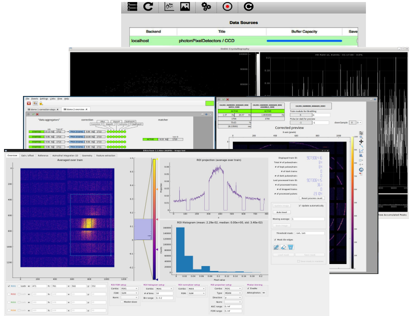
Online data analysis with Metro
Hands-on Edition

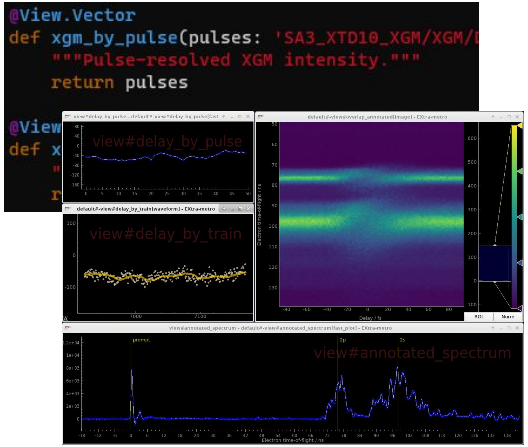
Philipp Schmidt
DA

Hamburg

Spectrum of online analysis solutions



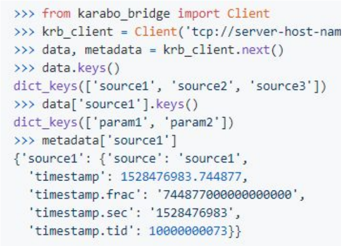
Karabo devices, EXtra-foam and others



```
@View.Vector
def xgm_by_pulse(pulses: 'SA3_XTD10_XGM/XGM/
    """Pulse-resolved XGM intensity."""
    return pulses

@View
def x
```

view#delay_by_pulse - default#-view#delay_by_pulse[fast
view#delay_by_pulse
default#-view#delay_by_train[waveform] - EXtra-metro
view#delay_by_train
default#-view#overlap_annotated[image] - EXtra-metro
Electron time-of-flight / ns
Delay / fs
ROI
Norm
view#annotated_spectrum - default#-view#annotated_spectrum[fast_plot] - EXtra-metro
view#annotated_spectrum
prompt
2p
2s
Electron time-of-flight / ns

EXtra-metro

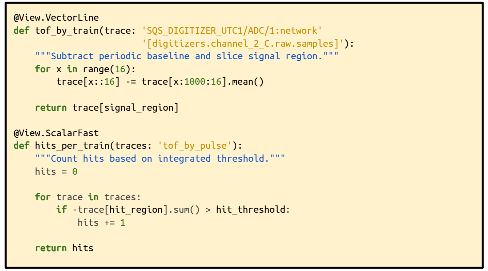


```
>>> from karabo_bridge import Client
>>> krb_client = Client('tcp://server-host-nam
>>> data, metadata = krb_client.next()
>>> data.keys()
dict_keys(['source1', 'source2', 'source3'])
>>> data['source1'].keys()
dict_keys(['param1', 'param2'])
>>> metadata['source1']
{'source1': {'source': 'source1',
  'timestamp': 1528476983.744877,
  'timestamp.frac': '744877000000000000',
  'timestamp.sec': '1528476983',
  'timestamp.tid': 10000000073}}
```

