
mimoCoRB2

Julian Baader

Mar 04, 2025

CONTENTS

1	Getting Started	3
2	User Guide	5
	Index	13

Add your content using reStructuredText syntax. See the [reStructuredText](#) documentation for details.

GETTING STARTED

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

USER GUIDE

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

2.1 Buffers

Buffers are at the core of mimoCoRB2. Each buffer is defined by

- name
- slot_count
- data_length
- data_dtype

Each buffer is divided into slots, where each slot holds data and metadata. Data is a numpy structured array of shape (data_length,) and dtype data_dtype. Metadata is a numpy structured array of shape (1,) and dtype metadata_dtype.

```
mimoBuffer.metadata_dtype = dtype([('counter', '<i8'), ('timestamp', '<f8'), ('deadtime', '<f8')])
```

Buffers are implemented in a multiprocessing safe way using shared memory. This means multiple processes can read and write to the same buffer at the same time without any issues.

2.2 Workers

Workers are used for interactions between buffers.

A worker is defined by

- name
- function
- args
- number_of_processes

The name is used as an identifier for the worker. The function is the function that the worker will run with the provided args. The number_of_processes is the number of processes that the worker will run.

2.3 Setup

A setup file is used to define the buffers and workers of the application. It is written in yaml format and divided into three sections.

2.3.1 Buffers

buffer_name:

slot_count: int data_length: int data_dtype:

field_name1: dtype1 field_name2: dtype2 ...

2.3.2 Workers

worker_name:

function: str file: str number_of_processes: int sinks: list[str] sources: list[str] observes: list[str]

2.4 Simple Workers

2.4.1 Importer

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
...     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.

2.4.2 Exporter

`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.

2.4.3 Filter

`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.

2.4.4 Processor

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

2.4.5 Observer

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

2.5 Complex Workers

In case the simple workers are not enough, you can create highly customized workers by using the context managers.

2.5.1 Buffer Reader

```
class mimocorb2.mimo_buffer.BufferReader(buffer)
    A context manager for reading data from a mimoBuffer.

    __enter__()
        Get a token and access the slot for reading.

    __exit__(exc_type, exc_value, traceback)
        Return the token after reading.

class mimocorb2.mimo_buffer.BufferWriter(buffer)
    A context manager for writing data to a mimoBuffer.

    __enter__()
        Get a token and access the slot for writing.

    __exit__(exc_type, exc_value, traceback)
        Return the token after writing.

    send_flush_event()
        Send a flush event to notify consumers.

class mimocorb2.mimo_buffer.BufferObserver(buffer)
    A context manager for observing data in a mimoBuffer.

    __enter__()
        Get a token and access the slot for observation.

    __exit__(exc_type, exc_value, traceback)
        Return the token after observation.
```

2.5.2 IO Object

```
class mimocorb2.mimo_worker.BufferIO(sources, sinks, observes, config)
    Collection of buffers for input/output operations.

    Object which is passed to each worker process to provide access to the buffers.

    sources
        Type
        list[BufferReader]

    sinks
        Type
        list[BufferWriter]

    observes
        Type
        list[BufferObserver]

    config
        Type
        dict
```

logger**Type**

logging.Logger

shutdown_sinks()

Shutdown all sink buffers.

__getitem__(*key*)

Get the value of a key from the configuration dictionary.

Examples

```
>>> with io.write[0] as (metadata, data):
```


Symbols

`__call__()` (*mimocorb2.worker_templates.Filter* method), 7
`__call__()` (*mimocorb2.worker_templates.Importer* method), 6
`__call__()` (*mimocorb2.worker_templates.Observer* method), 9
`__call__()` (*mimocorb2.worker_templates.Processor* method), 8
`__enter__()` (*mimocorb2.mimo_buffer.BufferObserver* method), 10
`__enter__()` (*mimocorb2.mimo_buffer.BufferReader* method), 10
`__enter__()` (*mimocorb2.mimo_buffer.BufferWriter* method), 10
`__exit__()` (*mimocorb2.mimo_buffer.BufferObserver* method), 10
`__exit__()` (*mimocorb2.mimo_buffer.BufferReader* method), 10
`__exit__()` (*mimocorb2.mimo_buffer.BufferWriter* method), 10
`__getitem__()` (*mimocorb2.mimo_worker.BufferIO* method), 11
`__init__()` (*mimocorb2.worker_templates.Exporter* method), 7
`__init__()` (*mimocorb2.worker_templates.Filter* method), 8
`__init__()` (*mimocorb2.worker_templates.Importer* method), 6
`__init__()` (*mimocorb2.worker_templates.Observer* method), 9
`__init__()` (*mimocorb2.worker_templates.Processor* method), 8

B

`BufferIO` (class in *mimocorb2.mimo_worker*), 10
`BufferObserver` (class in *mimocorb2.mimo_buffer*), 10
`BufferReader` (class in *mimocorb2.mimo_buffer*), 10
`BufferWriter` (class in *mimocorb2.mimo_buffer*), 10

C

`config` (*mimocorb2.mimo_worker.BufferIO* attribute),

10

`config` (*mimocorb2.worker_templates.Filter* attribute), 7
`config` (*mimocorb2.worker_templates.Importer* attribute), 6

D

`data_example` (*mimocorb2.worker_templates.Exporter* attribute), 7
`data_example` (*mimocorb2.worker_templates.Filter* attribute), 7
`data_example` (*mimocorb2.worker_templates.Importer* attribute), 6

E

`Exporter` (class in *mimocorb2.worker_templates*), 7

F

`Filter` (class in *mimocorb2.worker_templates*), 7

I

`Importer` (class in *mimocorb2.worker_templates*), 6

L

`logger` (*mimocorb2.mimo_worker.BufferIO* attribute), 10

M

`metadata_dtype` (*mimocorb2.mimo_buffer.mimoBuffer* attribute), 5
`metadata_example` (*mimocorb2.worker_templates.Exporter* attribute), 7
`metadata_example` (*mimocorb2.worker_templates.Filter* attribute), 7
`metadata_example` (*mimocorb2.worker_templates.Importer* attribute), 6

O

`Observer` (class in *mimocorb2.worker_templates*), 9

`observer` (*mimocorb2.worker_templates.Observer* attribute), [9](#)

`observes` (*mimocorb2.mimo_worker.BufferIO* attribute), [10](#)

P

`Processor` (*class in mimocorb2.worker_templates*), [8](#)

R

`reader` (*mimocorb2.worker_templates.Processor* attribute), [8](#)

S

`send_flush_event()` (*mimocorb2.mimo_buffer.BufferWriter* method), [10](#)

`shutdown_sinks()` (*mimocorb2.mimo_worker.BufferIO* method), [11](#)

`sinks` (*mimocorb2.mimo_worker.BufferIO* attribute), [10](#)

`sources` (*mimocorb2.mimo_worker.BufferIO* attribute), [10](#)

W

`writers` (*mimocorb2.worker_templates.Processor* attribute), [8](#)