
        min_count: 1
  }
}
```

```
string_domain {
  name: "payment"
  value: "cash"
  value: "creditcard"
}
```

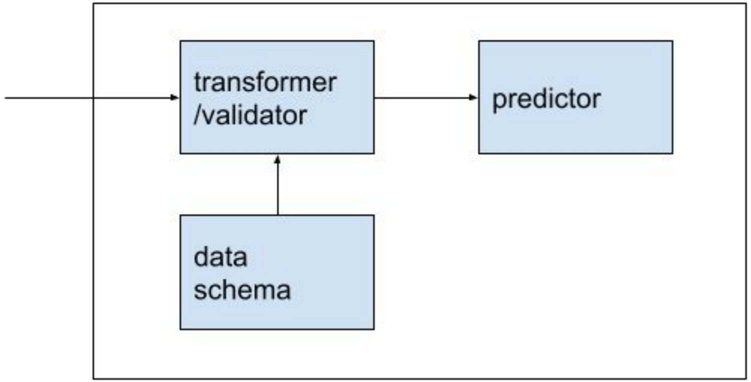




https://www.tensorflow.org/tfx/tutorials/data_validation/tfdv_basic

Data validation during serving

- Data schema




```
feature {
  name: "fare"
  value_count {
        min: 0
        max: 700
  }
  type: INT
  presence {
        min_fraction: 1.0
        min_count: 1
  }
}
```

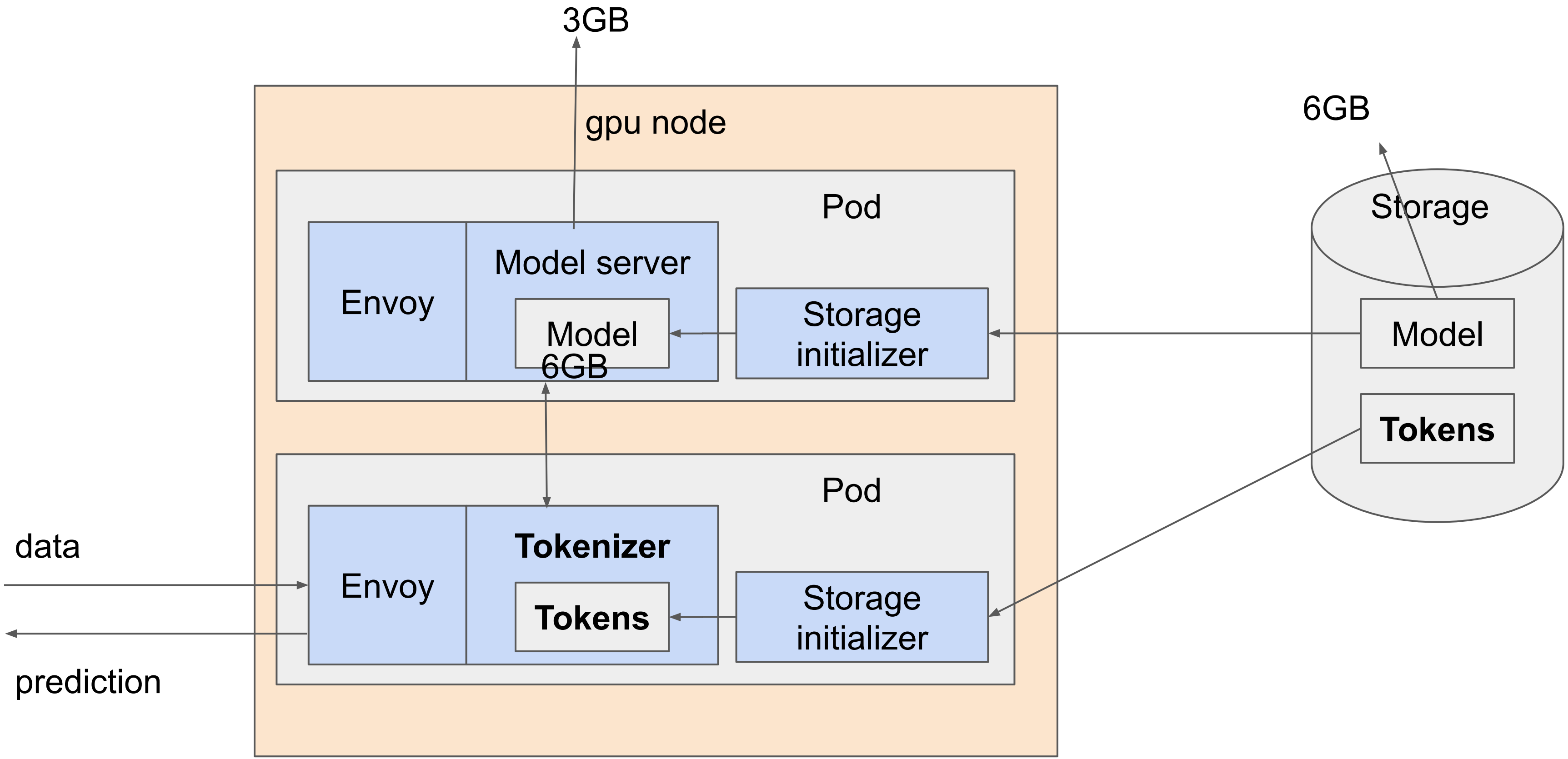
```
string_domain {
  name: "payment"
  value: "cash"
  value: "creditcard"
}
```