karabo-bridge

accessible ⟵⟶ **flexible**

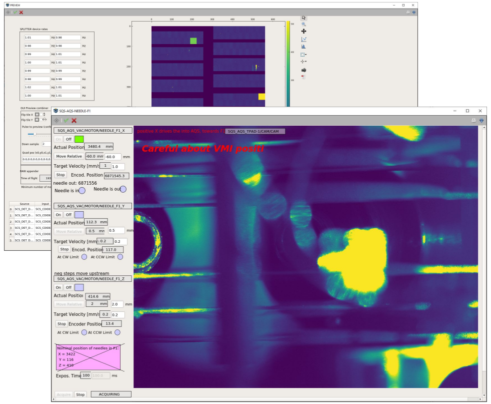
Spectrum of online analysis solutions



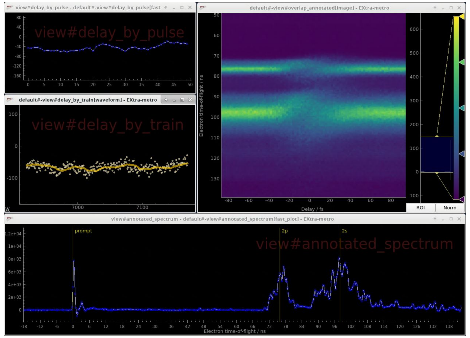
```
@View.VectorLine
def tof_by_train(trace: 'SQS_DIGITIZER_UTC1/ADC/1:network'
                        '[digitizers.channel_2_C.raw.samples]'):
    """Subtract periodic baseline and slice signal region."""
    for x in range(16):
        trace[x::16] -= trace[x:1000:16].mean()

    return trace[signal_region]

@View.ScalarFast
def hits_per_train(traces: 'tof_by_pulse'):
    """Count hits based on integrated threshold."""
    hits = 0

    for trace in traces:
        if -trace[hit_region].sum() > hit_threshold:
            hits += 1

    return hits
```

