Dora advanced features

Haixuan Xavier Tao

tao.xavier@outlook.com

Table of content

YAML Dataflows

- Declarative Programming

Declarative programming is programming by **stating what the task or desired outcome is.**

- Descriptions of Nodes and Operators
- Descriptions of Inputs, Outputs
- Can pass on Environment variables

```yaml
nodes:
 - id: webcam
   operator:
     python: webcam.py
     inputs:
       tick: dora/timer/millis/50
     outputs:
       - image

 - id: object_detection
   operator:
     python: object_detection.py
     inputs:
       image: webcam/image
     outputs:
       - bbox

 - id: plot
   operator:
     python: plot.py
     inputs:
       image: webcam/image
       bbox: object_detection/bbox
      env:
         CUDA_VISIBLE_DEVICES:  ""
         PYTORCH_DEVICE:  cuda
```

Custom Nodes

Make it possible to use dora-rs anywhere with just function calls using Inter-Process Communication (IPC)

- .init_from_env()
- .recv()
- .send_output()

```rust
use dora_node_api::{ self, dora_core::config::DataId, DoraNode, Event};

fn main() -> eyre::Result<()> {
   let (mut node, mut events) = DoraNode::init_from_env()?;

   let output = DataId::from( "random".to_owned());
   for let Some(event) = events.recv() {
       match event {
           Event:: Input {
               id,
               metadata,
               data: _,
           } =>  match id.as_str() {
                "tick" => {
                    let random: u64 = rand::random();
                    node.send_output (output.clone(), metadata.parameters, random.into_arrow())?;
```

Operators

Feature rich dora-rs APIs that can benefit from being defined as a trait

- .Default()
- .on_event()
- output_sender.send()

```rust
use dora_operator_api::{
   register_operator , DoraOperator, DoraOutputSender, DoraStatus, Event, IntoArrow,
};

register_operator! (ExampleOperator);

#[derive(Debug, Default)]
struct ExampleOperator {
   ticks: usize,
}

impl DoraOperator  for ExampleOperator {
   fn on_event (
       &mut self,
       event: &Event,
       output_sender : &mut DoraOutputSender,
   ) -> Result<DoraStatus, String> {
       match event {
           Event:: Input  { id, data } =>  match *id {
               "random"  => {
                   let value = u64::try_from(data)
                       .map_err(|err|   format! ("unexpected data type:  {err}"))?;

                   let output =  format! (
                       "operator received random value  {value:#x} after  {} ticks",
                       self .ticks
                   );
                   output_sender .send("status" .into(), output.into_arrow())?;
```

Custom Nodes

Make it possible to use dora-rs anywhere with just function calls using Inter-Process Communication (IPC)

- .Node()
- .next()
- .send_output()

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import os
import time
import numpy as np
import cv2

from dora import Node

node = Node()

while True:
   # Wait next dora_input
   event = node.next()
   match event["type"]:
       case "INPUT":
           ret, frame = video_capture.read()

           node.send_output(
               "image",
               cv2.imencode( ".jpg", frame)[1].tobytes(),
               event[ "metadata" ],
           )
```

Operators

Feature rich dora-rs APIs that can benefit from being defined as a trait

- .___init___()
- .on_event(*)
- send_output()

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

from typing import Callable, Optional

import cv2
import numpy as np
import pyarrow as pa
import torch

from dora import DoraStatus

CAMERA_WIDTH = 640
CAMERA_HEIGHT = 480

class Operator:
    """
    Infering object from images
    """

    def __init__(self):
        self.model = torch.hub.load("ultralytics/yolov5", "yolov5n")

    def on_event(
        self,
        dora_event: dict,
        send_output: Callable[[str, bytes | pa.Array, Optional[dict]], None],
    ) -> DoraStatus:
        if dora_event["type"] == "INPUT":
            frame = dora_input["value"].to_numpy().reshape((CAMERA_HEIGHT, CAMERA_WIDTH, 3))
            frame = frame[:, :, ::-1]  # OpenCV image (BGR to RGB)
            results = self.model(frame)  # includes NMS
            arrays = pa.array(np.array(results.xyxy[0].cpu()).ravel())
            send_output("bbox", arrays, dora_input["metadata"])

        return DoraStatus.CONTINUE
```

Operators

Make it possible to have a feature-rich operators.

