dora-drives

Haixuan Xavier Tao

tao.xavier@outlook.com

Table of content

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

- #[derive(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())?;
```

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.

Joining our discord server

Invite Link: https://discord.gg/unqMNYVVBT

QR Code:

Check out dora-rs and dora-drives

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

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

https://dora.carsmos.ai/

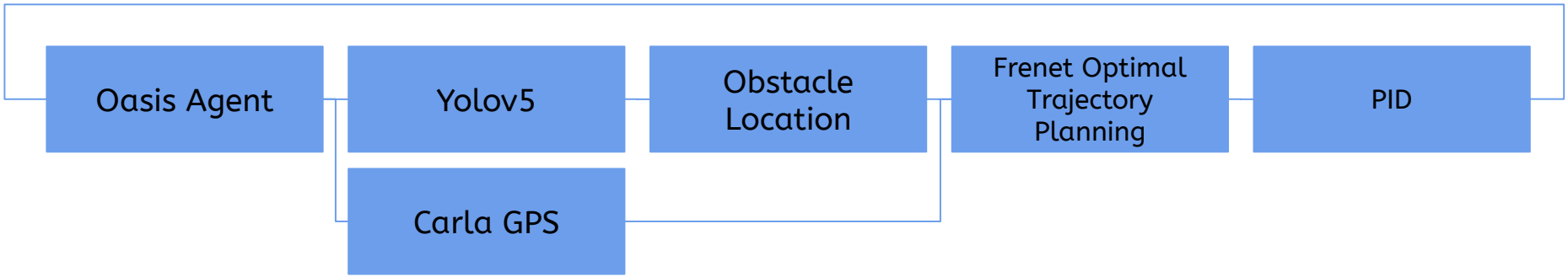
Oasis (live demo)

- Introduction
- Installation Guide
- Getting Started
- Running a scenario
- Q&A

Dora-drives (Live Demo)

- Introduction
- Getting Started
- Hot-reloading
- Q&A

Starter kit Graph



Node hub documentation

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

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

Development cheatsheet

- Use the volume binding within your installed folder

- Use `--hot-reload` feature

- Use tracing feature. Check out how to create dataset: https://github.com/Ashmita152/jaeger-datasets/pull/1

- Use logs feature

- Check out `dora-record` to create dataset from your runs

Q&A Session