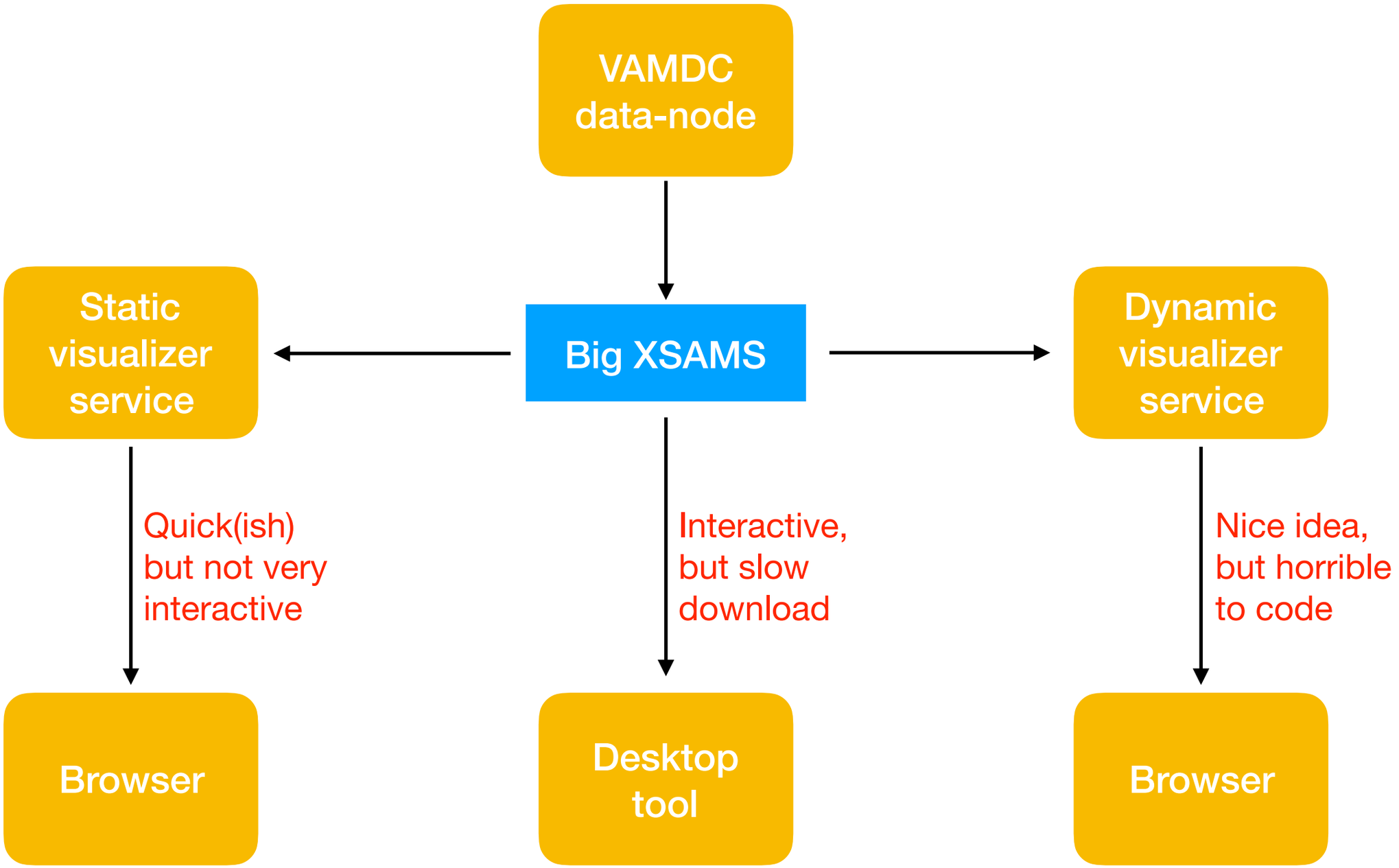
Dynamic visualization with Bokeh and Holoviews

