Деплой моделей. Nvidia Triton Server

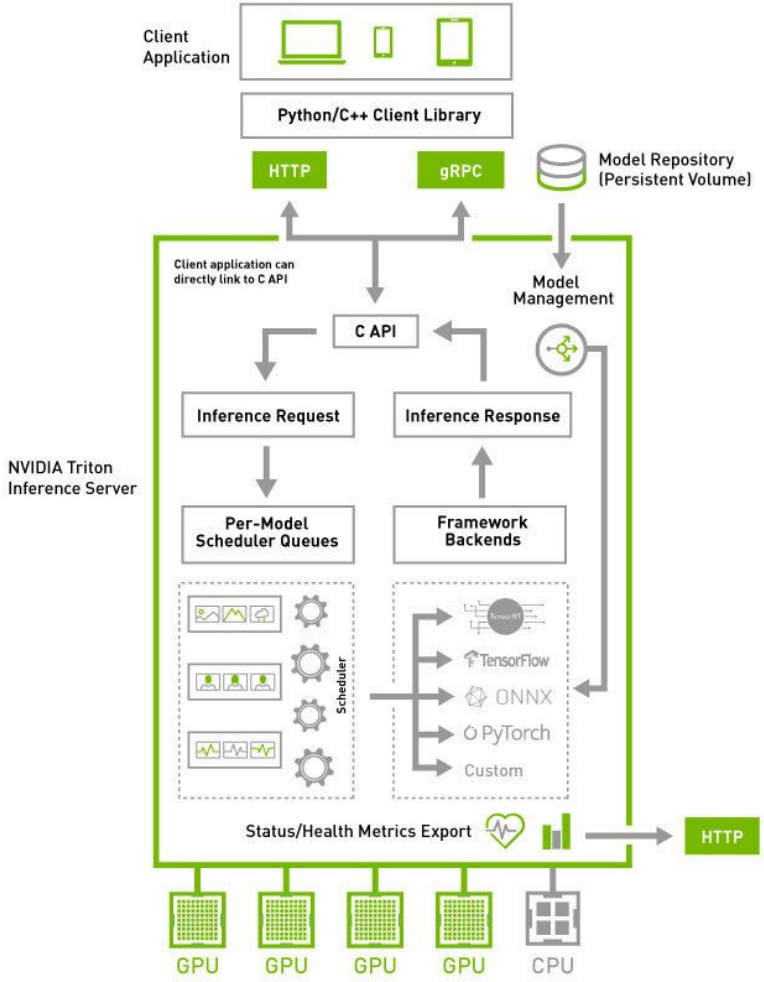
Astafurov Eugene

MIPT, MSU, fall 2023

Зачем?

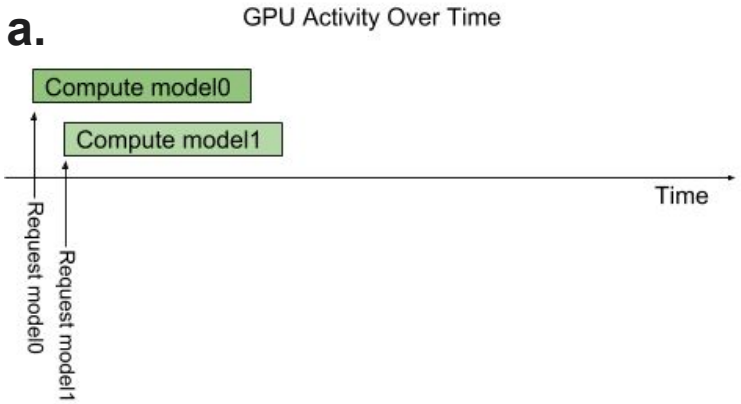
- Enterprise grade, security & api stability
- Concurrent model execution
- Dynamic batching
- Many supported frameworks out of the box
- Live updates
- Allowed both CPU and GPU instances
- Allowed multiple instances for the same model
- Support for arbitrary function execution
- Model ensembles out of the box
- Optimized and certified to deploy anywhere: Cloud, Datacenter, Edge,...

