Publicly Releasing a Large Simulation Dataset

Nathan Goldbaum

Postdoc, Data Exploration Lab

National Center for Supercomputing Applications

In collaboration with Matthew Turk, Kacper Kowalik, Mark Krumholz, and John Forbes

Outline

- Isolated galaxy simulations
- How to write analysis pipelines for reproducible data workflows
  - Useful python tips!
- Making the data public
- Value-adds for public data

Running these simulations is sort of like…

All credit to Ridley Scott, Donald Glover, and 20th Century Fox

Running these simulations is sort of like…

All credit to Ridley Scott, Donald Glover, and 20th Century Fox

The reality:

10. ssh pfe (ssh)

```
ngoldbau@pfe20:~> cd /nobackup/ngoldbau/nofeedback_hgf/
ngoldbau@pfe20:/nobackup/ngoldbau/nofeedback_hgf> cat jobsub.sh
#PBS -S /bin/bash
#PBS -N nofeedback_hgf
#PBS -l select=1:ncpus=24:mpiprocs=12:model=has+10:ncpus=24:mpiprocs=24:model=has,
walltime=24:00:00
#PBS -W group_list=s1414
#PBS -q long
#PBS -m abe
#PBS -r n

module load comp-intel/2015.3.187 mpi-sgi/mpt.2.12r16 python hdf5 szip

export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/u/ngoldbau/local/lib

cd $PBS_O_WORKDIR

mpiexec -np 252 ./enzo.exe -d -r DD0186/DD0186 >> shell.out 2>&1
ngoldbau@pfe20:/nobackup/ngoldbau/nofeedback_hgf> qsub jobsub.sh
```

The reality:

Job 1434849 for ngoldbau on Pleiades failed

Inbox x

support@nas.nasa.gov via illinois.e 2:56 AM (18 hours ago)

to ngoldbau

Your Pleiades job 1434849.pbspl1.nas.nasa.gov terminated due to one or more nodes running out of memory. Node r509i5n12 ran out of memory and rebooted; others may have run out of memory as well.

While this is typically caused by a user program using too much memory, it may also be caused by a system issue. If you need help determining the source of the problem, please email support@nas.nasa.gov and we will be happy to help.

For information on how to check the memory usage or request more memory for your PBS jobs, see
http://www.nas.nasa.gov/hecc/support/kb/memory-usage-overview_216.html