Guy Rixon

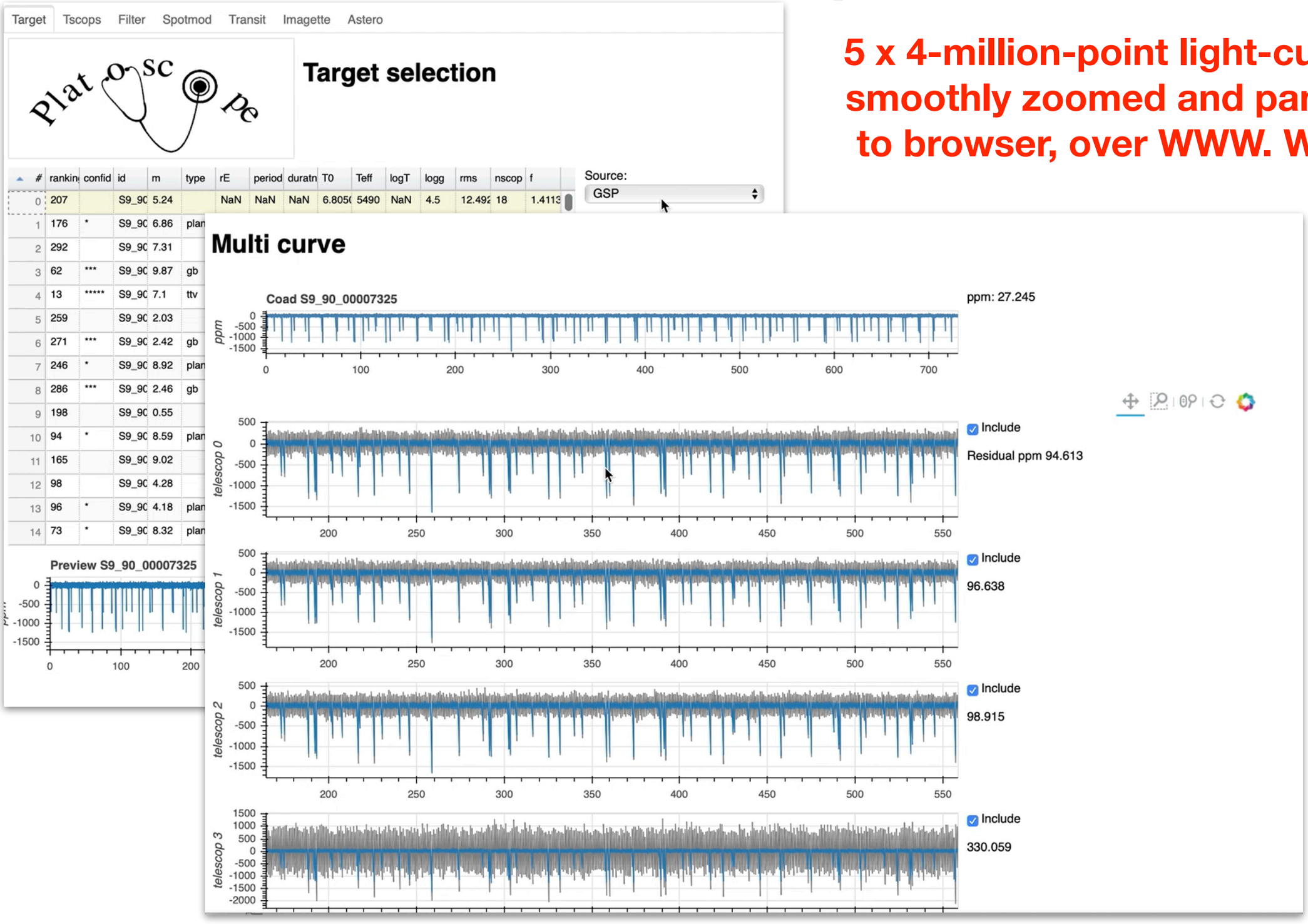
VAMDC annual meeting, 2020

VAMDC displays to date



Platoscope

5 x 4-million-point light-curves smoothly zoomed and panned to browser, over WWW. Wow.



Credit: Pascal Gutenburg and Jean-Christophe Malapert

Can I do that too?
Yes: demo here.