Архитектура

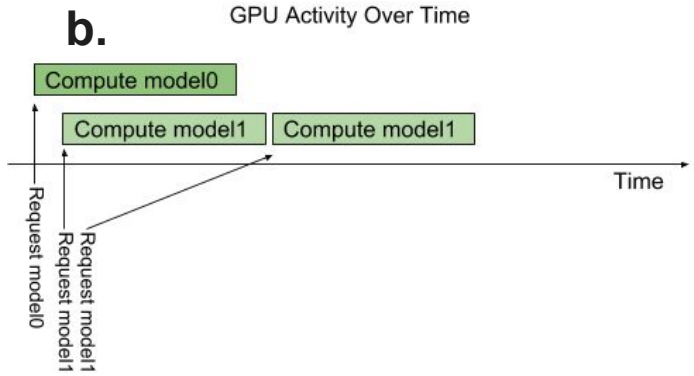


Concurrent Execution

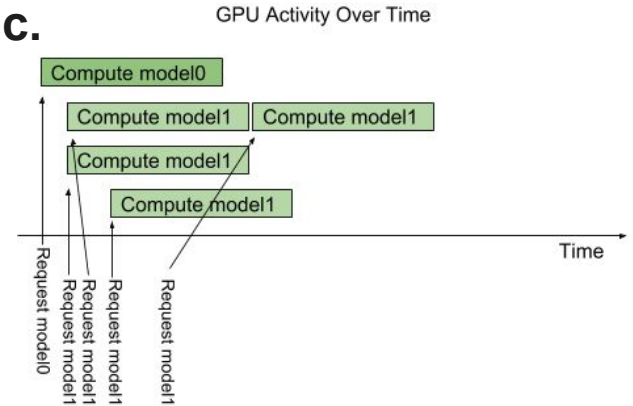
a.



b.



c.



Model repository

```
<model-repository-path>/
  <model-name>/
    [config.pbtxt]
    [<output-labels-file> ...]
    <version>/
      <model-definition-file>
    <version>/
      <model-definition-file>
    ...
  <model-name>/
    [config.pbtxt]
    [<output-labels-file> ...]
    <version>/
      <model-definition-file>
    <version>/
      <model-definition-file>
    ...
  ...
```

Model configuration

```
platform: "tensorrt_plan"
max_batch_size: 8


input [
    {
        name: "input0"
        data_type: TYPE_FP32
        dims: [ 16 ]
    },
    {
        name: "input1"
        data_type: TYPE_FP32
        dims: [ 16 ]
    }
]

output [
    {
        name: "output0"
        data_type: TYPE_FP32
        dims: [ 16 ]
    }
]
```

Explicit/implicit batch size

```
platform: "tensorrt_plan"
max_batch_size: 8


input [
    {
        name: "input"
        data_type: TYPE_FP32
        dims: [ 16 ]
    }
]

output [
    {
        name: "output"
        data_type: TYPE_FP32
        dims: [ 16 ]
    }
]
```

```
platform: "tensorrt_plan"
max_batch_size: 0


input [
    {
        name: "input"
        data_type: TYPE_FP32
        dims: [ 8, 16 ]
    }
]

output [
    {
        name: "output"
        data_type: TYPE_FP32
        dims: [ 8, 16 ]
    }
]
```

```
platform: "tensorrt_plan"
max_batch_size: 0


input [
    {
        name: "input"
        data_type: TYPE_FP32
        dims: [ -1, 16 ]
    }
]

output [
    {
        name: "output"
        data_type: TYPE_FP32
        dims: [ -1, 16 ]
    }
]
```

Inplace reshaping

```
platform: "tensorrt_plan"
max_batch_size: 8


input [
    {
        name: "input"
        data_type: TYPE_FP32
        dims: [ 1 ]
        reshape: { shape: [ ] }  # (batch_size, 1) -> (batch_size)
    }
]
```