Your own analysis code on that data concentrating on the math & science

Any control or pipeline data from within Karabo

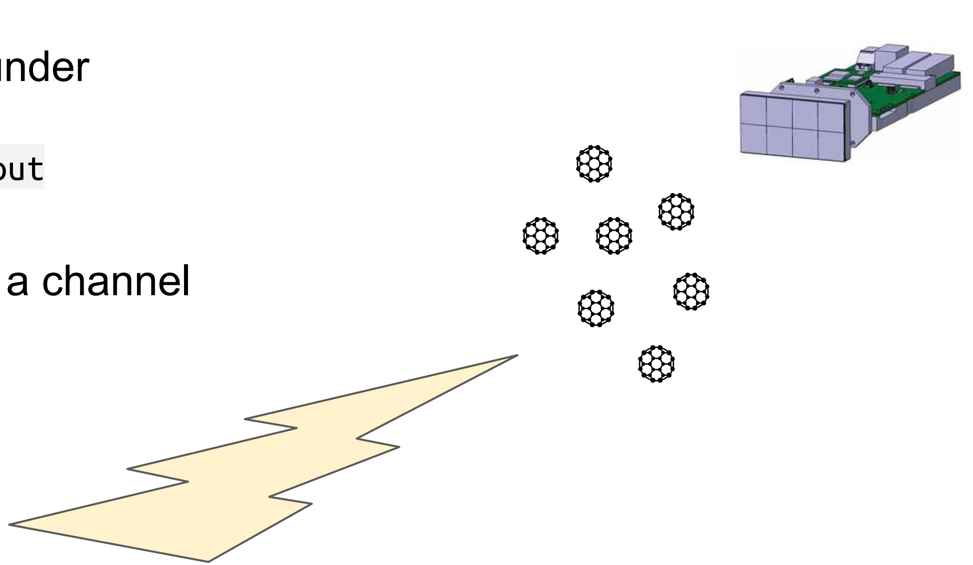


Automatic real-time plotting of results



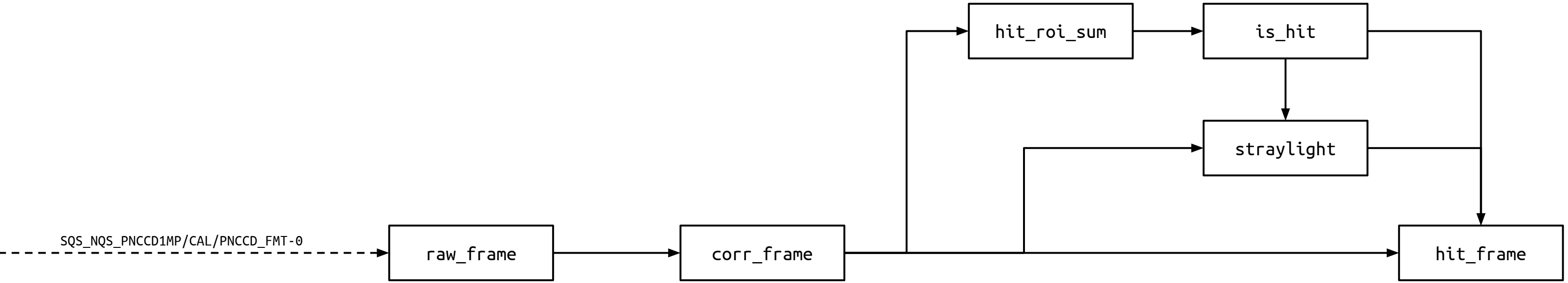
Hands-on

- Let’s assume a simple scattering experiment with a single detector
- In files, frame data would be found under the key `data.image` of source `SQS_NQS_PNCCD1MP/CAL/PNCCD_FMT-0:output`
- In Karabo live data, the device `SQS_NQS_PNCCD1MP/CAL/PNCCD_FMT-0` has a channel `output` with key `data.image`

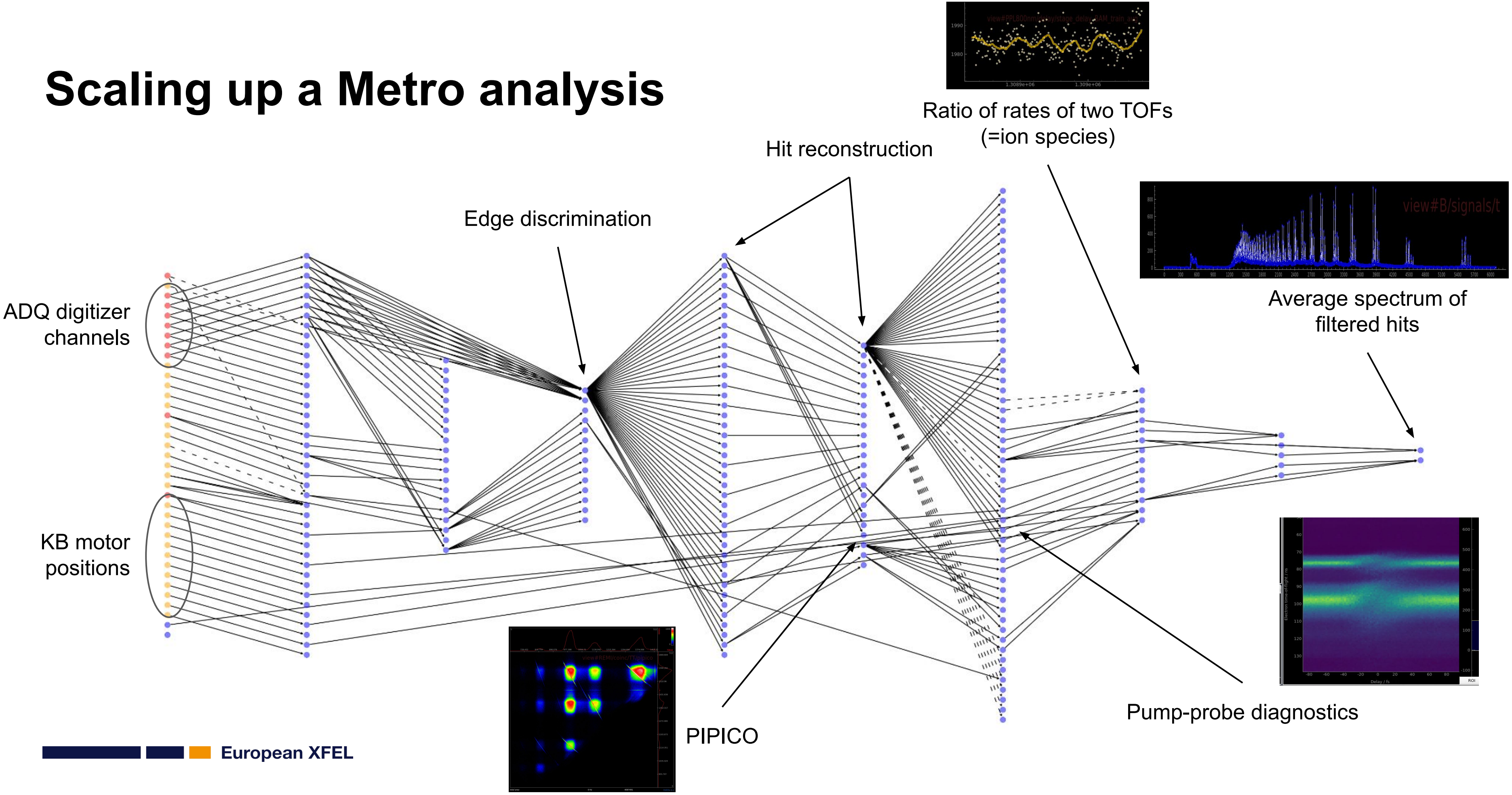


Hands-on

Analysis graph from hands-on demo



Scaling up a Metro analysis



Ratio of rates of two TOFs (=ion species)

