Holoscan and GXF

Table of contents

Design differences

There are 2 main elements at the core of Holoscan and GXF designs:

1. How to define and execute application graphs

2. How to define nodes' functionality

How Holoscan SDK interfaces with GXF on those topics varies as Holoscan SDK evolves, as described below:

Holoscan SDK v0.2

Holoscan SDK was tightly coupled with GXF's existing interface:

1. GXF application graphs are defined in **YAML** configuration files. **GXE** (Graph Execution Engine) is used to execute AI application graphs. Its inputs are the YAML configuration file, and a list of GXF Extensions to load as plugins (manifest yaml file). This design allows entities to be swapped or updated without needing to recompile an application.

2. Components are made available by registering them within a **GXF extension**, each of which maps to a shared library and header(s).

Those concepts are illustrated in the GXF by example section.

The only additions that Holoscan provided on top of GXF were:

- domain specific reference applications

- new extensions

- CMake configurations for building extensions and applications

Holoscan SDK v0.3

The Holoscan SDK shifted to provide a more developer-friendly interface with C++:

1. GXF application graphs, memory allocation, scheduling, and message routing can be defined using a C++ API, with the ability to read parameters and required GXF

extension names from a YAML configuration file. The backend used is still GXF as Holoscan uses the GXF C API, but this bypasses GXE and the full YAML definition.

2. The C++ **Operator** class was added to wrap and expose GXF extensions to that new application interface (See dev guide).

Holoscan SDK v0.4

The Holoscan SDK added Python wrapping and native operators to further increase ease of use:

1. The C++ API is also wrapped in Python. GXF is still used as the backend.

2. The Operator class supports **native operators**, i.e. operators that do not require to implement and register a GXF Extension. An important feature is the ability to support messaging between native and GXF operators without any performance loss (i.e. zero-copy communication of tensors).

Holoscan SDK v0.5

1. The built-in Holoscan GXF extensions are loaded automatically and don't need to be listed in the YAML configuration file of Holoscan applications. This allows Holoscan applications to be defined without requiring a YAML configuration file.

2. No significant changes to build operators. However, most built-in operators were switched to native implementations, with the ability to convert native operators to GXF codelets for GXF application developers.

Holoscan SDK v1.0

1. The remaining GXF-based DemosiacOp operator was switched to a native implementation. Now all operators provided by the SDK are native operators.

Current limitations

Here is a list of GXF capabilities not yet available in the Holoscan SDK which are planned to be supported in future releases:

- Job Statistics

The GXF capabilities below are not available in the Holoscan SDK either. There is no plan to support them at this time:

- Graph Composer
- Behavior Trees
- Epoch Scheduler
- Target Time Scheduling Term
- Multi-Message Available Scheduling Term
- Expiring Message Available Scheduling Term

© Copyright 2022-2024, NVIDIA.. PDF Generated on 06/06/2024