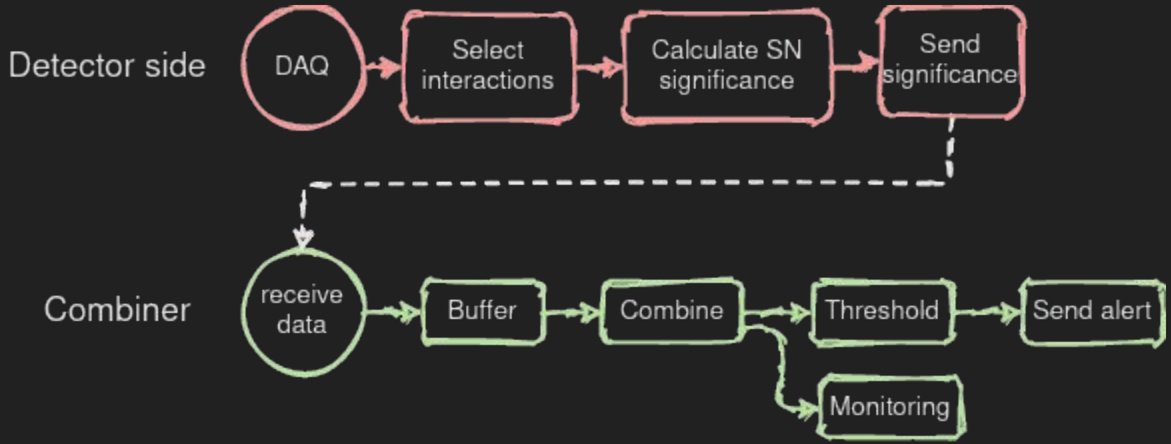
SuperNova Asynchronous Pipeline

Andrey Sheshukov
DLNP JINR

SNEWS2021 collaboration meeting
Implementation

12 May 2021

Example of analysis flow for the SN trigger in NOvA



- Data flows in one direction in chains: from source through several steps
- Steps in one chain are not synchronous: input and output of each step can be different and not directly synchronized.
- Chains can have branching
- Computationally heavy steps need to be parallelized
- Each step/source can have many configuration parameters
- Order of the steps is also a configuration

Usually this requires setting up threads, produces/consumers, worker pools...

SuperNova Async Pipeline (SNAP)

A python framework for constructing realtime data processing pipelines.

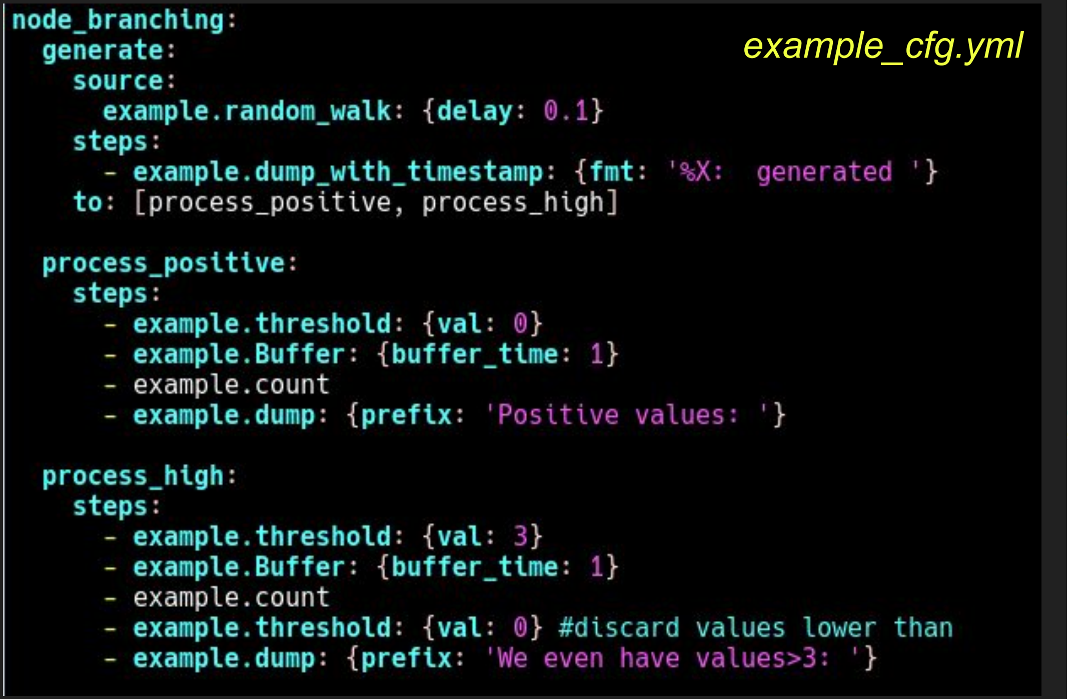
SNAP allows to easily chain the python functions/coroutines/classes.
Features:

- Allows to describe processing sources/steps
- Based on python asyncio - no need for threading
- Automatic separating event loops, where needed (i.e. buffering)
- All the configuration is separated from the code, kept in a YAML format.
- Extendable plugin system
- Can be used to construct microservices

Works for many tasks: combination, filtering, monitoring, sending alarm, visualization etc.