Hit reconstruction

Edge discrimination

Average spectrum of filtered hits

ADQ digitizer channels

KB motor positions

PIPICO

Pump-probe diagnostics

More context code features

View prototypes and groups

```
@ViewPrototype.Vector
def corr_trace(trace: 'SQS_DIGITIZER_UTC1/ADC/1:network'
                      '[digitizers.channel_{channel}.raw.samples]'):
    # Do your analysis.

# Instantiate prototype multiple times for different channels.
corr_trace(name='raw/1A', channel='1_A')
corr_trace(name='raw/1C', channel='1_C')


class DigitizerChannel(ViewGroup):
    @View.Vector(name='{prefix}corr_trace')
    def corr_trace(trace: 'SQS_DIGITIZER_UTC1/ADC/1:network'
                          '[digitizers.channel_{channel}.raw.samples]'):
        # Do important analysis

    @View.Scalar(name='{prefix}foo)
    def foo(trace: '{prefix}corr_trace'):
        # Do even more important analysis

# Instantiate multiple views at once.
DigitizerChannel(prefix='raw1A/', channel='1_A')
```

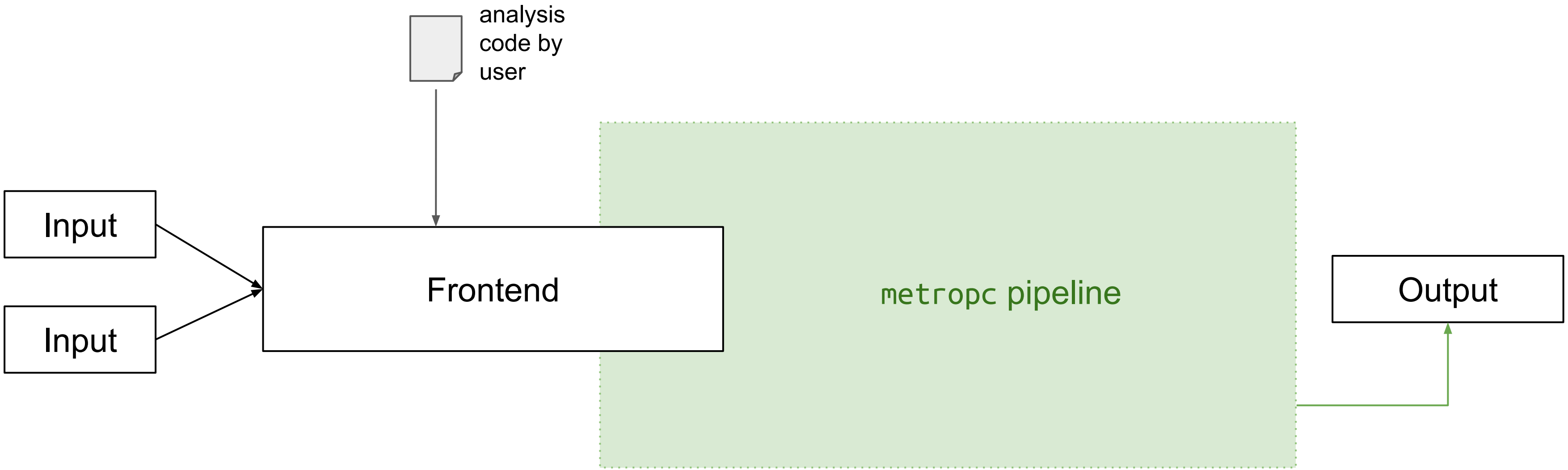
Action views and result feedback

```
@View.Vector_Action(feedback=True)
def reference(trace: 'SQS_DIGITIZER_UTC1/ADC/1:network'
                     '[digitizers.channel_1_A.raw.samples]'):
    # Result sticks around until triggered again.
    return trace
```