Scaling techniques

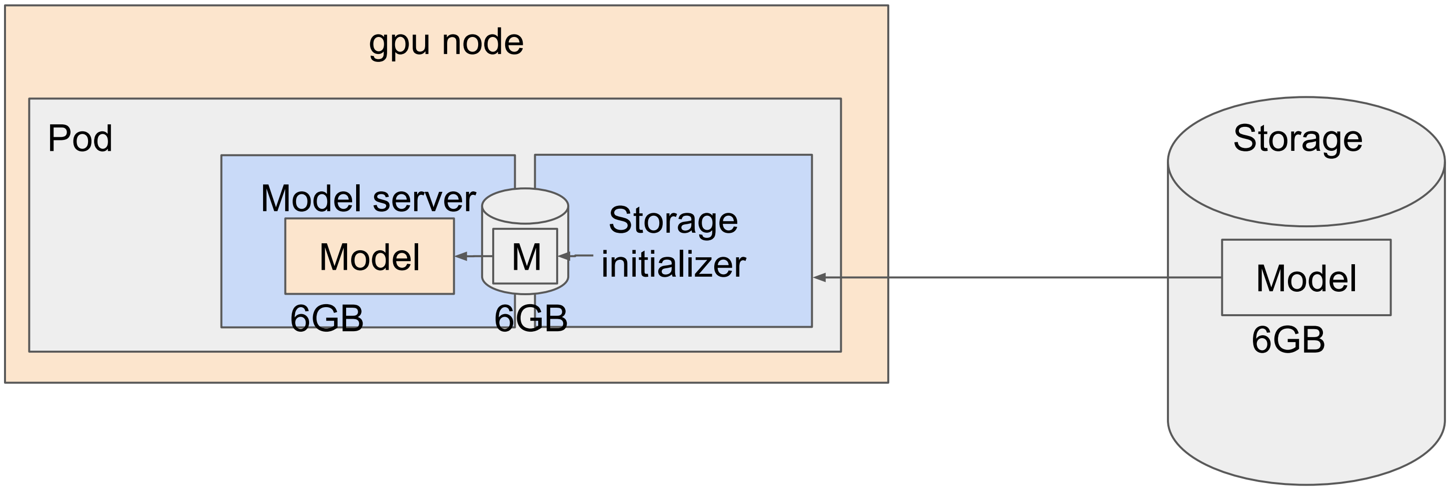
Model size

Linear	2 parameters	8 bytes	
Polynomial	3 parameters	12 bytes	
Simple NN	1.000.000 parameters	4.000.000 bytes	4MB
BERT	340.000.000 parameters	1.360.000.000 bytes	1.35GB
GPT-2	1.500.000.000 parameters	6.000.000.000 bytes	6GB

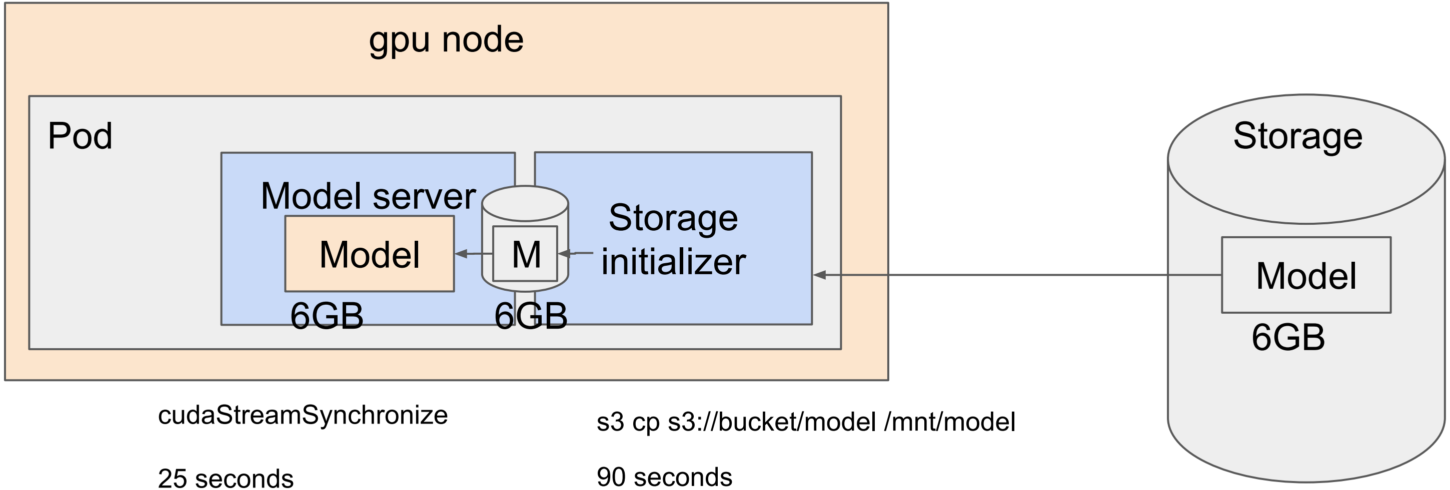
GPT-2 Deployment



GPT-2 Deployment: model load



GPT-2 Deployment: model load



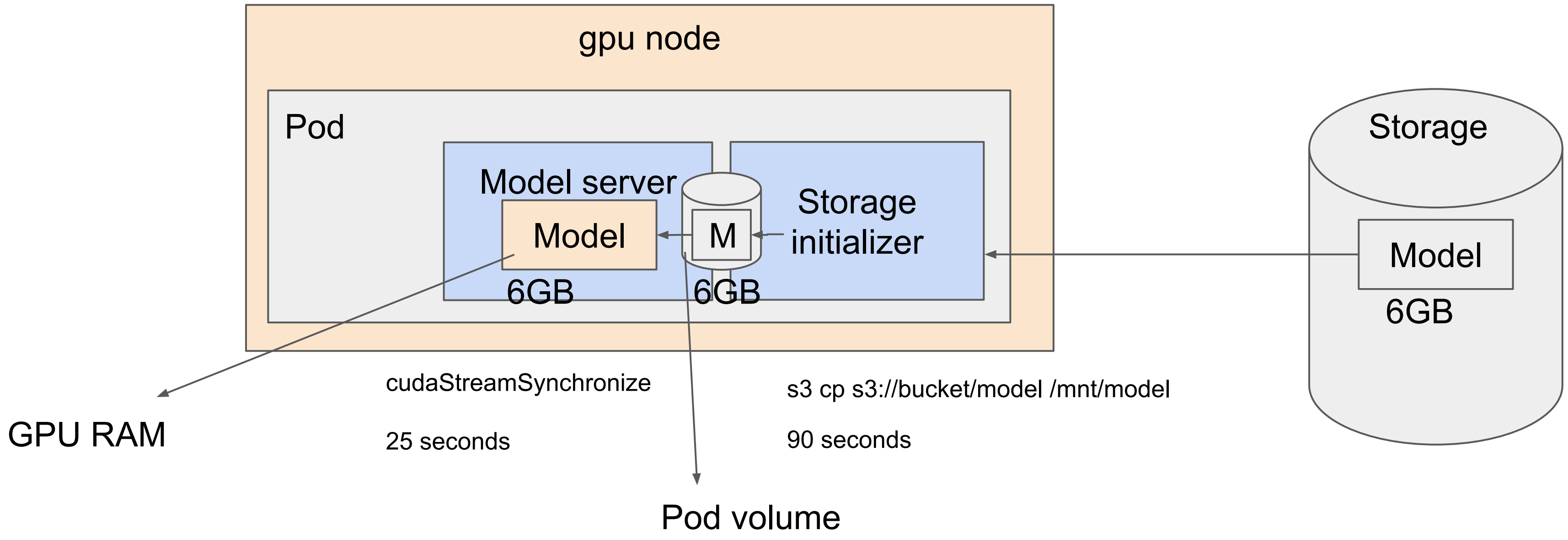
cudaStreamSynchronize

25 seconds

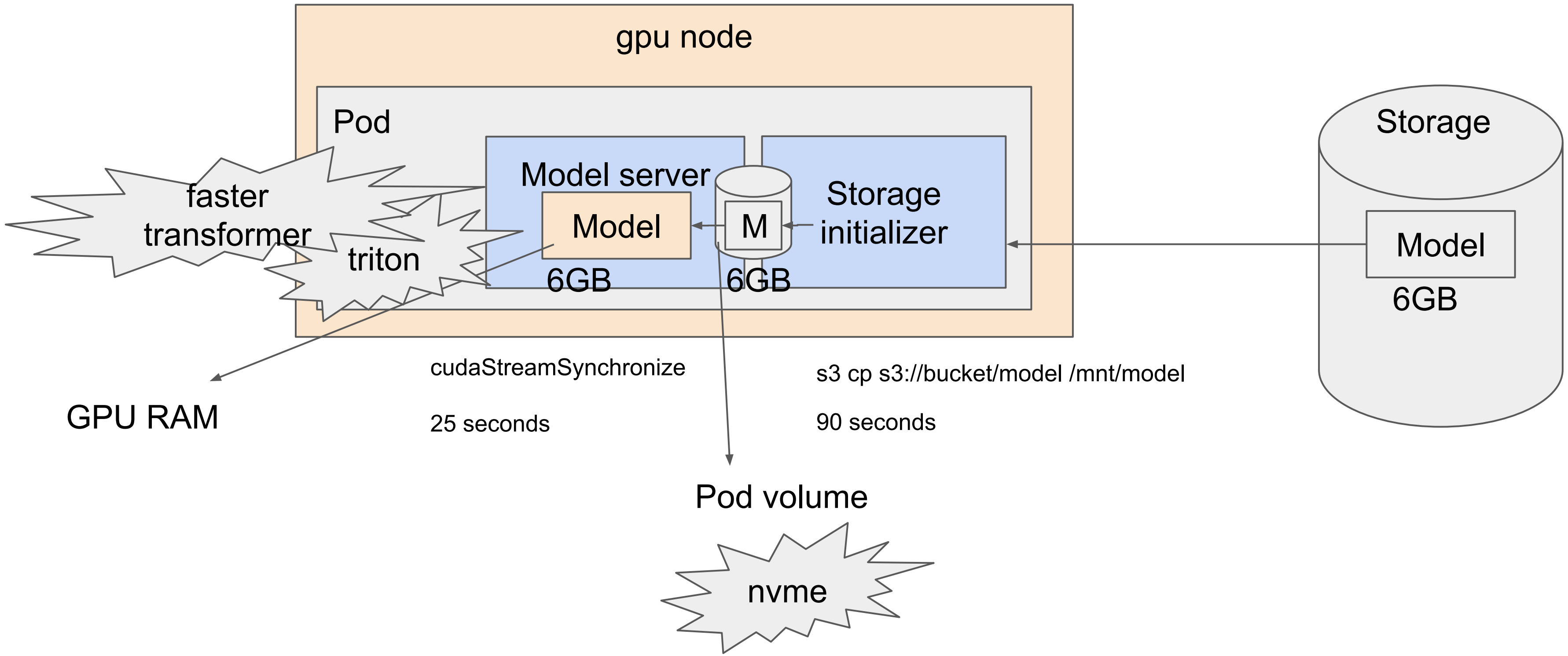
s3 cp s3://bucket/model /mnt/model

90 seconds