Thank you,

NAS User Services

Email: support@nas.nasa.gov
Local: 650-604-4444
Toll Free: 800-331-8737

Outputs are written to disk every million years of simulation time

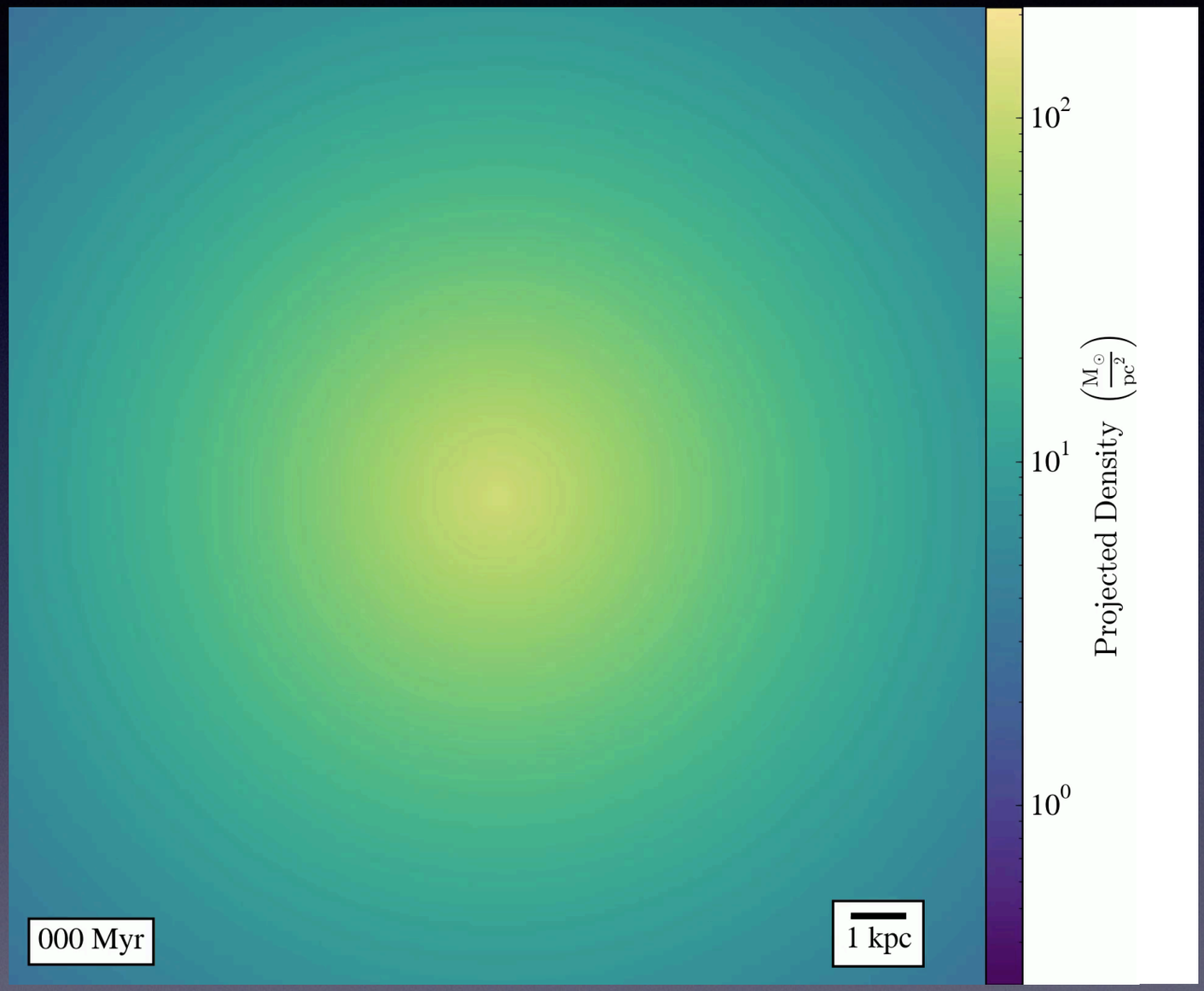
600 outputs

Each simulation needs:

50,000 — 200,000 CPU-hours

200-400 cores for a few weeks

2.5 TB disk space



https://youtu.be/t5RpOu1SEyA

Outputs are written to disk every million years of simulation time

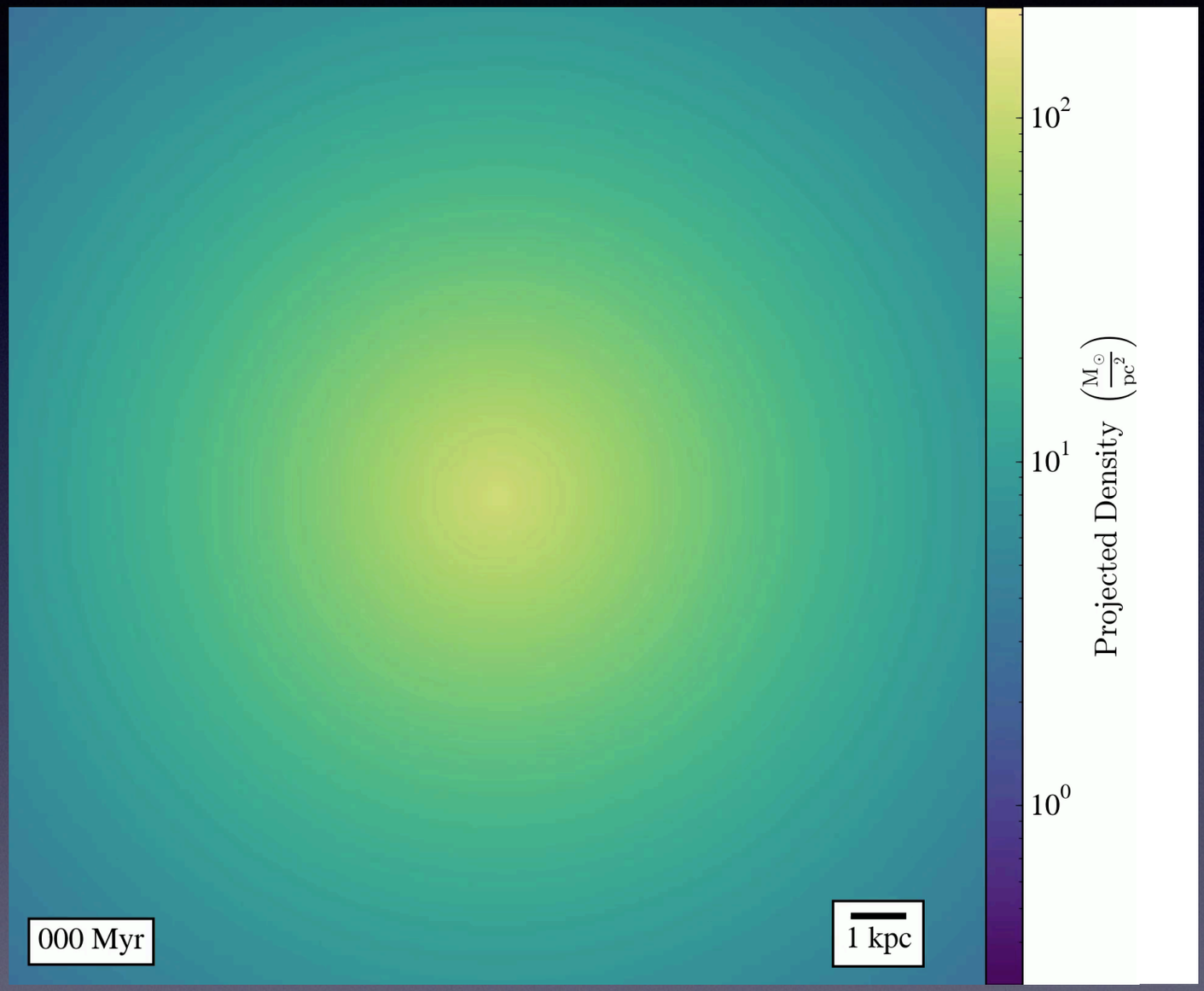
600 outputs

Each simulation needs:

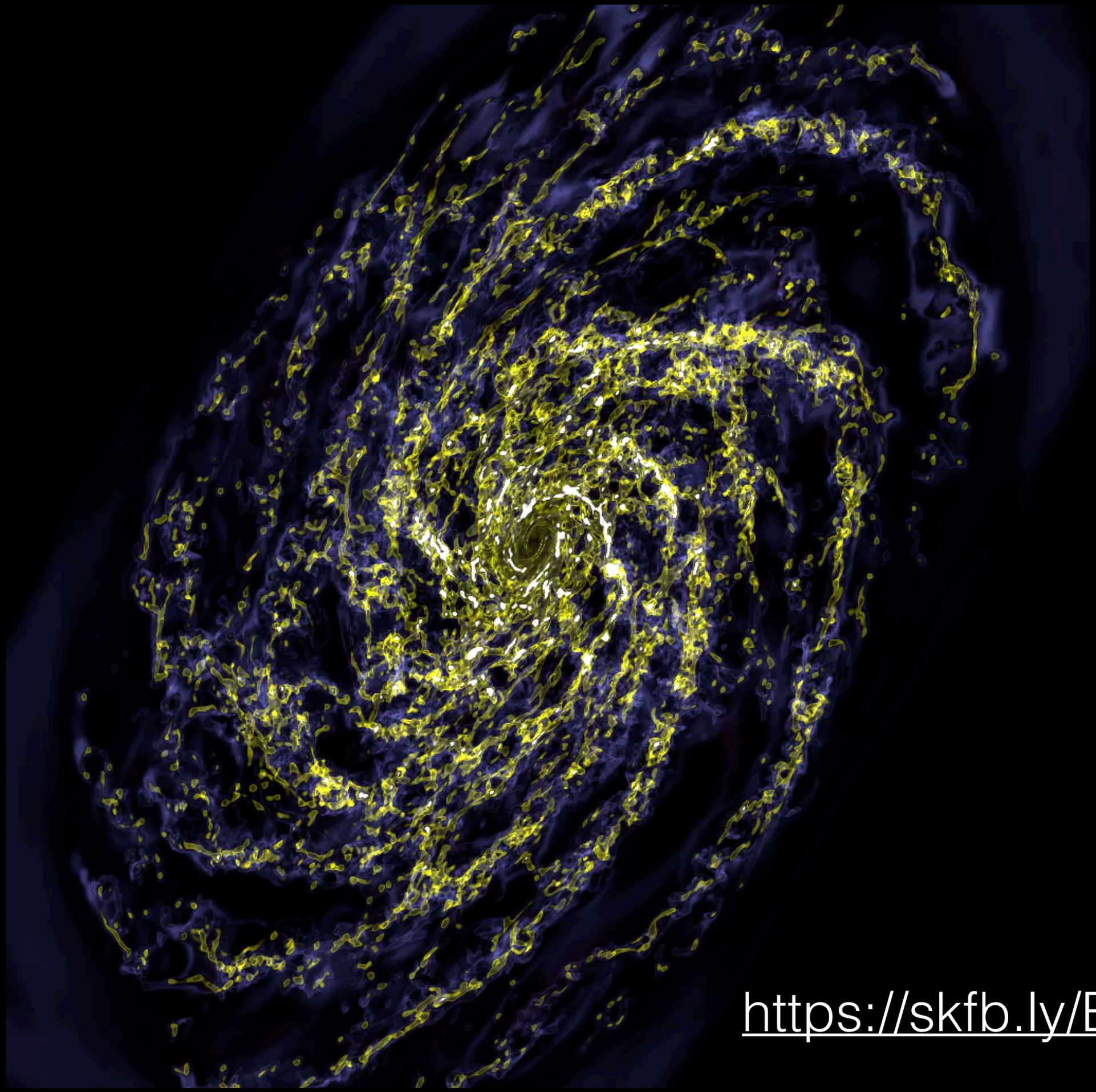
50,000 — 200,000 CPU-hours

200-400 cores for a few weeks

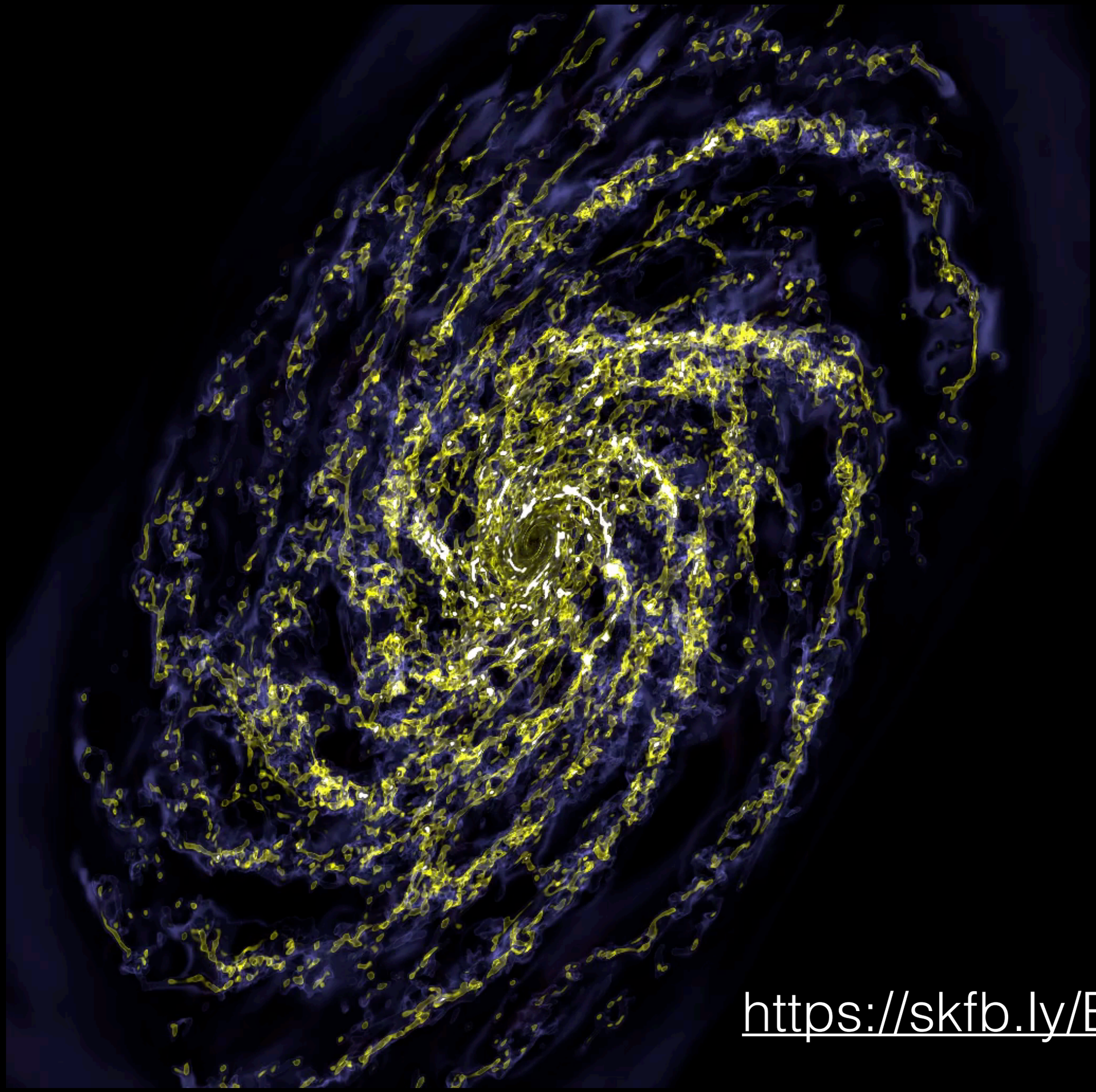
2.5 TB disk space



https://youtu.be/t5RpOu1SEyA



https://skfb.ly/EOvP



https://skfb.ly/EOvP

Open Science

- Enzo simulation code:

  http://enzo-project.org

- yt analysis package

  http://yt-project.org

- Custom analysis pipeline