Config Datatypes

Model Config	TensorRT	TensorFlow	ONNX Runtime	PyTorch	API	NumPy
TYPE_BOOL	kBOOL	DT_BOOL	BOOL	kBool	BOOL	bool
TYPE_UINT8	kUINT8	DT_UINT8	UINT8	kByte	UINT8	uint8
TYPE_UINT16		DT_UINT16	UINT16		UINT16	uint16
TYPE_UINT32		DT_UINT32	UINT32		UINT32	uint32
TYPE_UINT64		DT_UINT64	UINT64		UINT64	uint64
TYPE_INT8	kINT8	DT_INT8	INT8	kChar	INT8	int8
TYPE_INT16		DT_INT16	INT16	kShort	INT16	int16
TYPE_INT32	kINT32	DT_INT32	INT32	kInt	INT32	int32
TYPE_INT64		DT_INT64	INT64	kLong	INT64	int64
TYPE_FP16	kHALF	DT_HALF	FLOAT16		FP16	float16
TYPE_FP32	kFLOAT	DT_FLOAT	FLOAT	kFloat	FP32	float32
TYPE_FP64		DT_DOUBLE	DOUBLE	kDouble	FP64	float64
TYPE_STRING		DT_STRING	STRING		BYTES	dtype(object)
TYPE_BF16					BF16	

Version Policy

```
# Доступны все версии модели
version_policy: { all { }}

# Доступны только последние две
version_policy: { latest: { num_versions: 2}}

# Доступны первая и третья версии
version_policy: { specific: { versions: [1,3]}}
```

```
platform: "tensorrt_plan"
max_batch_size: 0


input [
    {
        name: "input"
        data_type: TYPE_FP32
        dims: [ -1, 16 ]
    }
]

output [
    {
        name: "output"
        data_type: TYPE_FP32
        dims: [ -1, 16 ]
    }
]

version_policy: { all { }}
```

Instance groups

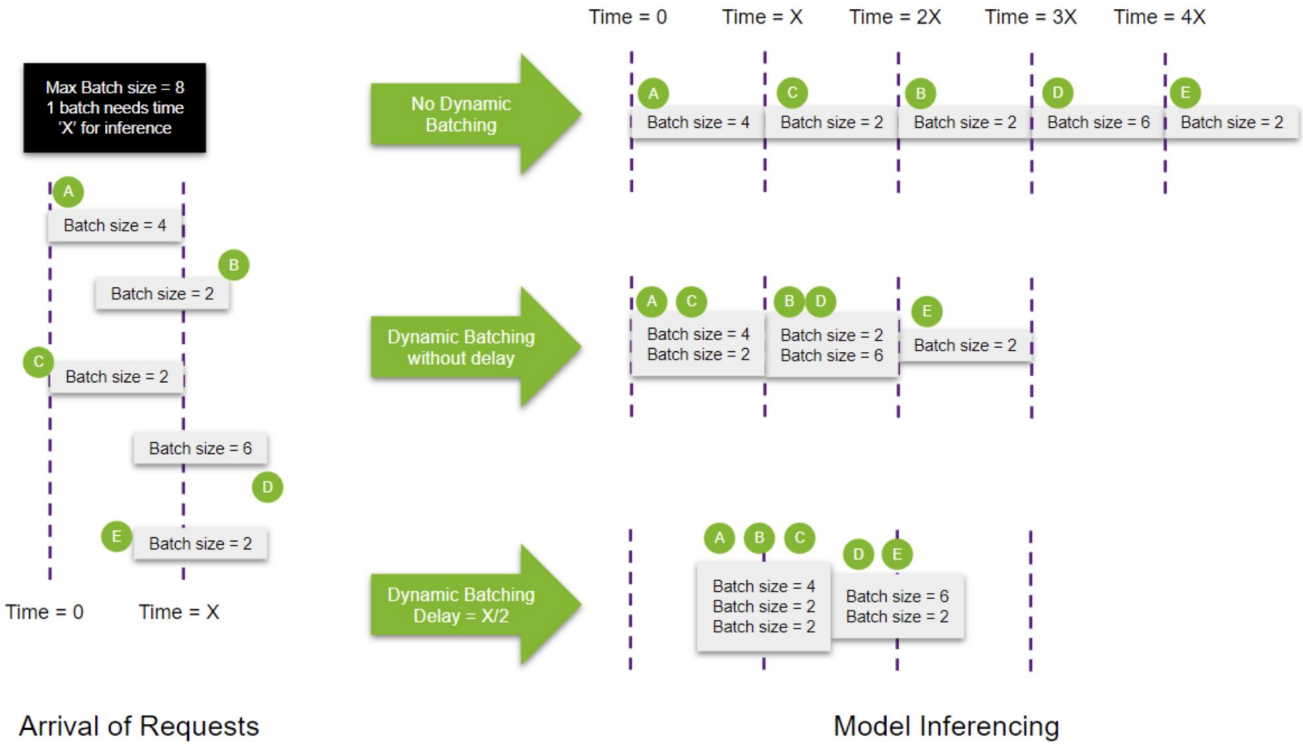
```
instance_group [
    {
        count: 2
        kind: KIND_GPU
    }
]
```

```
instance_group [
    {
        count: 1
        kind: KIND_GPU
        gpus: [ 0 ]
    },
    {
        count: 2
        kind: KIND_GPU
        gpus: [ 1, 2 ]
    }
]
```