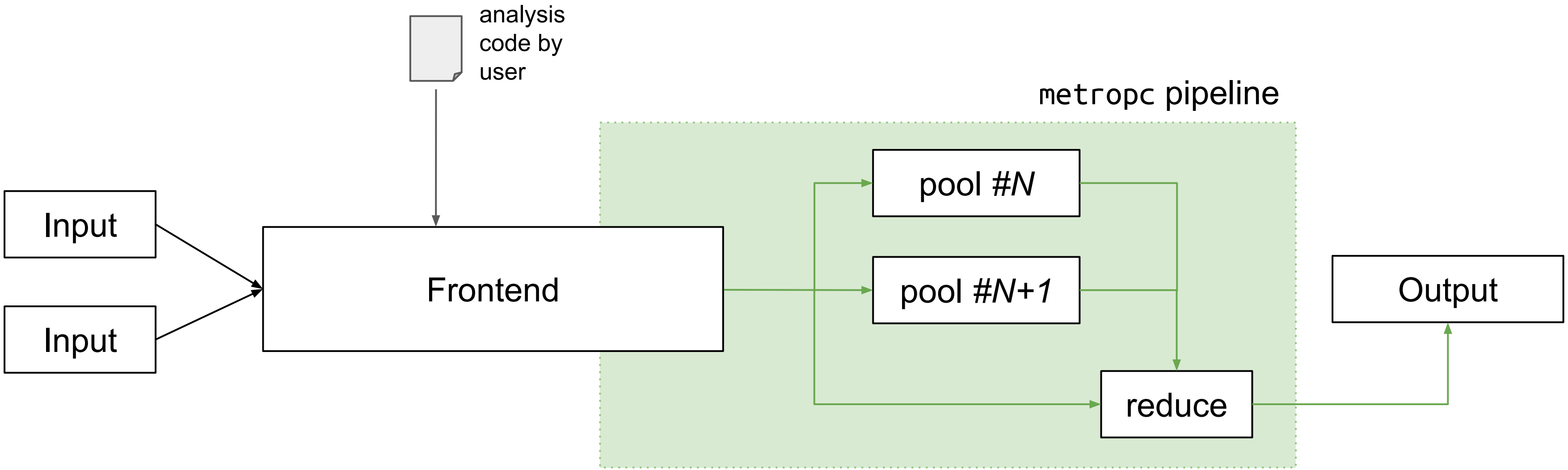
Wildcard paths and result annotations

```
@View.Vector
def all_(traces: 'karabo#[by-name]SQS_DIGITIZER_UTC1/ADC/1:network'
                 '[digitizers.channel_*.raw.samples]'):
    stacked = np.stack(traces.values())
    return xr.DataArray(
        stacked,
        dims=['channel', 'time / ns'],
        coords={'channel': traces.keys(),
                'time / ns': 0.5 * np.arange(stacked.shape[1])},
        attrs={'ylabel': 'ADU', 'vlines': [1000, 2000, 3000]}
    )
```

Pipeline layout



Pipeline layout



MetroProcessor device

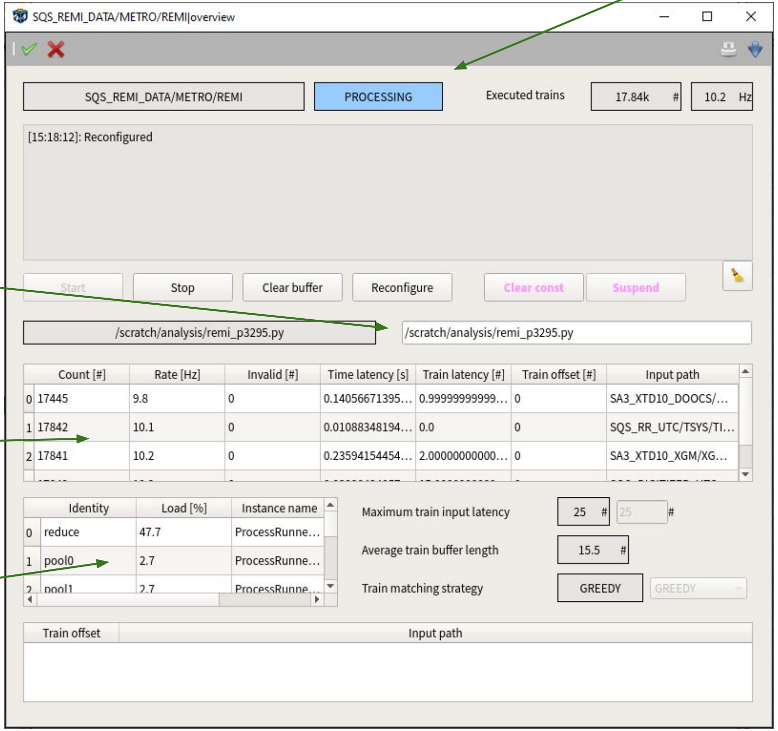
INIT: While reconfiguring
PASSIVE: Suspended
ACTIVE: Context loaded
PROCESSING: Running context