  http://bitbucket.org/ngoldbaum/galaxy_analysis

- Many other free and open libraries

  NumPy, SciPy, IPython, Cython, matplotlib, hdf5, h5py, scikit-image, numexpr

Open Science

- Enzo simulation code:

  http://enzo-project.org

- yt analysis package

Public from day one

100% Free Software (BSD License)

- Custom analysis pipeline

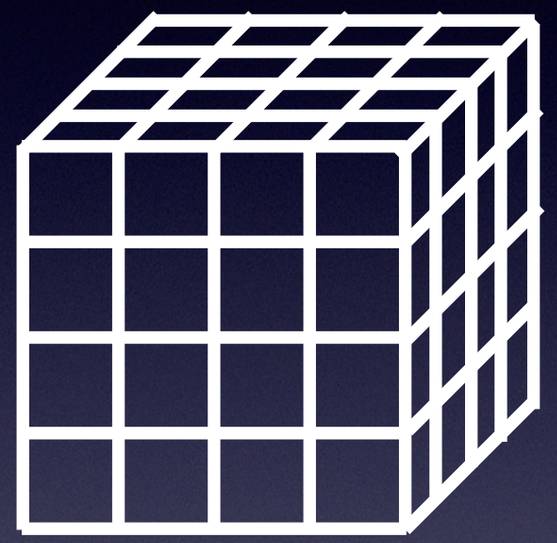
  http://bitbucket.org/ngoldbaum/galaxy_analysis