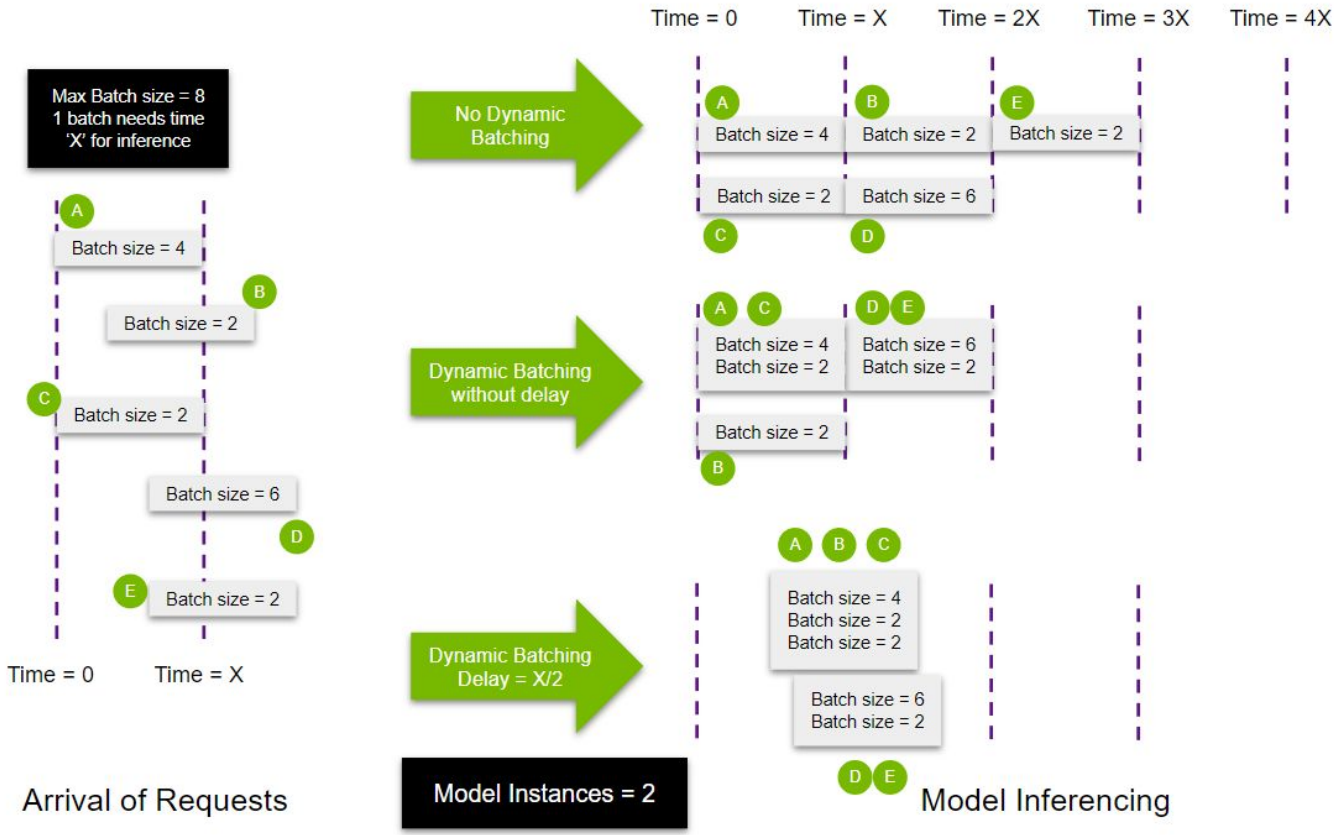
Dynamic batching

```
# Batching without delay
dynamic_batching { }

# Batching with max delay 0.1 ms
dynamic_batching {
    max_queue_delay_microseconds: 100
}

# Batching without deley to closest size
dynamic_batching {
    preferred_batch_size: [ 4, 8 ]
}
```