ERROR: Recoverable problem
DISABLED: Unrecoverable problem

Context code is loaded locally

Connected pipeline sources

Parts making up the pipeline



SQS_REMI_DATA/METRO/REMI | PROCESSING | Executed trains 17.84k # | 10.2 Hz

[15:18:12]: Reconfigured

/scratch/analysis/remi_p3295.py | /scratch/analysis/remi_p3295.py

	Count [#]	Rate [Hz]	Invalid [#]	Time latency [s]	Train latency [#]	Train offset [#]	Input path
0	17445	9.8	0	0.14056671395…	0.99999999999…	0	SA3_XTD10_DOOCS/…
1	17842	10.1	0	0.01088348194…	0.0	0	SQS_RR_UTC/TSYS/TI…
2	17841	10.2	0	0.23594154454…	2.00000000000…	0	SA3_XTD10_XGM/XG…

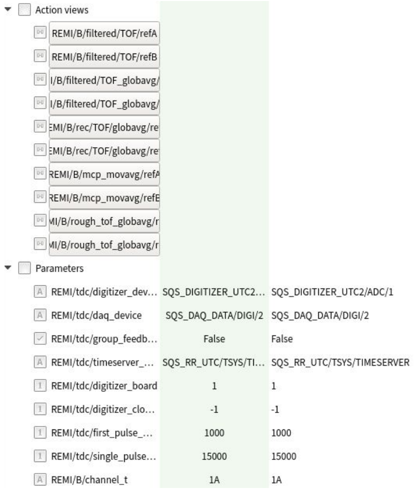
	Identity	Load [%]	Instance name
0	reduce	47.7	ProcessRunne…
1	pool0	2.7	ProcessRunne…
2	pool1	2.7	ProcessRunne…

Maximum train input latency 25 # | 25 #

Average train buffer length 15.5 #

Train matching strategy GREEDY | GREEDY

Train offset	Input path



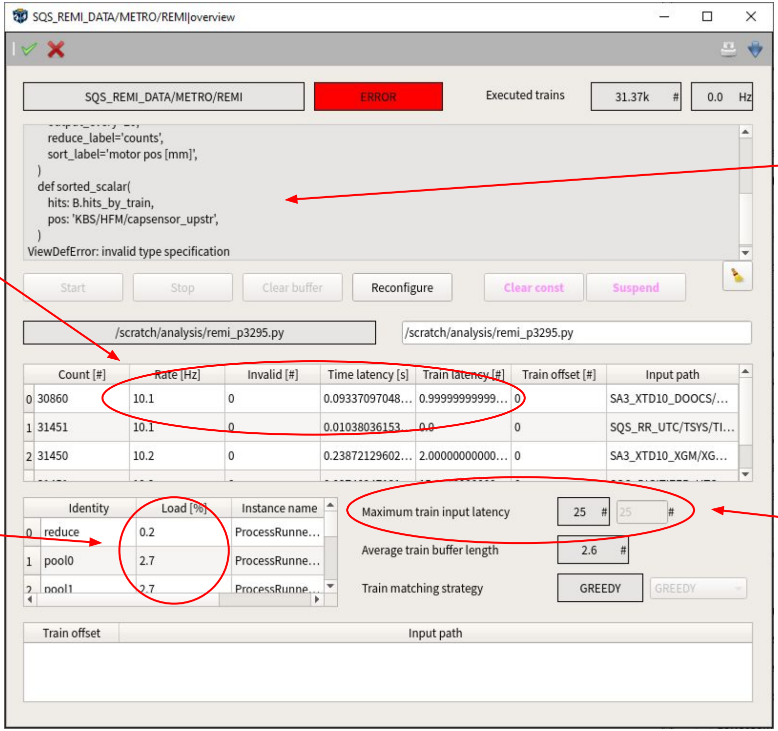
Action views

REMI/B/filtered/TOF/refA
REMI/B/filtered/TOF/refB
I/B/filtered/TOF_globavg/
I/B/filtered/TOF_globavg/
EMI/B/rec/TOF/globavg/re
EMI/B/rec/TOF/globavg/re
REMI/B/mcp_movavg/refA
REMI/B/mcp_movavg/refE
MI/B/rough_tof_globavg/r
MI/B/rough_tof_globavg/r