- Many other free and open libraries

  NumPy, SciPy, IPython, Cython, matplotlib, hdf5, h5py, scikit-image, numexpr

`galanyl`
a galaxy analysis toolkit

Simulation data



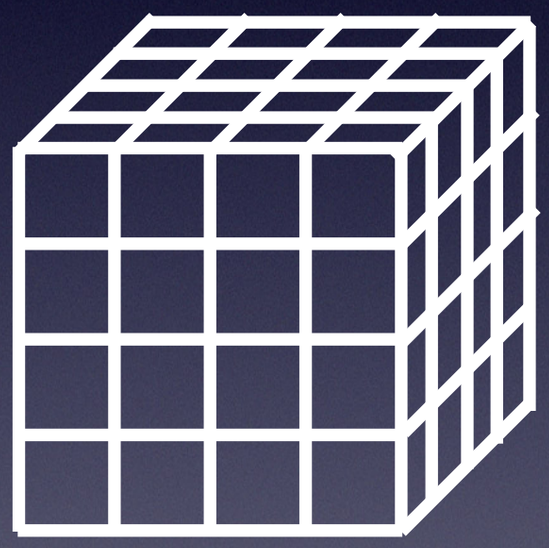
Enzo AMR
Custom data
format based
on HDF5

I/O
Intermediate
processing

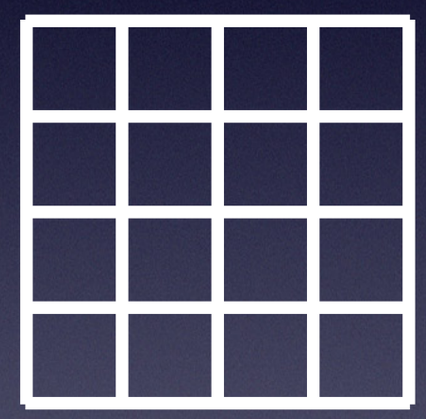
Density
Thermal energy
Velocity
Potential

`galanyl`
a galaxy analysis toolkit

Grid slabs



Derived 2D data



Surface density of gas, stars, star formation rate
2D Toomre Q maps
Velocity dispersions
Many other fields

Writing code for reproducible workflows

- Make your analysis code a proper python package
  - write a REAMDE and a setup.py, add a license
- Use version control
- Cache intermediate results
  - `joblib`, `h5py`, `pyfits`
- Write code you will grok six months later

Tips for reproducible data analysis pipelines in Python

Tips for reproducible data analysis pipelines in Python

Examples adapted from `galanyl` package,
see https://bitbucket.org/ngoldbaum/galaxy_analysis

```
class GalaxyAnalyzer(object):

    def save_to_hdf5(self):
        """Cache GalaxyAnalyzer data to an hdf5 file"""
        # save data to disk

    @classmethod
    def from_hdf5_file(cls, h5_path):
        """Create a GalaxyAnalyzer object from cached hdf5 data

        Parameters:
          h5_path: string
             Path to a directory containing cached hdf5 data.
             Must have the same format as data created by
             the save_to_hdf5 function.

        >>> g = GalaxyAnalyzer.from_hdf5_file("/path/to/data")
        """
        g = super(GalaxyAnalyzer, cls).__new__(cls)
        # initialize g from data located at h5_path
```

```
import numpy as np

def get_line_circle_intercepts(x1, x2, y1, y2, r):
    d_x = x2 - x1
    d_y = y2 - y1
    d_r = np.sqrt(d_x*d_x + d_y*d_y)
    D = x1*y2 - x2*y1

    if x1 == x2:
        yint = (-D*d_x + abs(d_y)*
                 np.sqrt(r**2*d_r**2 - D**2))/d_r**2
        xint = x1
        return [(xint, yint), (xint, -yint)]
    else:
        xint = (D*d_y + d_x*s*np.sqrt(r**2*d_r**2 - D**2))/d_r**2
        yint = y1
        return [(xint, yint), (-xint, yint)]
```

```
cimport cython
from libc.math cimport sqrt, copysign, abs

@cython.cdivision(True)
def get_line_circle_intercepts(double x1, double x2, double y1,
                               double y2, double r):
    cdef double d_x = x2 - x1
    cdef double d_y = y2 - y1
    cdef double d_r = sqrt(d_x*d_x + d_y*d_y)
    cdef double D = x1*y2 - x2*y1


    if x1==x2:
        yint = (-D*d_x + abs(d_y)*sqrt(r**2*d_r**2 - D**2))/d_r**2
        xint = x1
        return [(xint, yint), (xint, -yint)]
    else:
        xint = (D*d_y + d_x*s*sqrt(r**2*d_r**2 - D**2))/d_r**2
        yint = y1
        return [(xint, yint), (-xint, yint)]
```

```
cimport cython
from libc.math cimport sqrt, copysign, abs

@cython.cdivision(True)
def get_line_circle_intercepts(double x1, double x2, double y1,
                               double y2, double r):
    cdef double d_x = x2 - x1
    cdef double d_y = y2 - y1
    cde
    cde
    if x1==x2:
        yint = (-D*d_x + abs(d_y)*sqrt(r**2*d_r**2 - D**2))/d_r**2
        xint = x1
        return [(xint, yint), (xint, -yint)]
    else:
        xint = (D*d_y + d_x*s*sqrt(r**2*d_r**2 - D**2))/d_r**2
        yint = y1
        return [(xint, yint), (-xint, yint)]
```

