mimoCoRB2

Julian Baader

Apr 16, 2025

CONTENTS

Add your content using `reStructuredText` syntax. See the reStructuredText documentation for details.

CHAPTER

ONE

GETTING STARTED

This section will guide you through the process of installing mimoCoRB2 and running the examples.

1.1 Introduction

1.1.1 What is mimoCoRB2?

mimoCoRB2 (multiple in multiple out configurable ringbuffer manager) provides a central component of each data acquisition system needed to record and preanalyse data from randomly occurring processes. Typical examples are waveform data as provided by detectors common in quantum mechanical measurements, or in nuclear, particle and astro particle physics, e. g. photo tubes, Geiger counters, avalanche photo-diodes or modern SiPMs. The random nature of such processes and the need to keep read-out dead times low requires an input buffer for fast collection of data and an efficient buffer manager delivering a constant data stream to the subsequent processing steps. While a data source feeds data into the buffer, consumer processes receive the data to filter, reduce, analyze or simply visualize the recorded data. In order to optimally use the available resources, multi-core and multi-processing techniques must be applied.

This project originated from an effort to structure and generalize data acquisition for several experiments in advanced physics laboratory courses at Karlsruhe Institute of Technology (KIT) and has been extensively tested with Ubuntu Linux.

1.1.2 What can it do?

Amongst the core features of mimoCoRB2 are:

- multiprocessing safe ringbuffer for NumPy structured arrays
- setup of multiple ringbuffers and workers from configuration files
- templates for common interactions between buffers (importing/exporting, filtering, processing, observing)
- pre built functions for common operations (oscilloscope, histogram, pulse height analysis)
- gui for monitoring and controlling the system

1.2 Examples

mimoCoRB2 comes with a set of examples that can be used to get started quickly. The examples are located in the *examples* directory of the mimoCoRB2 package alongside an explenation of the experiment. Each example is self-contained and can be run independently. The examples stem from experiments currently running at KIT and are meant to be used as a starting point for your own experiments.

CHAPTER

TWO

USER GUIDE

This user guide provides an in-depth description of mimoCoRB2 features required to build your own applications.

2.1 Setup

The setup file which is provided to the main script is a yaml file that defines the buffers and workers that will be used in the application. It describes the buffers and the data flow between them.

```yaml
Buffers:
    buffer_name_1:
        slot_count: int
        data_length: int
        data_dtype:
            field_name_1: field_dtype_1
            ...
    ...

Workers:
    worker_name_1:
        file: path_to_function_file
        function: function_name
        config: dict | str | [str]
        number_of_processes: int
        sources: [str]
        sinks: [str]
        observes: [str]
    ...
```

2.1.1 Slot Count

The slot count is the number of slots that the buffer will have. Each slot can hold a data packet in the form of a numpy structured array. It should be higher than the number of processes that will be using the buffer, as well as high enough to buffer a reasonable amount of data.

2.1.2 Data Length

The data is stored in the form of a numpy structured array of shape (data_lenght,). It is therefore the number of elements per field in the data.

2.1.3 Data Dtype

The data type of the data stored in the buffer. It is a dictionary where the keys are the field names and the values are the field data types. The field data types are the same as the numpy data types. For example, 'int32', 'float64', 'S10' (string of length 10), etc.

2.1.4 File

This is the path (relative to the setup file) to the file that contains the function that will be used in the worker. If the key is missing or empty, the prebuilt functions will be used.

2.1.5 Function

The name of the function that will be used in the worker. (TODO see prebuilt functions)

2.1.6 Config

The configuration of the worker. It can be a dictionary, a string or a list of strings. If it is a dictionary, it will be passed directly to the worker. If it is a string or a list of strings the yaml file at each path (relative to the setup file) will be loaded in the config dictionary (duplicate keys will be overwritten).

2.1.7 Number of Processes

The number of processes run by the worker.

2.1.8 Sources, Sinks, Observes

The sources, sinks and observes of the worker. They are the names of the buffers that will be used by the worker. Sources are the buffers that will be used to read data from, sinks are the buffers that will be used to write data to and observes are the buffers that will be used to observe data.

2.2 Simple Workers

Simple Workers can be built using the classes provided in the **:py:module:`mimocorb2.worker_templates`** module.