Language	**Rust Operator**	**Python Operator**
Native Object Types	Struct	Class
Method	Impl `DoraOperator trait`	on_event method
Distribution	Compiles into a shared library	Called by the dora-rs python runtime

Hoping to be able to do Intra Process Communication with Operators in the Future.

Check out dora-rs and dora-drives

https://github.com/dora-rs/dora-drives

https://dora.carsmos.ai/

Dora-drives (Live Demo)

- Introduction
- Getting Started
- Oasis

Node hub documentation

> https://dora.carsmos.ai/docs/nodes_operators/search

- Yolov5
- Obstacle location
- Carla GPS
- Obstacle location
- Frenet Optimal Trajectory Planning
- PID
- Plot

On Contribution

We welcome bug reports, feature requests, and pull requests!

Please discuss non-trivial changes in a Github issue or on Discord first before implementing them. This way, we can avoid unnecessary work on both sides.

The `dora` project is set up as a cargo workspace. You can use the standard `cargo check`, `cargo build`, `cargo run`, and `cargo test` commands. To run a command for a specific package only, pass e.g. `--package dora-daemon`. Running a command for the whole workspace is possible by passing `--workspace`.

Example contribution

Title	Description
Rust hot-reloading	Enabling to code rust operator in real time just like for Python
Arrow tensor Type	`arrow-rs`: extension type for Tensor Type enables multi-dimensional array to be better handle in Rust. See:apache/arrow-rs#4472
Rust-Python binding	It could be worth looking at `rye` as a way to automate the creation of virtual environment and making sure to ship dora operator in a way that is safe and sound. To do this it could be worth looking into making `rye` a library or else, but find a way to automate python package management. See: mitsuhiko/rye#113
Hardware acceleration	In Python, we don't really have a way to automatically target the current hardware at hand. It could be interesting to have a very broad reach compiler such as `iree` or `tvm` or else, and look if we can automatically, compile a DL model for the hardware at hand and provide GPU-acceleration
CLI	Currently the CLI is separated into 3 binaries. It could be good to merge them into a single binary.
CLI	CLI could value standardising its argument for `stop` and `logs` and use dynamic argument to choose dataflow. I think we can get inspiration for docker and reuse their syntax
Test bench harness	Currently we have a quick test harness to measure performance but it would be great to improve it and use new tooling such as `criterion`, `hyperfine`
C/C++ API	C/C++ API has not been our priority but could be a great first issue to learn about FFIs
Opentelemetry logs implementation	Opentelemetry recently stabilised the log feature enabling distributed logging. It could be great to use it. Currently we have a simple write to file implementation.
Opentelemetry metrics implementation	Last year we experimented with opentelemetry metrics and automatically collected data about, CPU, Memory, Disk usage. It could be interesting to see if more can be done in automated telemetry
Opentelemetry tracing otlp	Currently we use a jaeger specific provider for tracing but it could be interesting to change it for the generic otlp format
ROS/ROS 2 runtime bridge optimization	There might be optimization in the way data are being serialize/deserialize, especially for byte, that are currently handle one element at a time instead of being memory copied
dora-daemon	Heartbeat protocol for detecting crashed nodes/operators
Windows Testing/Support	Windows has couples of CI/CD issues has it does not share UNIX syntax. A quick fix could be to investigate the reason and fix it.
Improvement	Improve performance by redoing part of Python in dora-drives in Rust
Improve documentation in Chinese	Our current website can handle both english and chinese so feel free to add chinese
Remote nodes and distributed dataflow	

What can open-source bring to you

- Improve your Github Profile
- *Talk is cheap. Show me the code. - Linus Torvalds*
- Work on problems that others are facing too creating common interests.

We at dora-rs are committed to support you in making the change you want to do!

Q&A Session