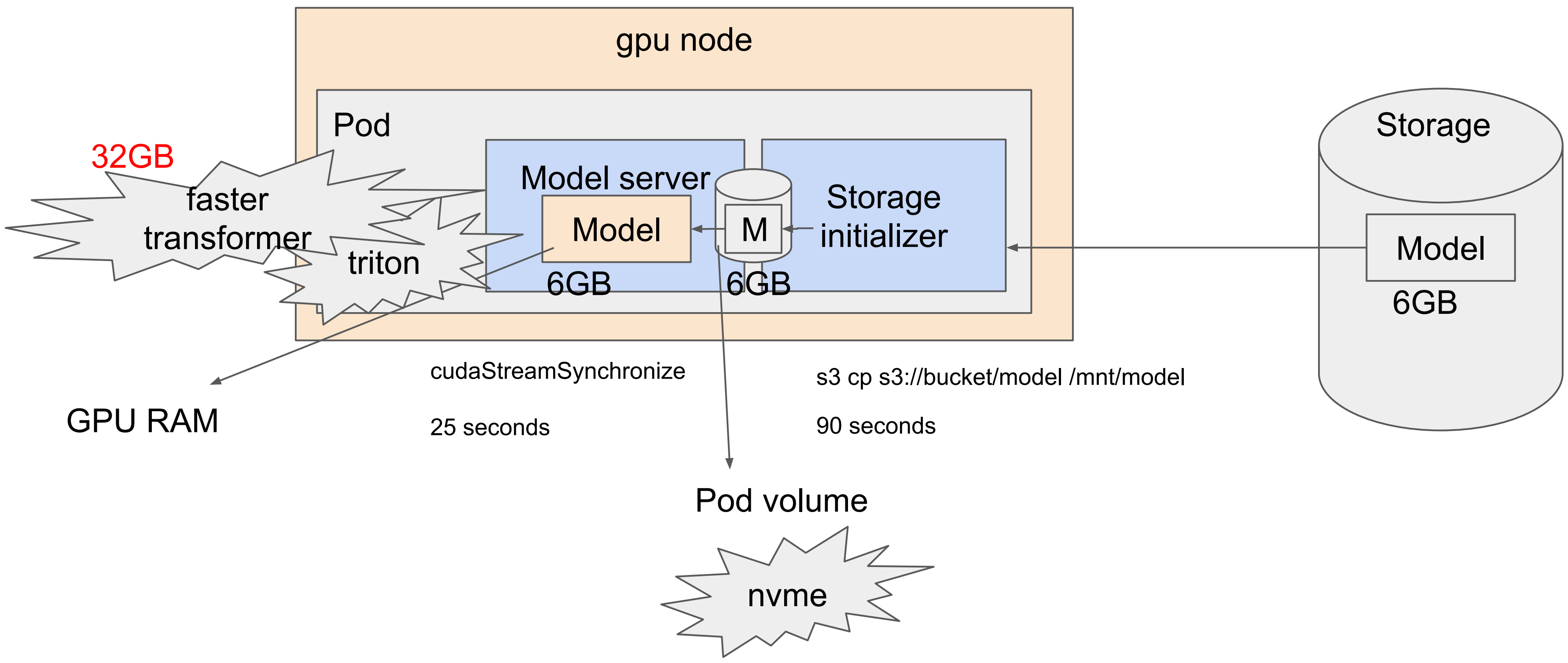
GPT-2 Deployment: model load



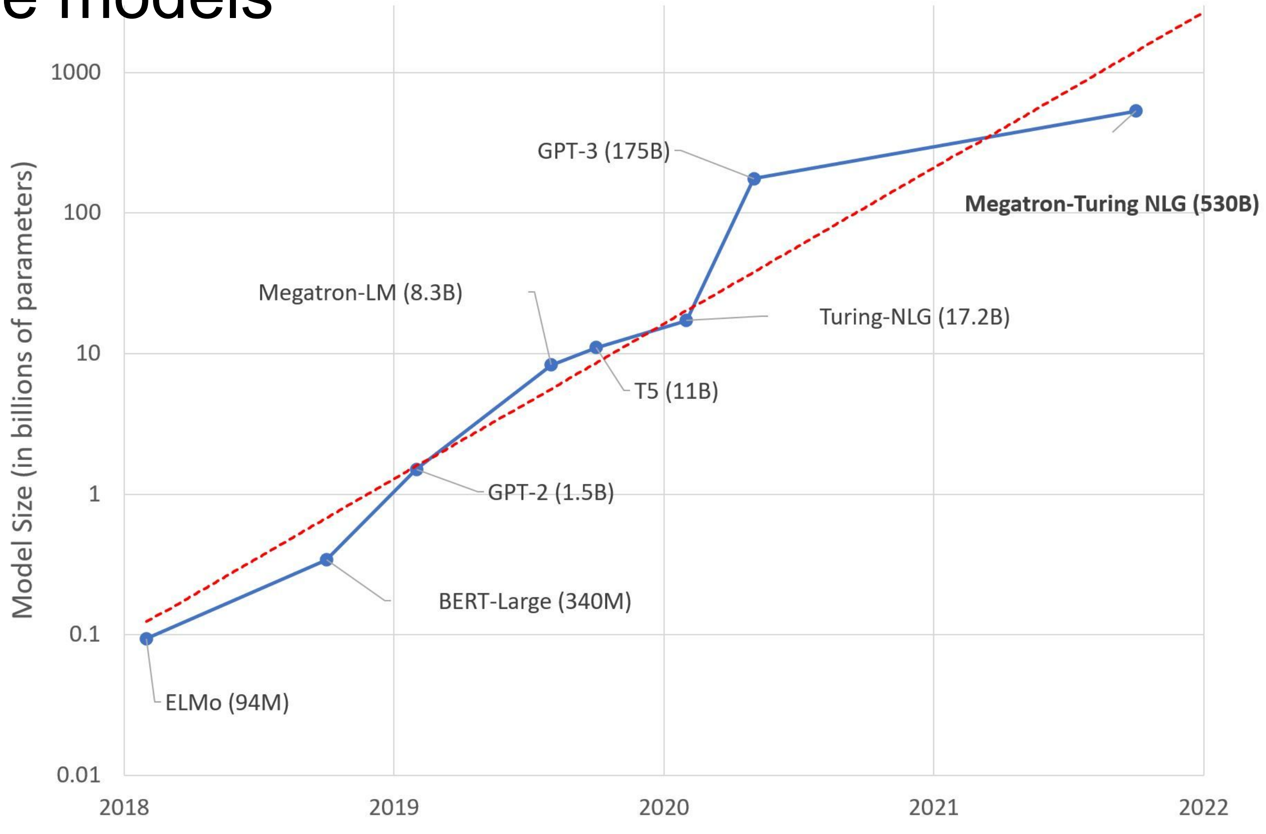
GPT-2 Deployment: model load hints



GPT-2 Deployment: model load hints



Language models

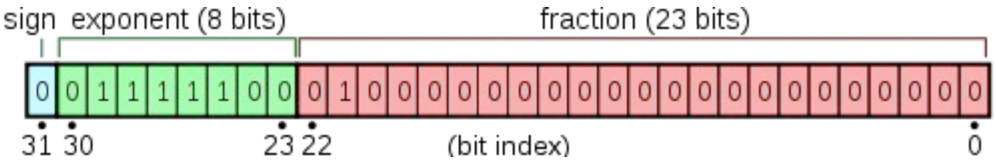


Source: https://huggingface.co/blog/large-language-models

Model size

Linear	2 parameters	8 bytes	
Polynomial	3 parameters	12 bytes	
Simple NN	1.000.000 parameters	4.000.000 bytes	4MB
BERT	340.000.000 parameters	1.360.000.000 bytes	1.35GB
GPT-2	1.500.000.000 parameters	6.000.000.000 bytes	6GB