- Framework core: https://github.com/Sheshuk/snap-base
- SN combination: https://github.com/Sheshuk/snap-combine

Usage example



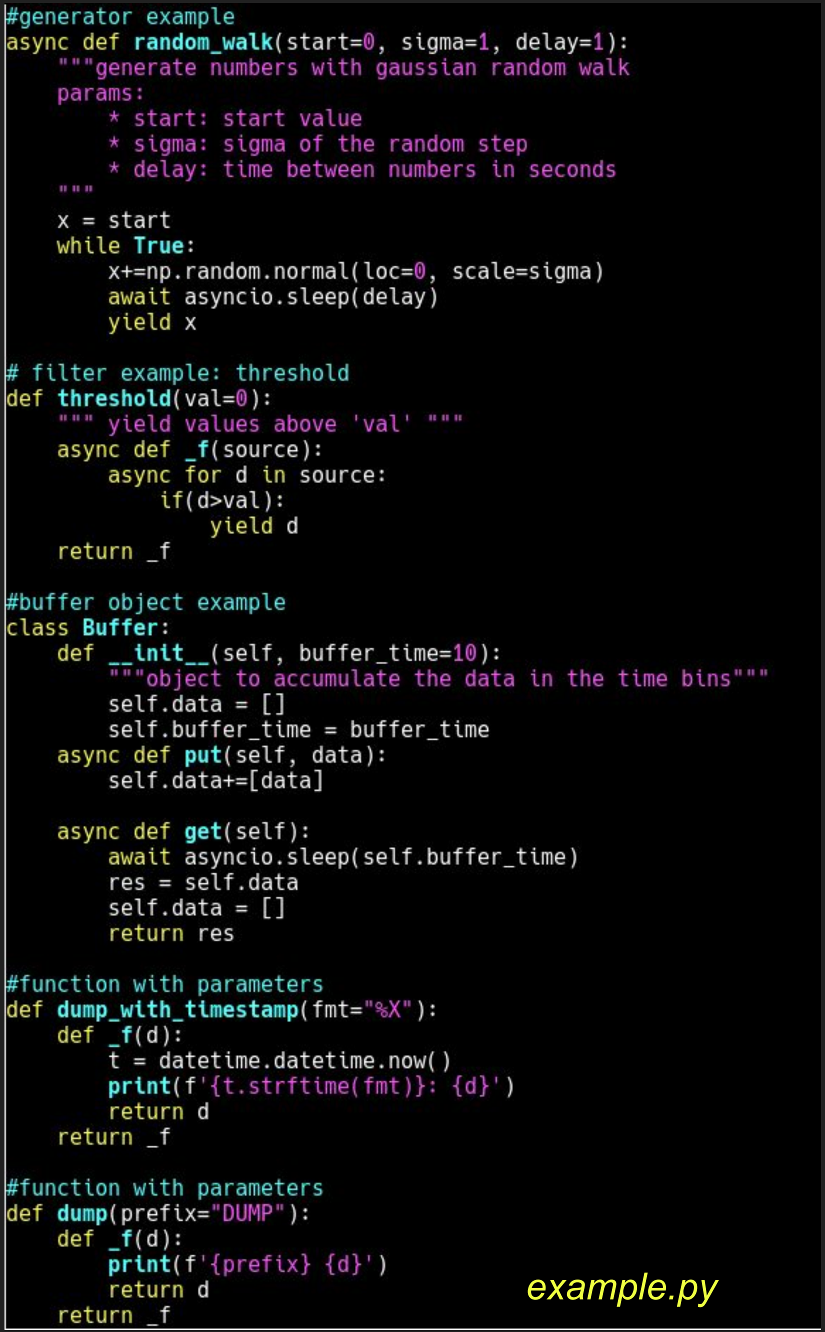
example_cfg.yml

```yaml
node_branching:
  generate:
    source:
      example.random_walk: {delay: 0.1}
    steps:
      - example.dump_with_timestamp: {fmt: '%X:  generated '}
    to: [process_positive, process_high]

  process_positive:
    steps:
      - example.threshold: {val: 0}
      - example.Buffer: {buffer_time: 1}
      - example.count
      - example.dump: {prefix: 'Positive values: '}

  process_high:
    steps:
      - example.threshold: {val: 3}
      - example.Buffer: {buffer_time: 1}
      - example.count
      - example.threshold: {val: 0} #discard values lower than
      - example.dump: {prefix: 'We even have values>3: '}
```

snap example_cfg.yml -n node_branching

Generate random numbers (via random walk)
Separate processing chains for x>0 and x>3