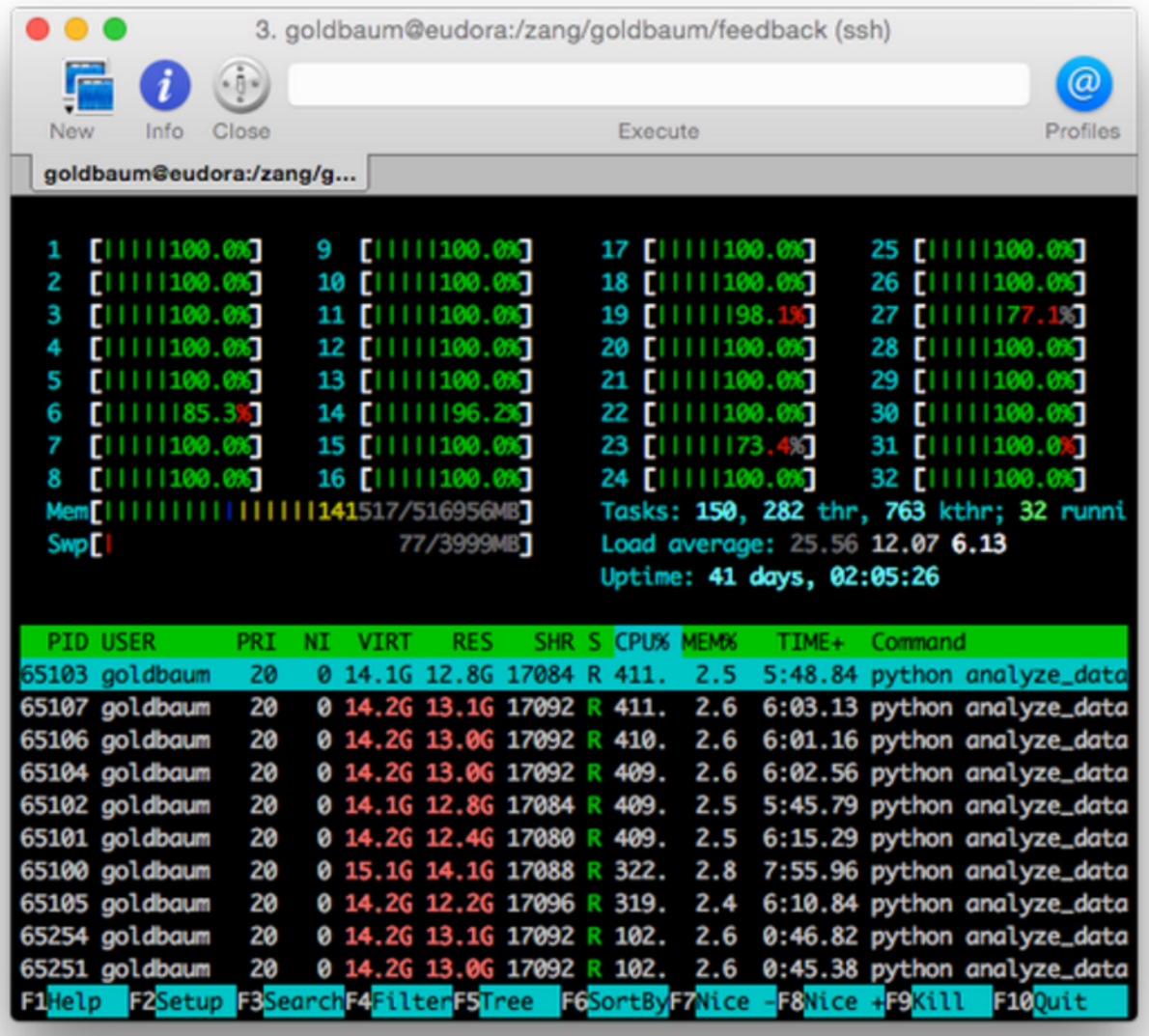
Cython version is ~1000x faster!

```
@cython.cdivision(True)
@cython.wraparound(False)
@cython.boundscheck(False)
cdef void reduce_dispersion(f8_t* ret, f8_t* s0, f8_t* s1, f8_t* s2,
                            intp_t* indices, f8_t* v, f8_t* m, intp_t nparticles,
                            intp_t size, intp_t* s, intp_t* m0ptr,
                            intp_t* m1ptr, intp_t* m2ptr) nogil:

    cdef intp_t i

    for i in parallel.prange(nparticles):
        process_particle(s0, s1, s2, indices[i], v[i], m[i],
                         m0ptr, m1ptr, m2ptr, size, s)

    for i in parallel.prange(s[0]*s[1]*s[2]):
        if s0[i] == 0:
            ret[i] = 0
        else:
            ret[i] = sqrt(s0[i]*s2[i] - s1[i]*s1[i])/s0[i]
        if isnan(ret[i]):
            ret[i] = 0
```

```
@cython.cdivisi
@cython.wraparou
@cython.boundscl
cdef void reduce                                              : nparticles,


    cdef intp_t

    for i in pa
        process_


    for i in pa
        if s0[i
            ret
        else:
            ret
        if isna
            ret
```