Dynamic Batching



Dynamic batching + Concurrent instance goups



Ensembling

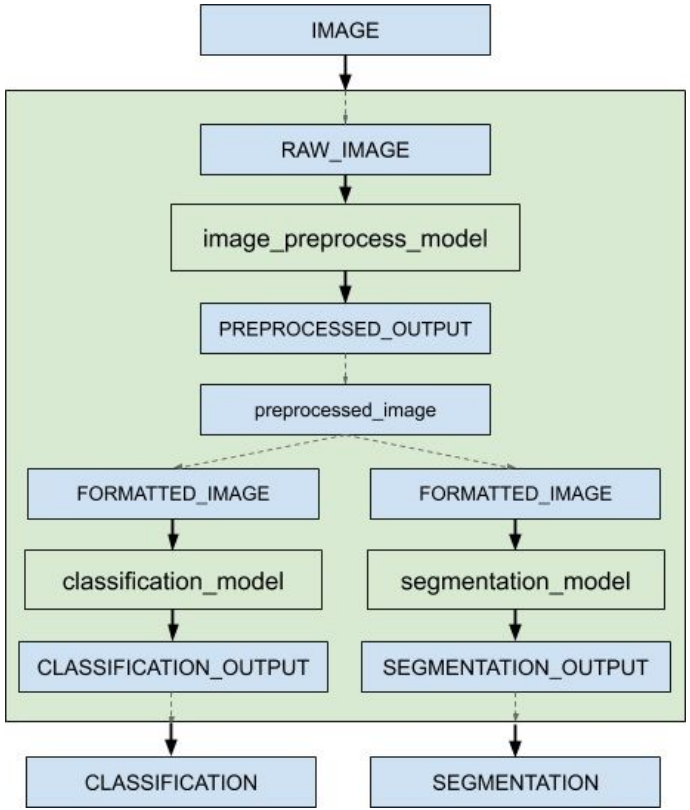


IMAGE
RAW_IMAGE
image_preprocess_model
PREPROCESSED_OUTPUT
preprocessed_image
FORMATTED_IMAGE
FORMATTED_IMAGE
classification_model
segmentation_model
CLASSIFICATION_OUTPUT
SEGMENTATION_OUTPUT
CLASSIFICATION
SEGMENTATION

Supported Backends

- TensorRT, TensorRT-LLM
- ONNX Runtime
- Tensorflow
- PyTorch
- OpenVINO
- Python
- Dali
- FIL (Forest Inference Library): xgboost, lightgbm, sklearn, cuML
- create your own?

Что еще?

- Stateful stateless models
- Ragged batching
- Decoupled mode
- Built-in rate limiter
- Response cache
- Model priority
- ....

Доп литература

- Git: https://github.com/triton-inference-server/server/tree/main
- User guide: https://github.com/triton-inference-server/server/tree/main/docs/user_guide

Спасибо за внимание!

Жду вопросов и обсуждений