GPT-3	175.000.000.000 parameters	700.000.000.000 bytes	700GB

Floating point precision



FP32

FP16

https://en.wikipedia.org/wiki/Single-precision_floating-point_format
https://en.wikipedia.org/wiki/Half-precision_floating-point_format

Post-training Quantization

- Reduce model size and latency
- Degradation of accuracy

```
model=tf.keras.models.load_model('.')
model.weights[0][0].numpy()[:10]
array([-0.02062028, -0.00791041,  0.02673002,  0.06981003, -0.06624269,
      -0.01446035,  0.01503156,  0.04210582,  0.0458509 ,  0.02943908],
      dtype=float32)


converter = tf.lite.TFLiteConverter.from_saved_model('.')
converter.optimizations = [tf.lite.Optimize.DEFAULT]
converter.target_spec.supported_types = [tf.float16]
tflite_quant_model = converter.convert()
model1=tf.lite.Interpreter(model_content=tflite_quant_model)

model1.get_tensor(4).transpose()[0][:10]
array([-0.0206, -0.0079,  0.0267,  0.0698 , -0.0662 ,
       -0.0145, 0.0150,  0.0421 ,  0.0458,  0.0294],
       dtype=float16)
```

Post-training Quantization

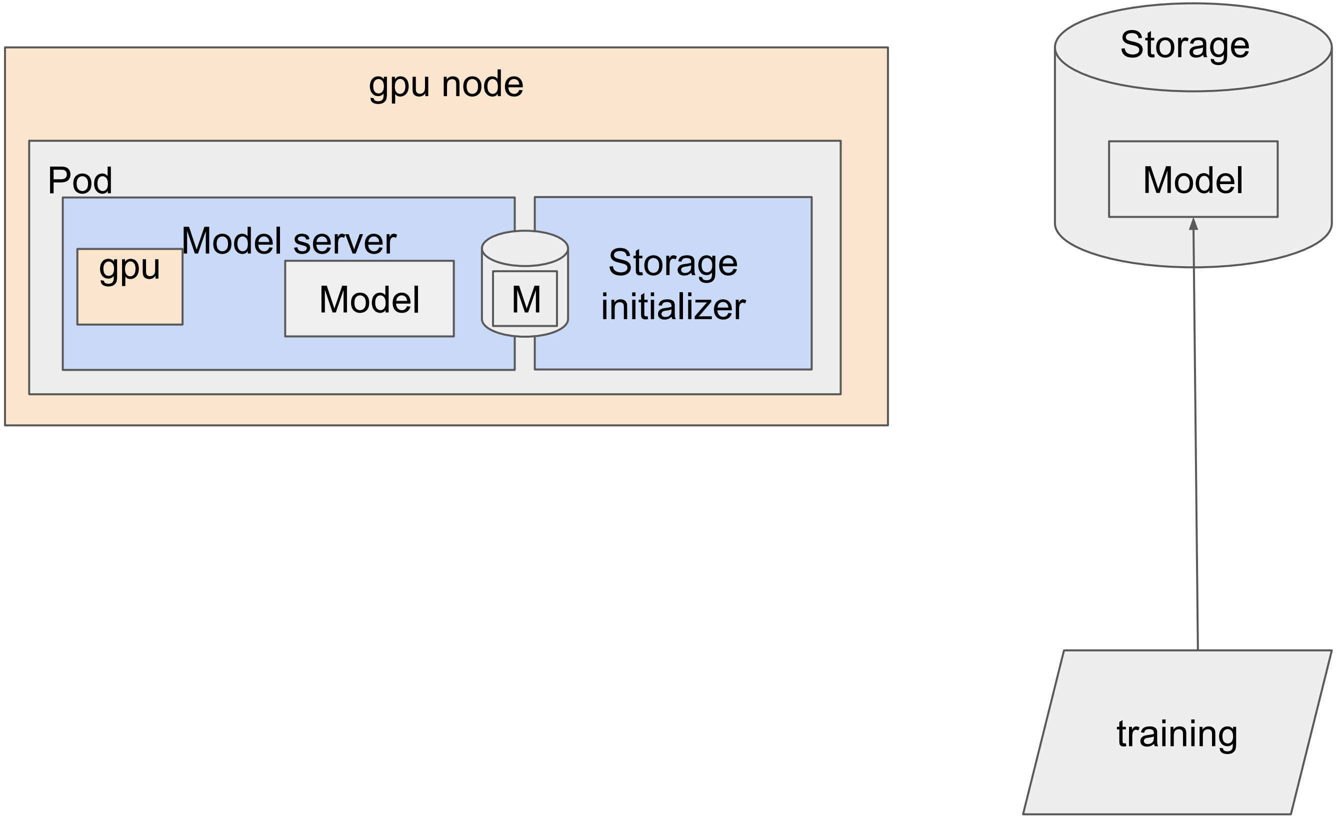
- Reduce model size and latency
- Degradation of accuracy

```
model=tf.keras.models.load_model('.')