Nathan Goldbaum @njgoldbaum · 9 Feb 2015

Takin' care of business



1

Making data publicly available

http://hub.yt/data/goldbaum2015a
http://hub.yt/data/goldbaum2016a

Index of /data/goldbaum2016a/

../		
feedback_20pc/	19-Jan-2016 17:54	-
feedback_20pc_hgf/	19-Jan-2016 17:54	-
feedback_20pc_lgf/	19-Jan-2016 17:54	-
LICENSE.txt	10-Nov-2015 16:09	123
README.txt	16-Dec-2015 22:58	4487
manifest	19-Jan-2016 17:55	263045

Data live on spinning disk, served by nginx
Backed up on tape at NASA Ames

National Data Service

The National Data Service (NDS) is an emerging vision for how scientists and researchers across all disciplines can find, reuse, and publish data. It builds on the data archiving and sharing efforts already underway within specific communities and links them together with a common set of tools designed around the following capabilities:

Search

The NDS will allow users to easily search for data across disciplinary boundaries. As users hone in on data of interest, they can easily switch to discipline-specific tools.

Publish

The NDS will connect users to tools for building and sharing collections of data. It will help users find and deliver data to the best repository for data-publishing.

Link

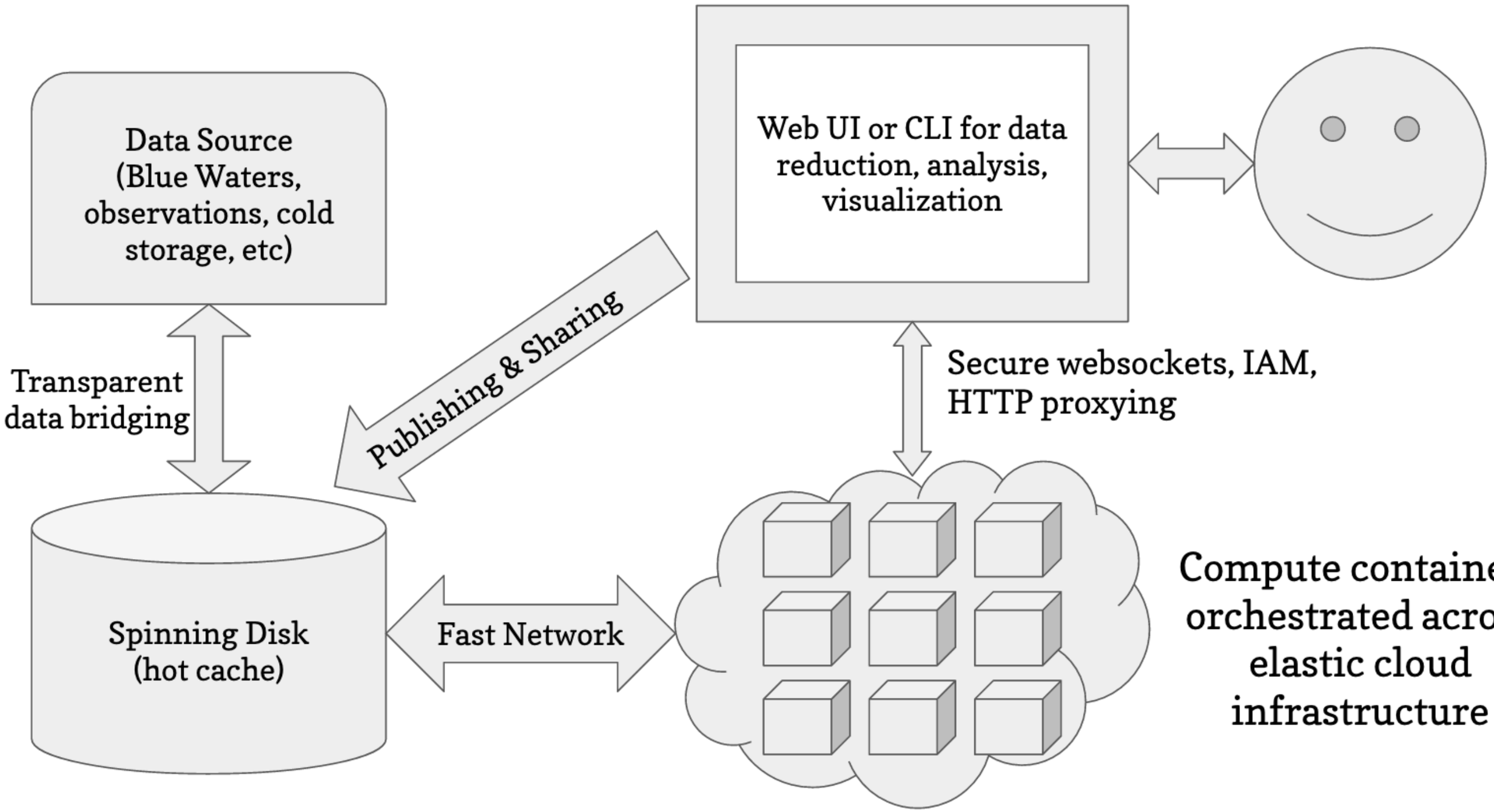
The NDS will create robust connections between data and published articles. When researchers reference an article, they have ready access to the underlying data.