Count number of positive values per second

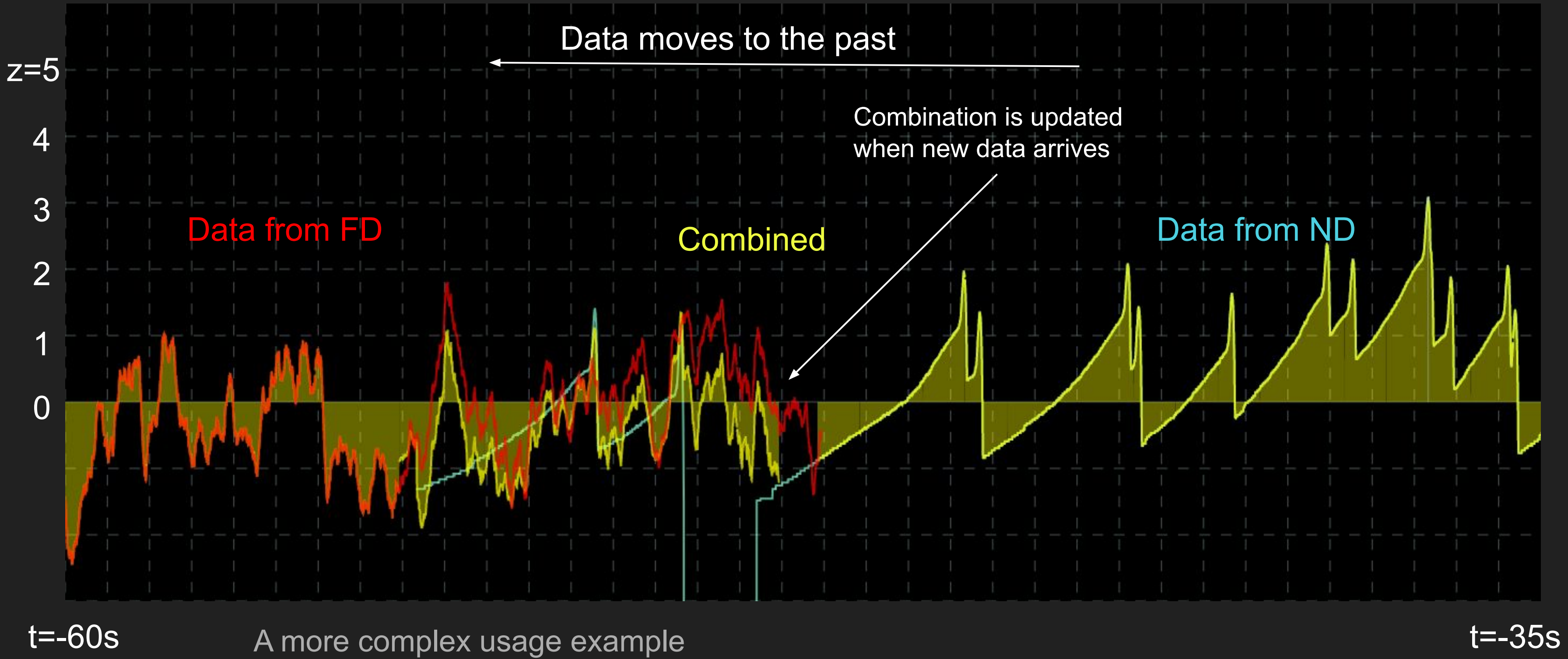


```python
#generator example
async def random_walk(start=0, sigma=1, delay=1):
    """generate numbers with gaussian random walk
    params:
        * start: start value
        * sigma: sigma of the random step
        * delay: time between numbers in seconds
    """
    x = start
    while True:
        x+=np.random.normal(loc=0, scale=sigma)
        await asyncio.sleep(delay)
        yield x

# filter example: threshold
def threshold(val=0):
    """ yield values above 'val' """
    async def _f(source):
        async for d in source:
            if(d>val):
                yield d
    return _f

#buffer object example
class Buffer:
    def __init__(self, buffer_time=10):
        """object to accumulate the data in the time bins"""
        self.data = []
        self.buffer_time = buffer_time
    async def put(self, data):
        self.data+=[data]

    async def get(self):
        await asyncio.sleep(self.buffer_time)
        res = self.data
        self.data = []
        return res

#function with parameters
def dump_with_timestamp(fmt="%X"):
    def _f(d):
        t = datetime.datetime.now()
        print(f'{t.strftime(fmt)}: {d}')
        return d
    return _f

#function with parameters
def dump(prefix="DUMP"):
    def _f(d):
        print(f'{prefix} {d}')
        return d
    return _f
```

example.py

Monitoring example:
Real-time web visualization for the SN significance

One of the modules acts as the WebSockets server, sending all the data to clients’s browser.



A more complex usage example