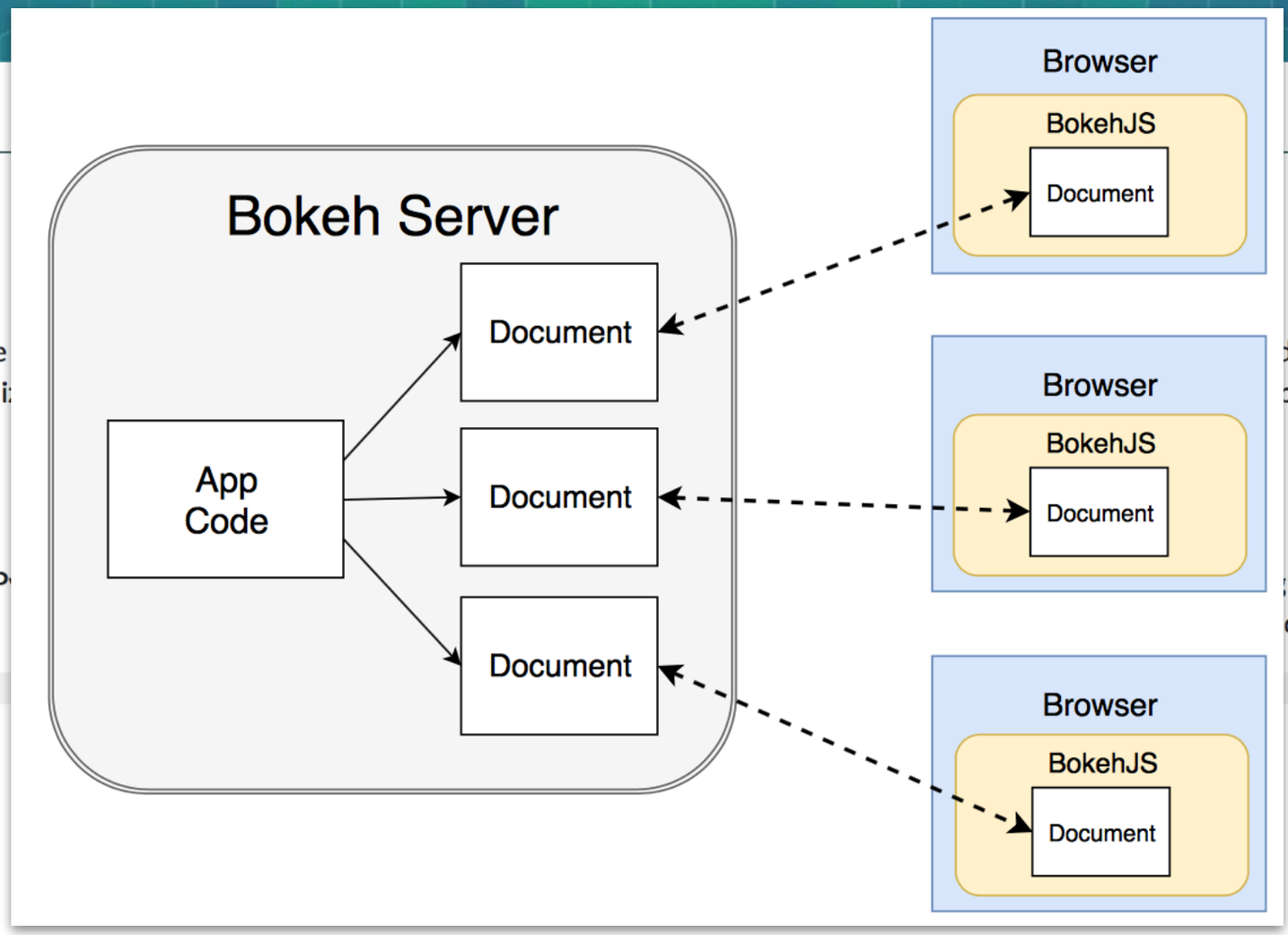
Bokeh...

bokeh Documentation Community Tutorials Demos Blog GitHub Donate

The Bokeh Visualization Library

Documentation

☆ Star 13,509



Flexible

Bokeh makes it simple to create
also can handle custom or specializ

d apps can be published in web
books.

Productive

Work in Python close to all the P
already familiar with.

the Bokeh server, is BSD
on GitHub.

…and Holoviews…

HoloViews

Stop plotting your data - annotate your data and let it visualize itself.

HoloViews is an open-source Python library designed to make data analysis and visualization seamless
and simple w lines of code, letting
you focus g.