Reuse

The NDS will not only provide access to data for download, it will provide tools for transferring data to processing platforms or allow analysis to be attached to the data.

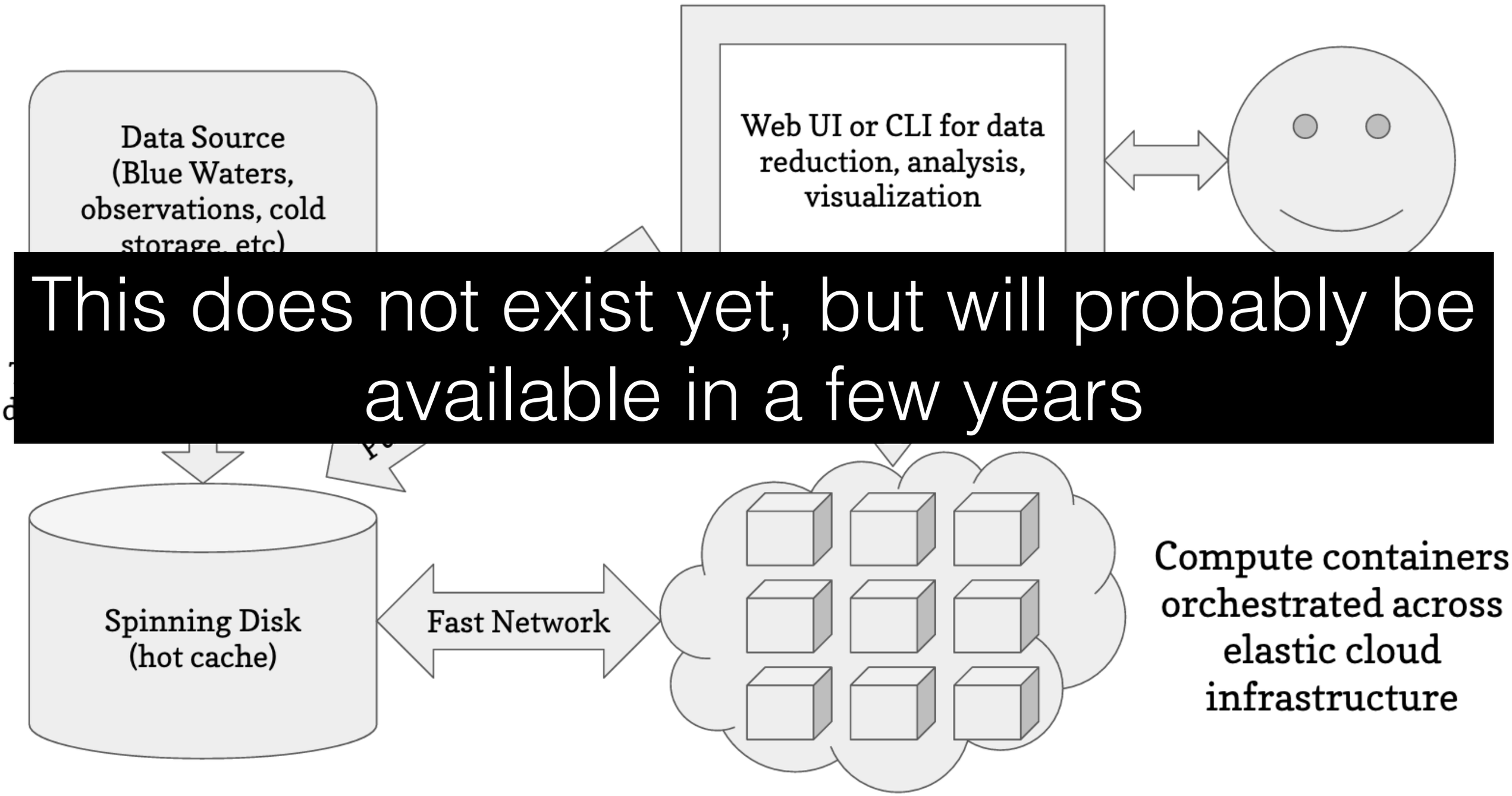
NSF-funded work in progress effort

The data deliverator



Slide courtesy Matthew Turk

The data deliverator



This does not exist yet, but will probably be available in a few years

Slide courtesy Matthew Turk

Live demos powered by public data

https://demo.use.yt