2.2.1 Importer

Importers are used to import data from an external source into the mimocorb2 system. They will automatically add the metadata to each event.

class mimocorb2.worker_templates.**Importer**(*io*)

Worker class for importing data from an external generator.

data_example

metadata_example

config

Examples

```
>>> def worker(*mimo_args):
...     importer = Importer(mimo_args)
...     config = importer.config
...     data_example = importer.writer.data_example
```

(continues on next page)

(continued from previous page)

```
...     buffer_name = importer.writer.name
...     def ufunc():
...         for i in range(config['n_events']):
...             data = np.random.normal(size=data_example.shape)
...             yield data
...         yield None
...     importer(ufunc)
```

__call__(*ufunc*)

Start the generator and write data to the buffer.

ufunc must yield data of the same format as the Importer.writer.data_example and yield None at the end. Metadata (counter, timestamp, deadtime) is automatically added to the buffer.

Parameters
ufunc (*Callable*) – Generator function that yields data and ends with None

Return type
None

__init__(*io*)

Checks the setup.

2.2.2 Exporter

Exporters are used to export data from the mimocorb2 system.

class mimocorb2.worker_templates.**Exporter**(*io*)

Worker class for exporting data and metadata.

If provided with an identical sink events will be copied to allow further analysis.

data_example

metadata_example

Examples

```
>>> def worker(*mimo_args):
...     exporter = Exporter(mimo_args)
...     data_example = exporter.reader.data_example
...     buffer_name = exporter.reader.name
...     for data, metadata in exporter:
...         print(data, metadata)
```

__init__(*io*)

Checks the setup.

2.2.3 Filter

Filters are used to filter data. This means that the data is not modified, but some of the data is removed.

class mimocorb2.worker_templates.**Filter**(*io*)

Worker class for filtering data from one buffer to other buffer(s).

Analyze data using ufunc(data) and copy or discard data based on the result.

data_example

metadata_example

config

Examples

```
>>> def worker(*mimo_args):
...     filter = Filter(mimo_args)
...     min_height = filter.config['min_height']
...     def ufunc(data):
...         if np.max(data) > min_height:
...             return True
...         else:
...             return False
...     filter(ufunc)
```

__call__(*ufunc*)

Start the filter and copy or discard data based on the result of ufunc(data).

Parameters

ufunc (*Callable*) – Function which will be called upon the data (Filter.reader.data_example). The function can return:

bool
True: copy data to every sink False: discard data

list[bool] (mapping to the sinks)
True: copy data to the corresponding sink False: dont copy data to the corresponding sink

Return type
None

__init__(*io*)

Checks the setup.

Check that the number of sources, sinks, and observes are correct. Check that the source and sink shapes and dtypes match.

2.2.4 Processor

Processors are used to process data. This means that the data is modified in some way.

class mimocorb2.worker_templates.**Processor**(*io*)

Worker class for processing data from one buffer to other buffer(s).

reader

BufferReader object for reading data from the buffer

Type
BufferReader

writers

List of BufferWriter objects for writing data to the buffers

Type
list[BufferWriter]

Examples

```
>>> def worker(*mimo_args):
...     processor = Processor(mimo_args)
...     def ufunc(data):
...         return [data + 1, data - 1]
...     processor(ufunc)
```

__call__(*ufunc*)

Start the processor and process data using ufunc(data).

Parameters
ufunc (*Callable*) – Function which will be called upon the data (Processor.reader.data_example). When the function returns None the data will be discarded. Otherwise the function must return a list of results, one for each sink. If the result is not None it will be written to the corresponding sink.

Return type
None

__init__(*io*)

Checks the setup.

Parameters
mimo_args (*ArgsAlias*) – List of sources, sinks, observes, and config dictionary

2.2.5 Observer

Observers are used to observe data. This means that a copy of the data is exported.

class mimocorb2.worker_templates.**Observer**(*io*)

Worker class for observing data from a buffer.

observer

BufferObserver object for observing data from the buffer

Type
BufferObserver

Examples

```
>>> def worker(*mimo_args):
...     observer = Observer(mimo_args)
...     generator = observer()
...     while True:
...         data, metadata = next(generator)
...         if data is None:
...             break
...         print(data, metadata)
...         time.sleep(1)
```

__call__()

Start the observer and yield data and metadata.

Yields data and metadata from the buffer until the buffer is shutdown.

Yields

- **data** (*np.ndarray, None*) – Data from the buffer
- **metadata** (*np.ndarray, None*) – Metadata from the buffer

Return type
`Generator`

__init__(*io*)

Checks the setup.

Parameters
mimo_args (*ArgsAlias*) – List of sources, sinks, observes, and config dictionary

INDEX

Symbols

C

D

E

F

I

M

O

P

R

W