Pycbc signal processing graph

- Shows processing graph from the application programmers viewpoint
- Shows how functions and kernels are called and where they live (layers)
- Shows how data flows
- Shows memory objects
  - Where they live
  - Who is the owner
  - Which kind of data do they hold
  - How long they live

Pycbc signal processing graph for single detector search pipeline using GPUs

Python

Start

StrainData

Injector

Resampler
16kHz -> 4096Hz

Highpass Filter

Straindata.
Convert()

SWIG WRAP

SWIG WRAP

SWIG WRAP

SWIG WRAP

clayer

real_vector_double_t s(t)

real_vector_single_t s(t)

GPU

libraries
(lal etc.)

frame_cpp

NRWaveInject

files

.gwf

.xml

Legend:

Data flow

Object destroyed

Function call

Composition (right side object
belongs to left side object)

Software layer

t

real_vector_single_t

Time domain data

real_vector_single_t

Frequency domain data

Pycbc signal processing graph for single detector search pipeline using GPUs

Python

[typ. 10^3] ParameterSpace

Straindata. Segmenting()

Overwhitening Filter

Straindata *= Overw.-Filt

TemplateBank

TemplateBank. PreConditionData()

Overwhiten render()

Straindata. render()

SWIG WRAP

SWIG WRAP

SWIG WRAP

SWIG WRAP

_vector_single_t s(t)

[typ. 15 segments freq data from typ 256s timedomain data -> (N=2^20)] complex_vector_single_t stilde(f) (Python list of datavector_cpu)

FFTs of overlapping input data

real_vector_single_t 1/Sn(f) alias inverse PSD alias overwhitening filter N=2^20

median - inverse

__mul()

render

render

IFFT - truncate - FFT (inplace)

[ ] complex_vector_single_t stilde(f)

real_vector_single_t 1/Sn(f)

GLUE XMLread

.xml

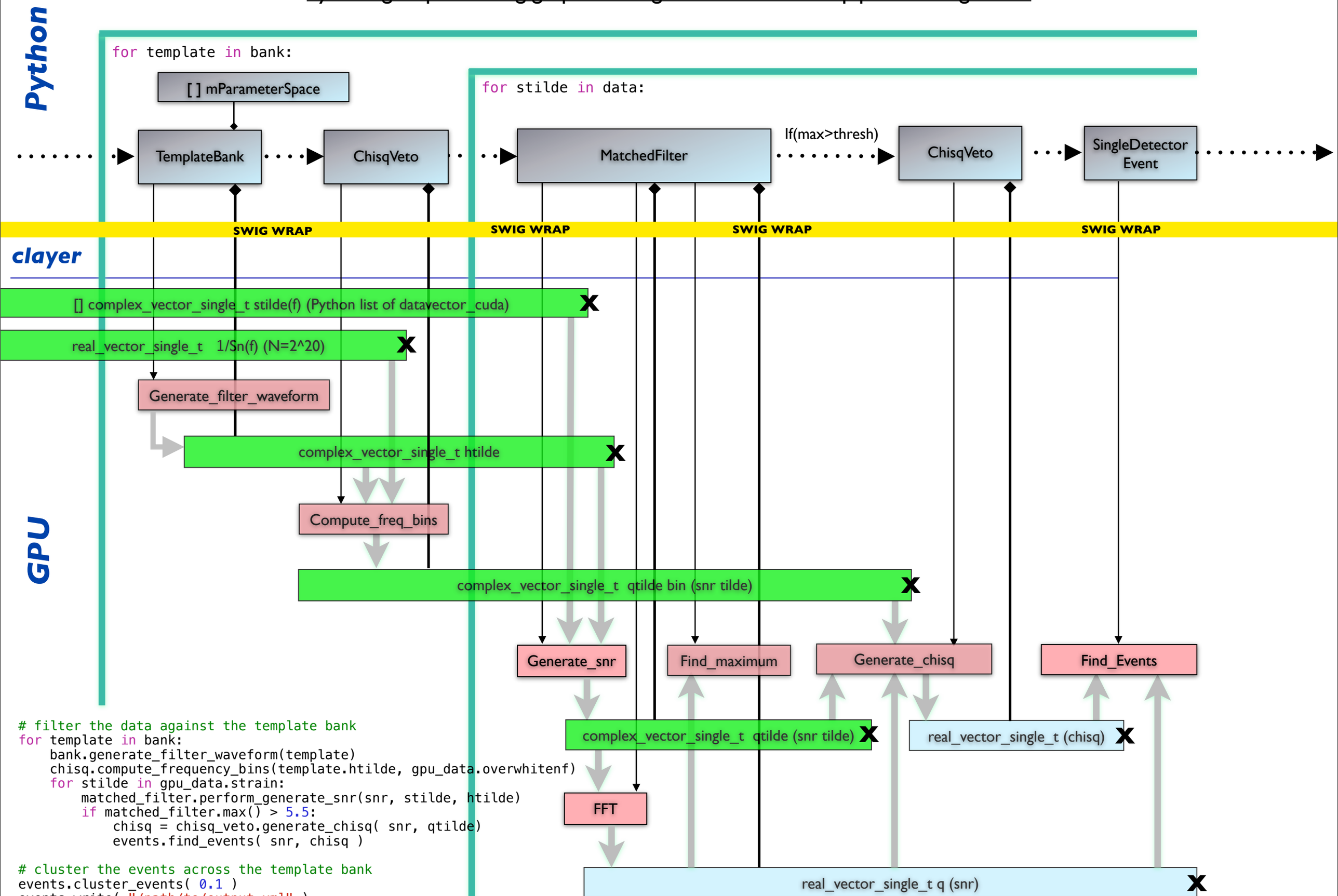
clay

GPU

GLUE/ Lal

files

Pycbc signal processing graph for single detector search pipeline using GPUs




```
# filter the data against the template bank
for template in bank:
    bank.generate_filter_waveform(template)
    chisq.compute_frequency_bins(template.htilde, gpu_data.overwhitenf)
    for stilde in gpu_data.strain:
        matched_filter.perform_generate_snr(snr, stilde, htilde)
        if matched_filter.max() > 5.5:
            chisq = chisq_veto.generate_chisq( snr, qtilde)
            events.find_events( snr, chisq )

# cluster the events across the template bank
events.cluster_events( 0.1 )
events.write( "/path/to/output.xml" )
```

Pycbc signal processing graph for single detector search pipeline using GPUs