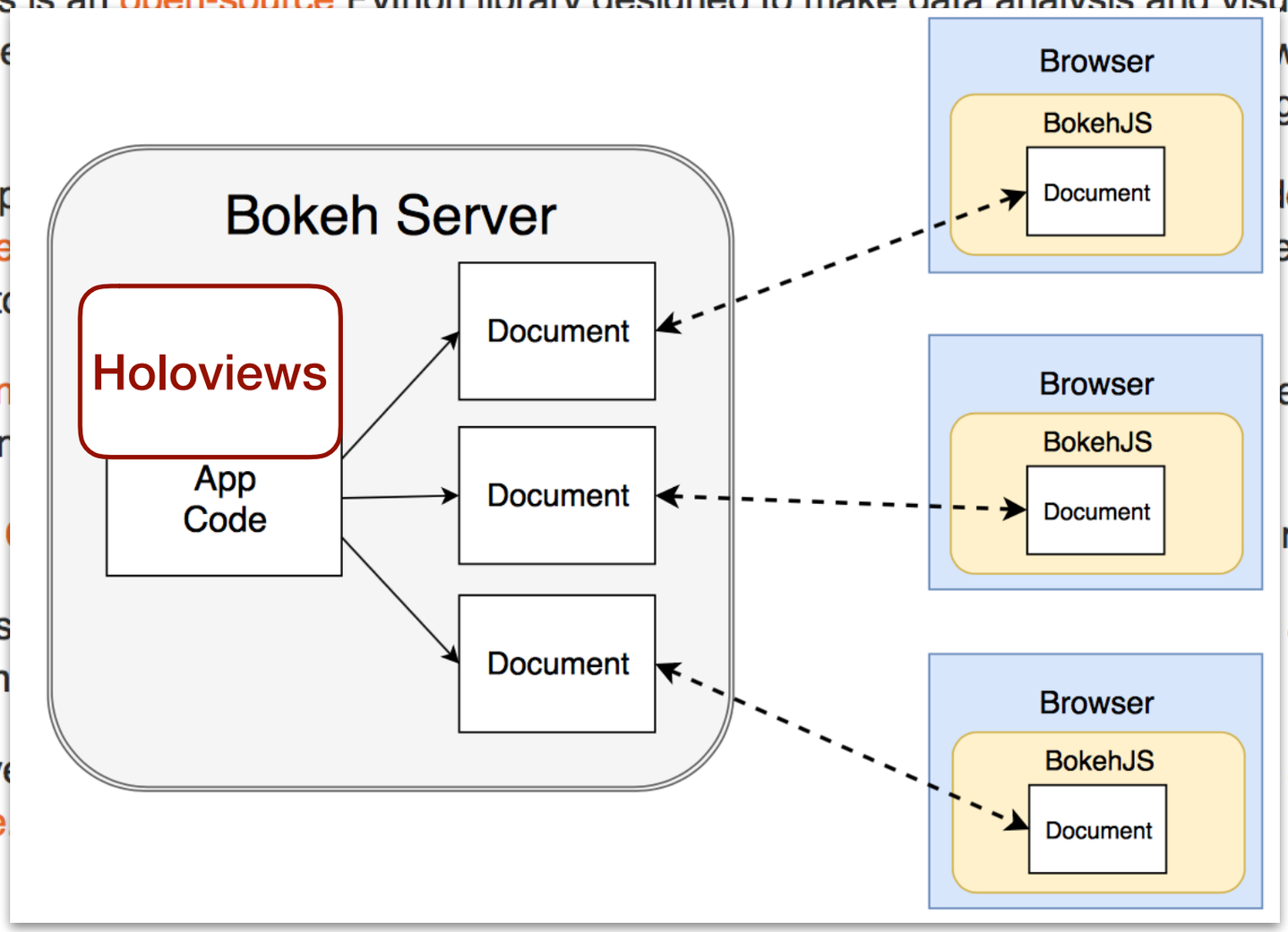
For examp lemos and apps and
the Refere e code, not just the
pictures, t

The Gettin ews, and is the
recommer

The User ready for further study.