model.weights[0][0].numpy()[:10]
array([-0.02062028, -0.00791041,  0.02673002,  0.06981003, -0.06
       -0.01446035,  0.01503156,  0.04210582,  0.0458509 ,  0.0294
       dtype=float32)


converter = tf.lite.TFLiteConverter.from_saved_model('.')
converter.optimizations = [tf.lite.Optimize.DEFAULT]
converter.target_spec.supported_types = [tf.float16]
tflite_quant_model = converter.convert()
model1=tf.lite.Interpreter(model_content=tflite_quant_model)

model1.get_tensor(4).transpose()[0][:10]
array([-0.02061, -0.00791,  0.02673,  0.0698 , -0.0662 ,
        -0.01446, 0.01503,  0.0421 ,  0.04584,  0.02943],
       dtype=float16)
```

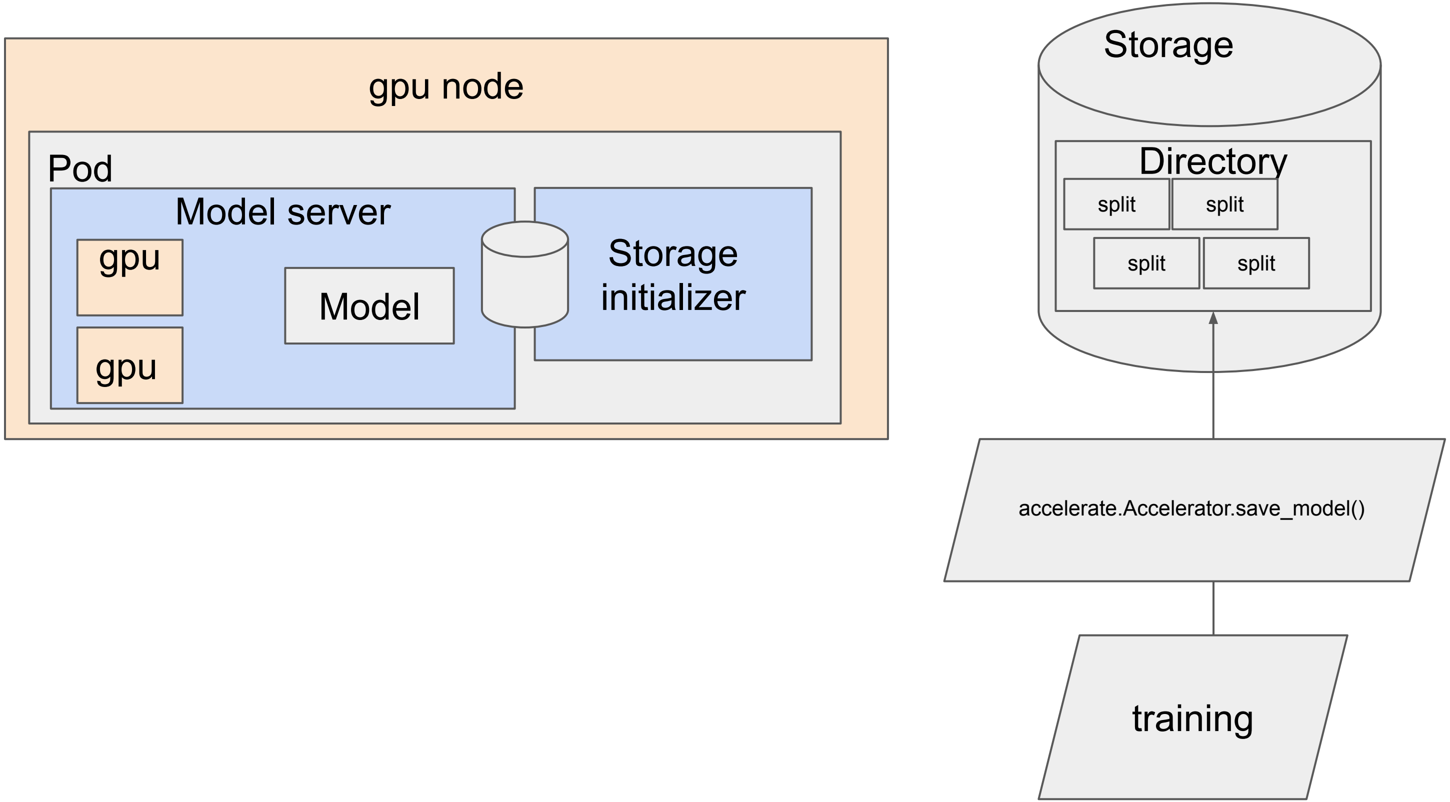
2MB

1MB

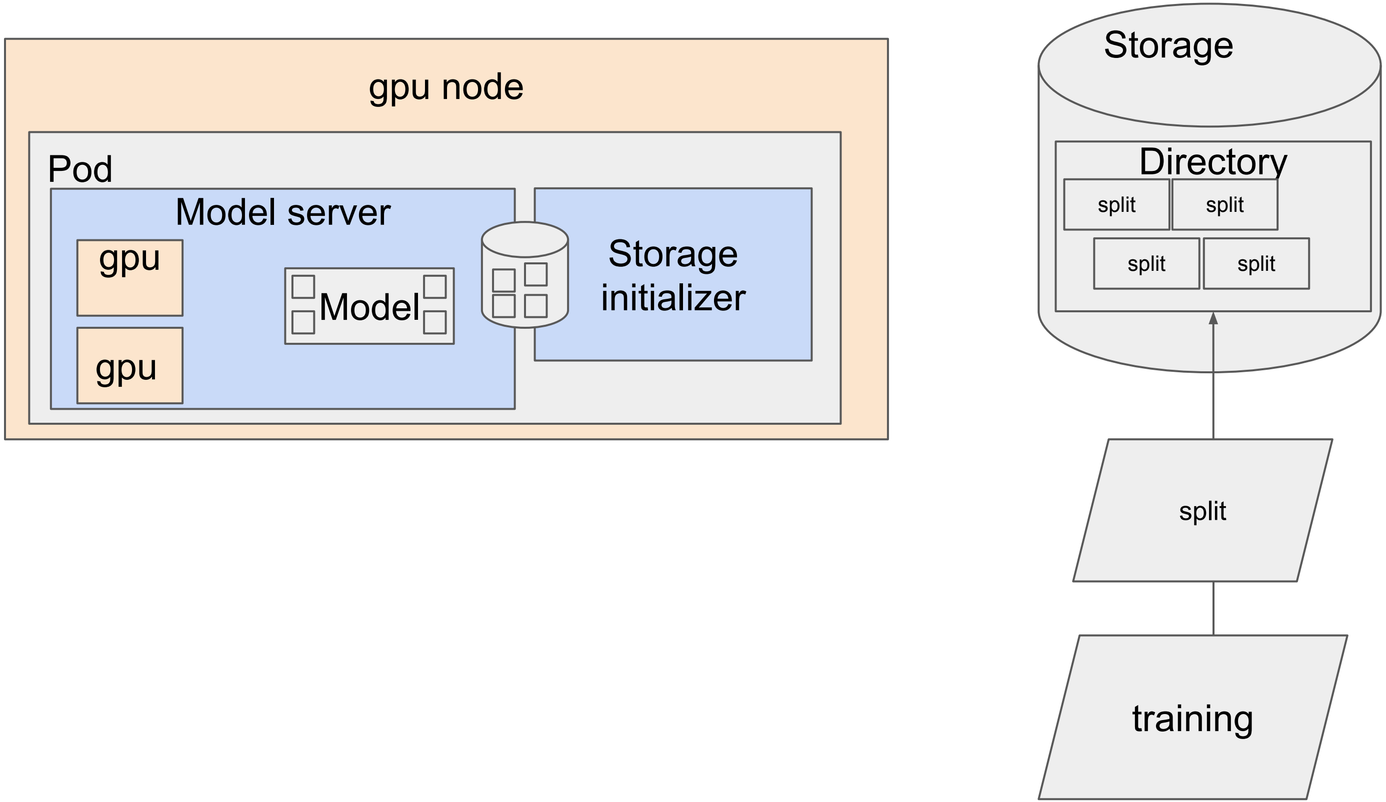
Checkpoints Sharding



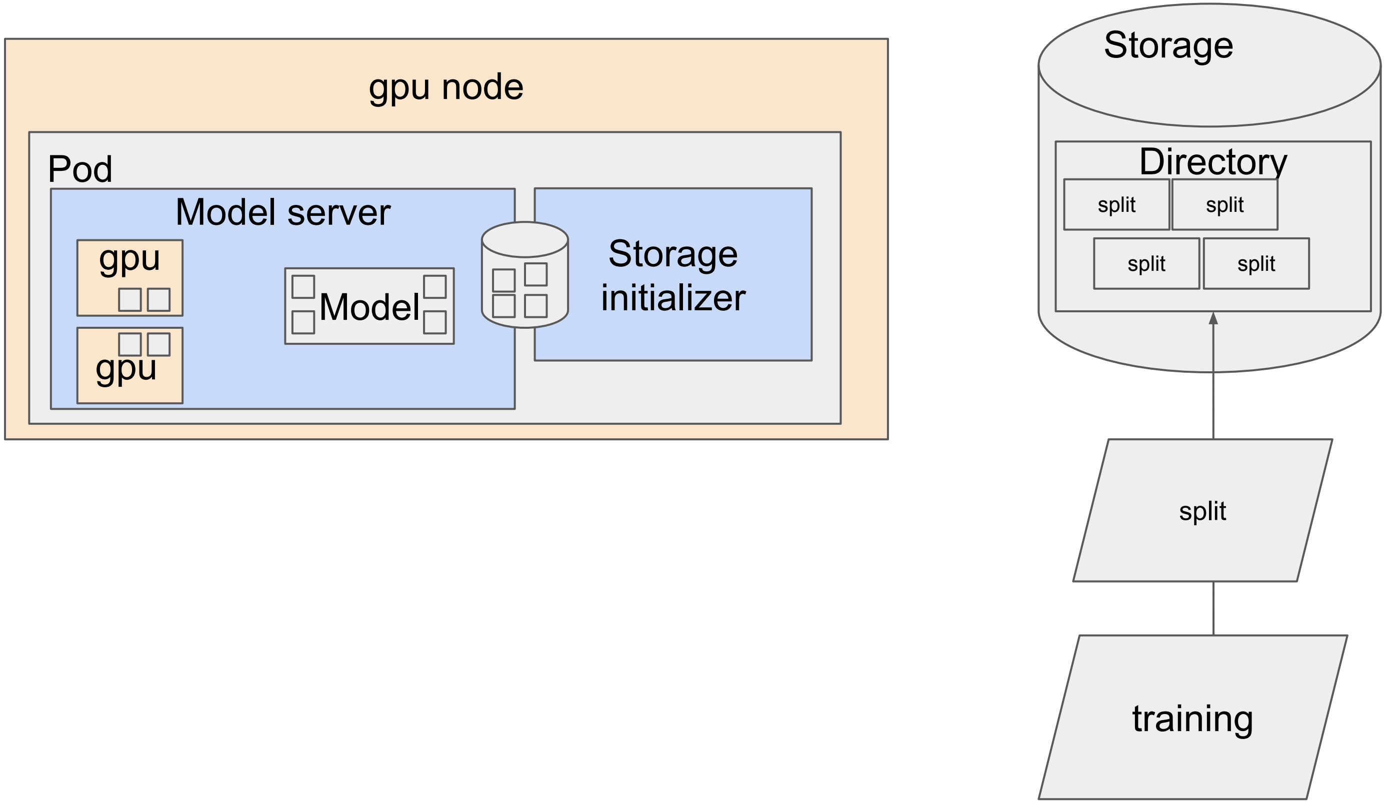
Checkpoints Sharding



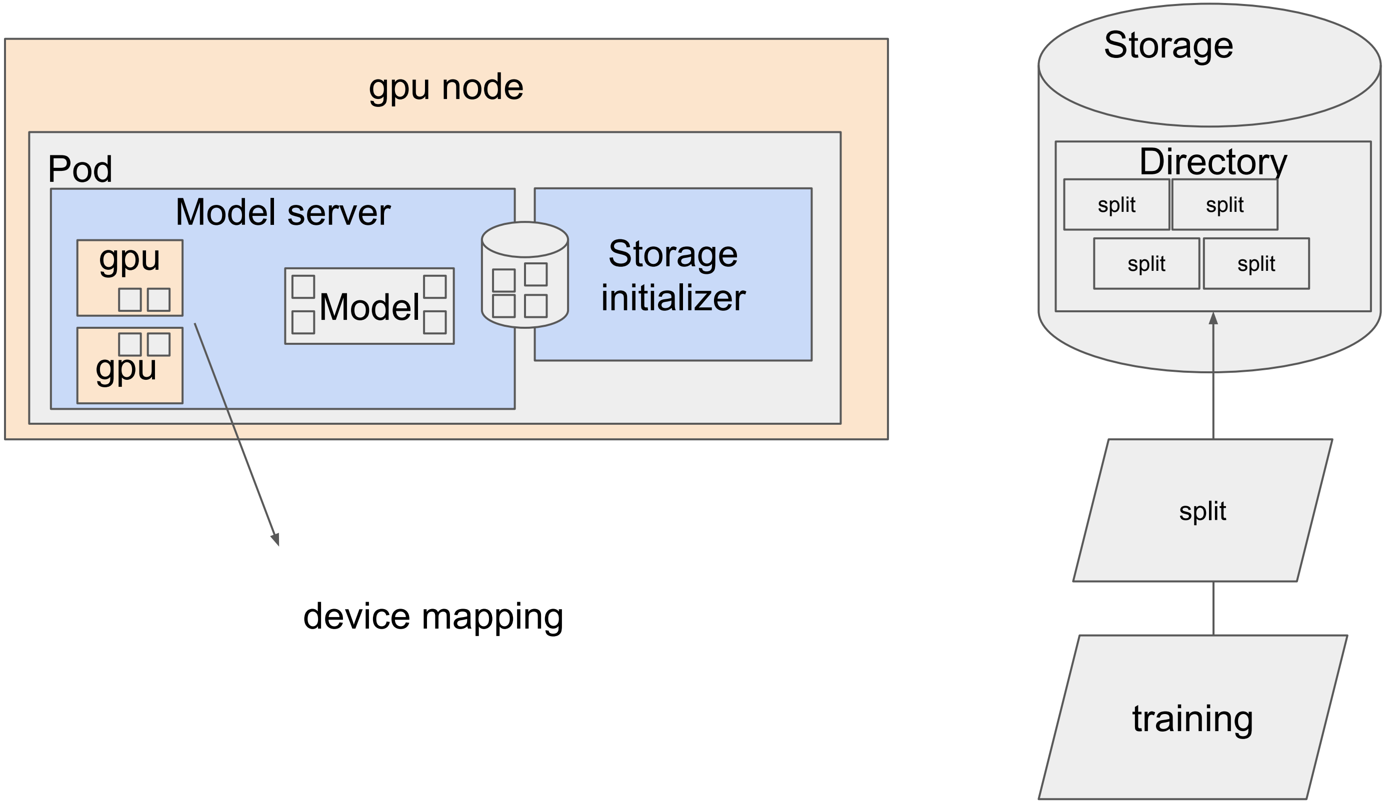
Checkpoints Sharding



Checkpoints Sharding



Checkpoints Sharding



Summary

- **Inference hardware requirements**
  - Cost: Requires significant computational resources with high end GPUs and large amount of memory
  - Latency: long response time up to tens of seconds

- **Model blob/file sizes in GBs(BLOOM):**
  - 176bln params = 360GB
  - 72 splits of 5GB which needs to mapped to multiple GPU devices

- **Model loading time**
  - From network (S3, minio) to instance disk
  - From instance disk to CPU RAM
  - From CPU RAM to GPU RAM

- **Model deployment**
  - Tokenizer/Predictor pattern

- **Model monitoring**
  - Service metrics
  - Model server metrics
  - Payload logging
  - Data monitoring, validation
  - Drift detection

- **Model Serving Runtime:**
  - FasterTransformer-Triton

- **Other scaling techniques**
  - Post training quantization
  - Checkpoints sharding