Class InferenceProcessorOp

Table of contents

- Defined in File inference_processor.hpp

Nested Relationships

Nested Types

- Struct InferenceProcessorOp::DataMap
- Struct InferenceProcessorOp::DataVecMap

Inheritance Relationships

Base Type

- `public holoscan::Operator` (Class Operator)

Class Documentation

class InferenceProcessorOp : public holoscan::Operator

Inference Processor Operator class to perform operations per input tensor.

==Named Inputs==

- **receivers** : multi-receiver accepting `nvidia::gxf::Tensor` (s)
  - Any number of upstream ports may be connected to this `receivers` port. The operator will search across all messages for tensors matching those specified in `in_tensor_names`. These are the set of input tensors used by the processing operations specified in `process_map`.

==Named Outputs==

- **transmitter** : `nvidia::gxf::Tensor` (s)
  - A message containing tensors corresponding to the processed results from operations will be emitted. The names of the tensors transmitted correspond to those in `out_tensor_names`.

==Parameters==

- **allocator**: Memory allocator to use for the output.
- **process_operations**: Operations (`DataVecMap`) in sequence on tensors.
- **processed_map**: Input-output tensor mapping (`DataVecMap`)
- **in_tensor_names**: Names of input tensors (`std::vector<std::string>`) in the order to be fed into the operator. Optional.
- **out_tensor_names**: Names of output tensors (`std::vector<std::string>`) in the order to be fed into the operator. Optional.
- **input_on_cuda**: Whether the input buffer is on the GPU. Optional (default: `false`).
- **output_on_cuda**: Whether the output buffer is on the GPU. Optional (default: `false`).
- **transmit_on_cuda**: Whether to transmit the message on the GPU. Optional (default: `false`).
- **cuda_stream_pool**: `holoscan::CudaStreamPool` instance to allocate CUDA streams. Optional (default: `nullptr`).
- **config_path**: File path to the config file. Optional (default: `""`).
- **disable_transmitter**: If `true` , disable the transmitter output port of the operator. Optional (default: `false`).

==Device Memory Requirements==

When using this operator with a `BlockMemoryPool` , `num_blocks` must be greater than or equal to the number of output tensors that will be produced. The `block_size` in bytes must be greater than or equal to the largest output tensor (in bytes). If `output_on_cuda` is true, the blocks should be in device memory (

`storage_type`=1), otherwise they should be CUDA pinned host memory (`storage_type`=0).

Public Functions

HOLOSCAN_OPERATOR_FORWARD_ARGS (InferenceProcessorOp)
InferenceProcessorOp()=default

virtual void setup(OperatorSpec &spec) override

Define the operator specification.

Parameters

spec – The reference to the operator specification.

virtual void initialize() override

Initialize the operator.

This function is called when the fragment is initialized by Executor::initialize_fragment().

virtual void start() override

Implement the startup logic of the operator.

This method is called multiple times over the lifecycle of the operator according to the order defined in the lifecycle, and used for heavy initialization tasks such as allocating memory resources.

virtual void compute(InputContext &op_input, OutputContext &op_output, ExecutionContext &context) override

Implement the compute method.

This method is called by the runtime multiple times. The runtime calls this method until the operator is stopped.

Parameters

- **op_input** – The input context of the operator.

- **op_output** – The output context of the operator.
- **context** – The execution context of the operator.

struct DataMap

DataMap specification

Public Functions

DataMap() = default

inline explicit operator bool() const noexcept

inline void insert(const std::string &key, const std::string &value)

inline std::map<std::string, std::string> get_map() const

Public Members

std::map<std::string, std::string> mappings_

struct DataVecMap

DataVecMap specification

Public Functions

DataVecMap() = default

inline explicit operator bool() const noexcept

inline void insert(const std::string &key, const std::vector<std::string> &value)

inline std::map<std::string, std::vector<std::string>> get_map() const

Public Members

std::map<std::string, std::vector<std::string>> mappings_