Video Replayer (Distributed)

Table of contents

List of Figures

In this example, we extend the previous video replayer application into a multi-node distributed application. A distributed application is made up of multiple Fragments (`C++` / `Python`), each of which may run on its own node.

In the distributed case we will:

- create one fragment that loads a video file from disk using **VideoStreamReplayerOp** operator
- create a second fragment that will display the video using the **HolovizOp** operator

These two fragments will be combined into a distributed application such that the display of the video frames could occur on a separate node from the node where the data is read.

> **Note**
>
> The example source code and run instructions can be found in the examples directory on GitHub, or under `/opt/nvidia/holoscan/examples` in the NGC container and the debian package, alongside their executables.

Operators and Workflow

Here is the diagram of the operators and workflow used in this example.

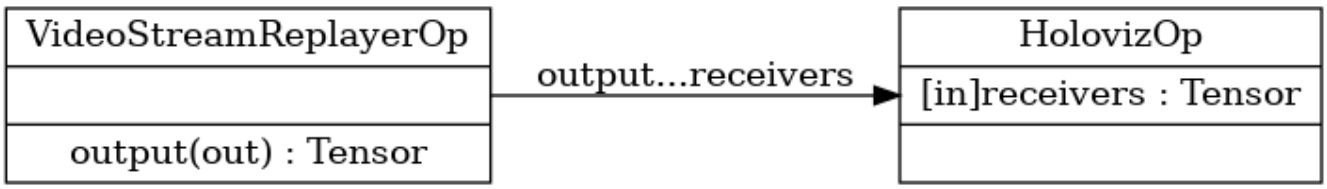


Fig. 9 *Workflow to load and display video from a file*

This is the same workflow as the single fragment video replayer, each operator is assigned to a separate fragment and there is now a network connection between the

fragments.

Defining and Connecting Fragments

Distributed applications define Fragments explicitly to isolate the different units of work that could be distributed to different nodes. In this example:

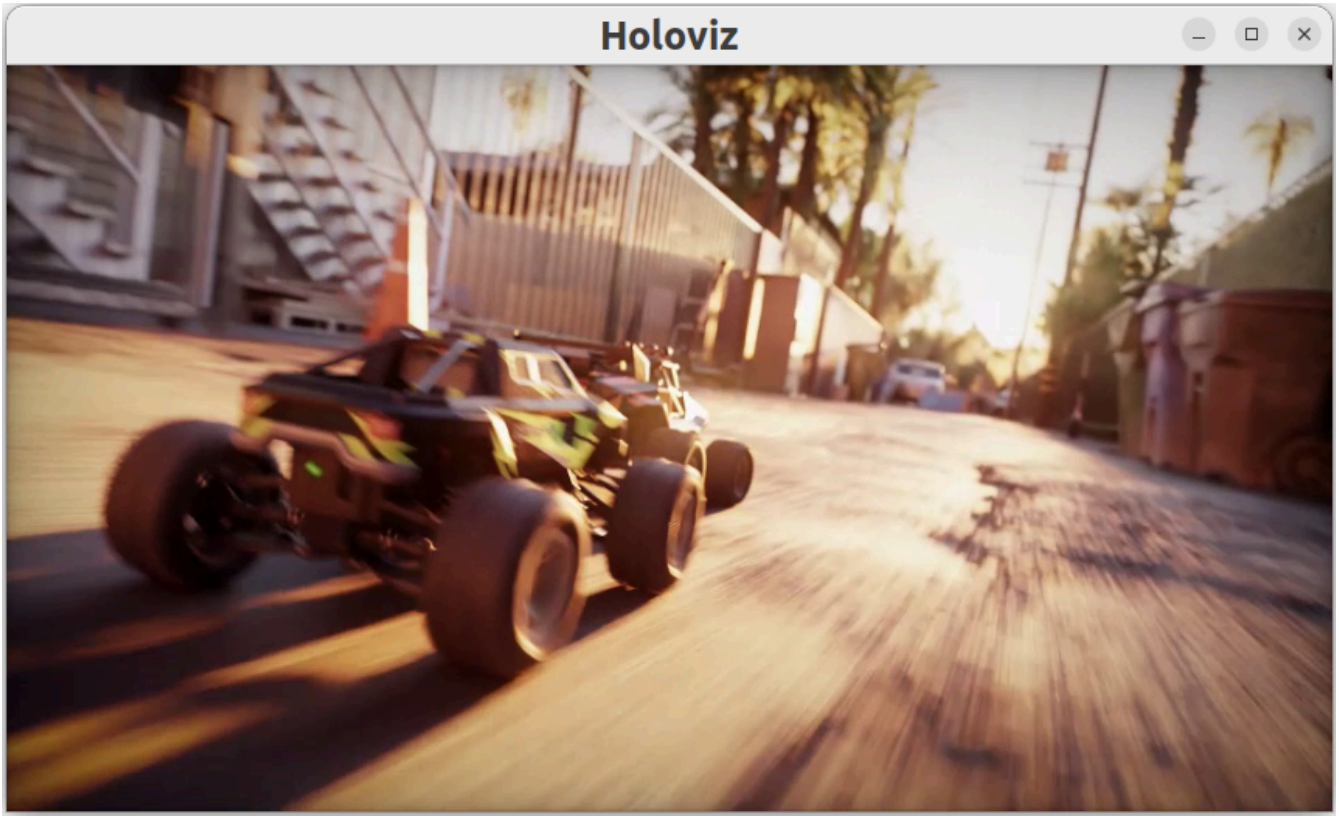
- We define two classes that inherit from `Fragment`:
    - **Fragment1** contains an instance of **VideoStreamReplayerOp** named “replayer”.
    - **Fragment2** contains an instance of **HolovizOp** name “holoviz”.
- We create an application, **DistributedVideoReplayerApp**. In its compose method:
    - we call **make_fragment** to initialize both fragments.
    - we then connect the “output” port of “replayer” operator in fragment1 to the “receivers” port of the “holoviz” operator in fragment2 to define the application workflow.
- The operators instantiated in the fragments can still be configured with parameters initialized from the YAML configuration ingested by the application using `from_config()` (C++) or `kwargs()` (Python).

Ingested Tab Module

This particular distributed application only has one operator per fragment, so the operators was added via **`add_operator`** **(** **`C++`** **/** **`Python`** **)**. In general, each fragment may have multiple operators and connections between operators within a fragment would be made using `add_flow()` (`C++` / `Python`) method within the fragment’s `compute()` (`C++` / `Python`) method.

Running the Application

Running the application should bring up video playback of the video referenced in the YAML file.



Note

Instructions for running the distributed application involve calling the application from the “driver” node as well as from any worker nodes. For details, see the application run instructions in the examples directory on GitHub, or under `/opt/nvidia/holoscan/examples/video_replayer_distributed` in the NGC container and the debian package.

Tip

Refer to UCX Network Interface Selection when running a distributed application across multiple nodes.