Parameters

REMI/tdc/digitizer_dev…	SQS_DIGITIZER_UTC2…	SQS_DIGITIZER_UTC2/ADC/1
REMI/tdc/daq_device	SQS_DAQ_DATA/DIGI/2	SQS_DAQ_DATA/DIGI/2
REMI/tdc/group_feedb…	False	False
REMI/tdc/timeserver_…	SQS_RR_UTC/TSYS/TI…	SQS_RR_UTC/TSYS/TIMESERVER
REMI/tdc/digitizer_board	1	1
REMI/tdc/digitizer_clo…	-1	-1
REMI/tdc/first_pulse_…	1000	1000
REMI/tdc/single_pulse…	15000	15000
REMI/B/channel_t	1A	1A

MetroProcessor device



SQS_REMI_DATA/METRO/REMI|overview

SQS_REMI_DATA/METRO/REMI — ERROR — Executed trains 31.37k # 0.0 Hz

```
    reduce_label='counts',
    sort_label='motor pos [mm]',
  )
  def sorted_scalar(
    hits: B.hits_by_train,
    pos: 'KBS/HFM/capsensor_upstr',
  )
ViewDefError: invalid type specification
```

/scratch/analysis/remi_p3295.py — /scratch/analysis/remi_p3295.py

	Count [#]	Rate [Hz]	Invalid [#]	Time latency [s]	Train latency [#]	Train offset [#]	Input path
0	30860	10.1	0	0.09337097048…	0.99999999999…	0	SA3_XTD10_DOOCS/…
1	31451	10.1	0	0.01038036153…	0.0	0	SQS_RR_UTC/TSYS/TI…
2	31450	10.2	0	0.23872129602…	2.00000000000…	0	SA3_XTD10_XGM/XG…

	Identity	Load [%]	Instance name
0	reduce	0.2	ProcessRunne…
1	pool0	2.7	ProcessRunne…
2	pool1	2.7	ProcessRunne…

Maximum train input latency 25 # 25 #

Average train buffer length 2.6 #

Train matching strategy GREEDY GREEDY

Train offset	Input path

Error usually found in context code and **always** with status update

Input not sending data, invalid or late data

Overloaded pipeline stages due to analysis code, in particular the synchronous reduction stage

Latency of some input source exceeds maximum

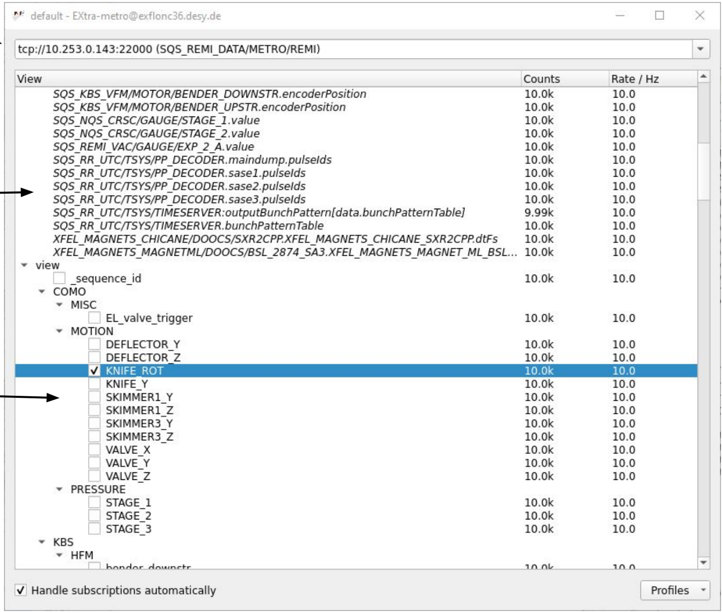
EXtra-metro client

```
you@exfloncNN:~% module load exfel EXtra-metro
you@exfloncNN:~% extra-metro
```