The API is available more
convenien

If you have a topic on the HoloViz
Discourse

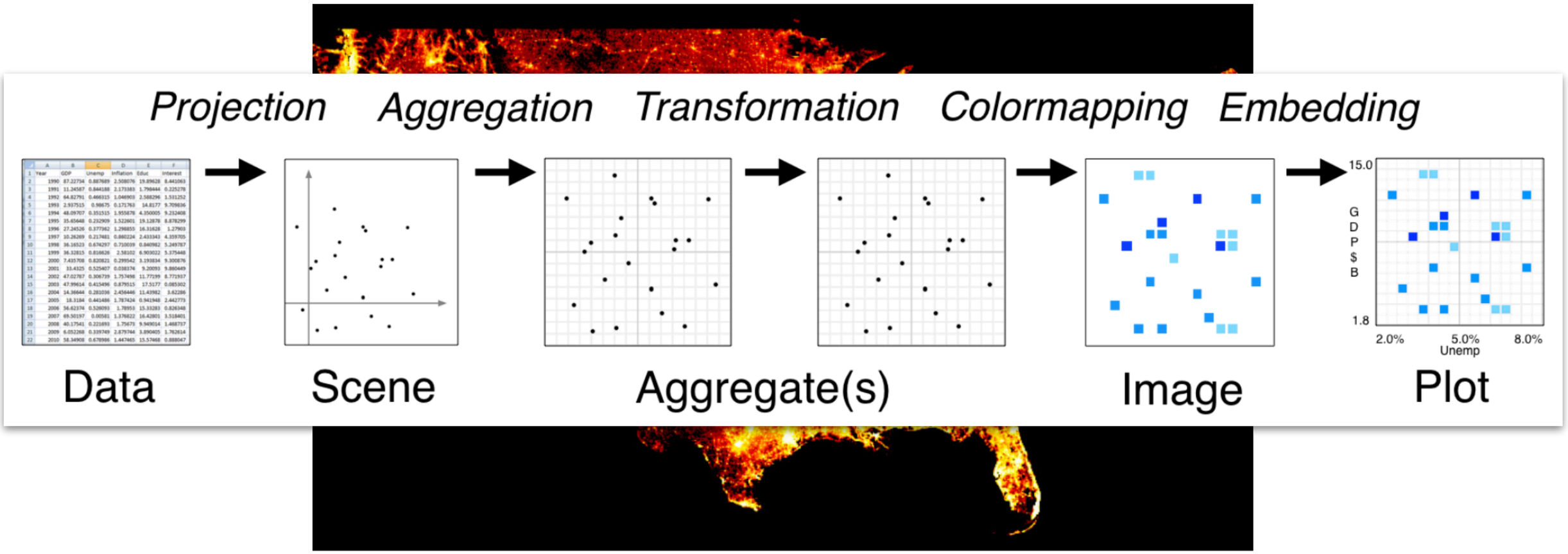


...and Datashader?

Accurately render even the largest data

Datashader is a graphics pipeline system for creating meaningful representations of large datasets quickly and flexibly. Datashader breaks the creation of images into a series of explicit steps that allow computations to be done on intermediate representations. This approach allows accurate and effective visualizations to be produced automatically without trial-and-error parameter tuning, and also makes it simple for data scientists to focus on particular data and relationships of interest in a principled way.

The computation-intensive steps in this process are written in Python but transparently compiled to machine code using Numba and flexibly distributed across CPU cores and processors using Dask or GPUs using CUDA, providing a highly optimized rendering pipeline that makes it practical to work with extremely large datasets even on standard hardware while exploiting distributed and GPU systems when available.



And is the code horrible?

No, not really.

```
import pandas as pd
import numpy as np
import holoviews as hv
from holoviews import opts, Dimension
hv.extension('bokeh')

from bokeh.plotting import curdoc
from bokeh.models import HoverTool


# Load the data table from file.
lines = pd.read_csv('all-lines.csv')
lines.Atom.apply(str)
print(lines.head())
print(lines.dtypes)
```

```
wavelength = hv.Dimension('Wavelength', label='Wavelength', unit='nm')
einsteinA = hv.Dimension('EinsteinA', label='Einstein A coefficient')
atomSymbol = hv.Dimension('Atom')

hover = HoverTool(tooltips=[('atom', '@Atom'), ('wavelength', '@Wavelength'), ('A', '@EinsteinA')])

# Organize a spike plot on the data.
spiky = hv.Spikes(lines, kdims=wavelength, vdims=[einsteinA, atomSymbol])
spiky.opts(width=1000, height=500, xlim=(350, 750), ylim=(0,5e10), tools=[hover])


# Reify the spike plot as a Bokeh figure.
fig = hv.render(spiky)

# Serve the figure as a Bokeh application
# making it available over the WWW.
curdoc().add_root(fig)
```

(That’s the whole program for my demo.)