Pipeline output address to connect to

Rates for data input

Rates for view outputs

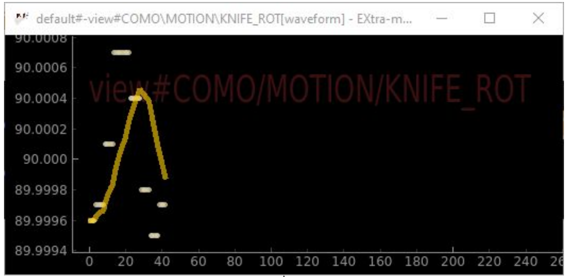


default - EXtra-metro@exflonc36.desy.de

tcp://10.253.0.143:22000 (SQS_REMI_DATA/METRO/REMI)

View	Counts	Rate / Hz
SQS_KBS_VFM/MOTOR/BENDER_DOWNSTR.encoderPosition	10.0k	10.0
SQS_KBS_VFM/MOTOR/BENDER_UPSTR.encoderPosition	10.0k	10.0
SQS_NQS_CRSC/GAUGE/STAGE_1.value	10.0k	10.0
SQS_NQS_CRSC/GAUGE/STAGE_2.value	10.0k	10.0
SQS_REMI_VAC/GAUGE/EXP_2_A.value	10.0k	10.0
SQS_RR_UTC/TSYS/PP_DECODER.maindump.pulseIds	10.0k	10.0
SQS_RR_UTC/TSYS/PP_DECODER.sase1.pulseIds	10.0k	10.0
SQS_RR_UTC/TSYS/PP_DECODER.sase2.pulseIds	10.0k	10.0
SQS_RR_UTC/TSYS/PP_DECODER.sase3.pulseIds	10.0k	10.0
SQS_RR_UTC/TSYS/TIMESERVER:outputBunchPattern[data.bunchPatternTable]	9.99k	10.0
SQS_RR_UTC/TSYS/TIMESERVER.bunchPatternTable	10.0k	10.0
XFEL_MAGNETS_CHICANE/DOOCS/SXR2CPP.XFEL_MAGNETS_CHICANE_SXR2CPP.dtFs	10.0k	10.0
XFEL_MAGNETS_MAGNETML/DOOCS/BSL_2874_SA3.XFEL_MAGNETS_MAGNET_ML_BSL...	10.0k	10.0
▾ view		
☐ _sequence_id	10.0k	10.0
▾ COMO		
▾ MISC		
☐ EL_valve_trigger	10.0k	10.0
▾ MOTION		
☐ DEFLECTOR_Y	10.0k	10.0
☐ DEFLECTOR_Z	10.0k	10.0
☑ KNIFE_ROT	10.0k	10.0
☐ KNIFE_Y	10.0k	10.0
☐ SKIMMER1_Y	10.0k	10.0
☐ SKIMMER1_Z	10.0k	10.0
☐ SKIMMER3_Y	10.0k	10.0
☐ SKIMMER3_Z	10.0k	10.0
☐ VALVE_X	10.0k	10.0
☐ VALVE_Y	10.0k	10.0
☐ VALVE_Z	10.0k	10.0
▾ PRESSURE		
☐ STAGE_1	10.0k	10.0
☐ STAGE_2	10.0k	10.0
☐ STAGE_3	10.0k	10.0
▾ KBS		
▾ HFM		
☐ bender_downstr	10.0k	10.0

☑ Handle subscriptions automatically



default#-view#COMO\MOTION\KNIFE_ROT[waveform] - EXtra-m...

Double-clicking any view opens up the default plot

- `metropc` is a framework to build runtime-programmable processing or analysis pipelines
  - Build analysis as a graph of nodes called *views*
  - Each view may deliver a plottable result visualized
  - Code may be changed and re-injected within seconds
  - Takes care of IO, parallelization and common statistics automatically
- MetroProcessor device is a metropc frontend to take in data from Karabo
- EXtra-metro is a standalone GUI client to visualize view results
  - Loadable via modules from any GPFS-connected machine
- MetroOutput device is a client to plot view results on scenes
- https://rtd.xfel.eu/docs/metroprocessor/en/latest/
  https://rtd.xfel.eu/docs/metropc/en/latest/