XXX Canary Islands Winter School, Nov 5-9, 2018.

http://ls.st/fqn

The Challenge of Large Dataset Analysis

or why I believe exciting times are ahead

Prof. Mario Juric
DIRAC Institute | eScience Institute | UW Astronomy

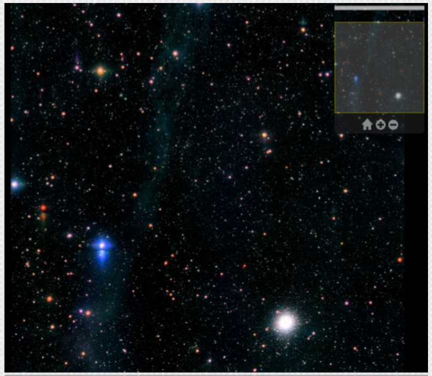
Data Intensive Research in Astrophysics and Cosmology
COLLEGE OF ARTS & SCIENCES | UNIVERSITY *of* WASHINGTON

Synopsis

- How to think about survey datasets
- What do surveys deliver (with LSST as an example)
- How will we analyze PB-scale datasets
- The possibilities beyond the current paradigm

Telescopes as just (Expensive) Peripherals

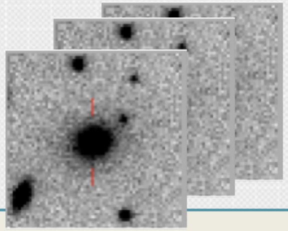
Peripherals ➔



Datasets

This is the Important Bit: From Data to Knowledge

Computationally (and cognitively) expensive, science-case speciific



And metadata!

Scientists

Scientists **Model** ← *inference* – **Data** *Projects*

Scientists **Model** ← *inference* – (Scientists) **Catalog** (Projects) ← Data Processing (Projects) – **Data** (Projects)

Computationally cheaper,
Easier to understand,
Science-case specific

- Computationally expensive, general
- Reprojection; may or may not involve compression
- Almost always introduces some information loss
- Data Processing == Instrumental Calibration + Measurement

Surveys: Turning the Sky into a Databases

- **The ultimate deliverable of a survey is not the telescope, nor the instruments; it is the fully reduced data.**
- All science comes from catalogs and images
- The telescope is still an (expensive) data collection peripheral

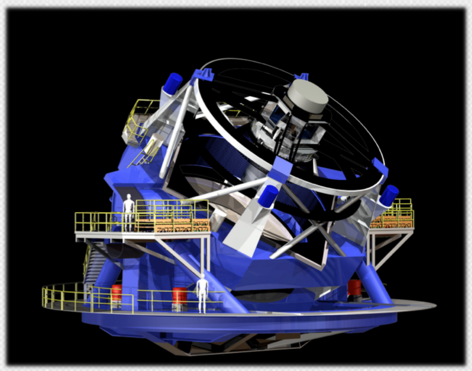


Table 4: Level 2 Catalog `Object` Table

Name	Type	Unit	Description
psRadecTai	double	time	Point source model: Time at which the object was at position `radec`.
psPm	float[2]	mas/yr	Point source model: Proper motion vector.
psParallax	float	mas	Point source model: Parallax.
psFlux	float[ugrizy]	nmgy	Point source model fluxes[58].
psCov	float[66]	various	Point-source model covariance matrix[59].
psLnL	float		Natural *log* likelihood of the observed data given the point source model.
bdRadec	double[2]	degrees	B+D model[60]: (α, δ) position of the object at time `radecTai`, in each band.

Peripherals → Code & Machines → Databases

#1 Challenge:

General purpose processing while minimizing information loss.

Guiding Principles for LSST Data Products

- There are virtually infinite options on what quantities (features) one can measure on images. But if catalog generation is understood as a *(generalized) cost reduction tool*, the guiding principles become easier to define:

1. **Maximize science enabled by the catalogs**
   - Working with images takes time and resources; a large fraction of science cases should be enabled by just the catalog.
   - Be considerate to the user: provide even sub-optimal measurements if they will enable leveraging of existing experience and tools
2. **Minimize information loss**
   - Choose good models
   - Provide (as much as possible) estimates of likelihood surfaces, not just single point estimators
3. **Provide and document the transformation (the software)**
   - Measurements are becoming increasingly complex and systematics limited; need to be maximally transparent about *how* they're done

What LSST will Deliver:
A Data Stream and a Database

- A stream of ~10 million time-domain events per night, detected and transmitted to event distribution networks within 60 seconds of observation.
- A catalog of orbits for ~6 million bodies in the Solar System.

Prompt

- A catalog of ~37 billion objects (20B galaxies, 17B stars), ~7 trillion single-epoch detections (“sources”), and ~30 trillion forced sources, produced annually, accessible through online databases.
- Deep co-added images.

Data Rel.

Prompt: Time-Domain Event Alerts

- We expect a high rate of alerts, **approaching 10 million per night**. We'll also provide an ***alert filtering service***, to select subsets of alerts, as well as serve the full stream to external ***event brokers***.

- **Each alert will include the following:**
  - Alert and database ID: IDs uniquely identifying this alert.
  - The photometric, astrometric, and shape characterization of the detected source
  - 30x30 pixel (on average) cut-out of the difference image (FITS)
  - 30x30 pixel (on average) cut-out of the template image (FITS)
  - The time series (up to a year) of all previous detections of this source
  - Various summary statistics ("features") computed of the time series

- **The goal is to *quickly* transmit nearly everything LSST knows about any given event, enabling downstream classification and decision making**

Annual Data Releases

- **Made available in *Data Releases***
  - Annually, except for Year 1
    - Two DRs for the first year of data

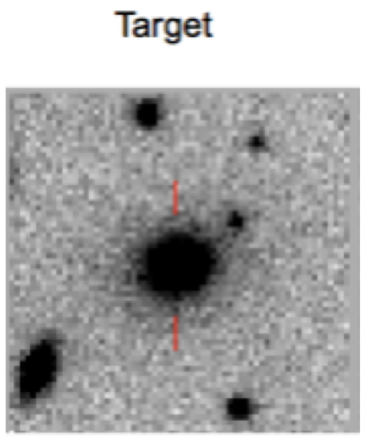
- **Well calibrated, consistently processed, catalogs and images**
  - Catalogs of objects, detections, detections in difference images, etc.

- **Complete reprocessing of all data, for each release**
  - Every DR will reprocess ***all*** data taken up to the beginning of that DR

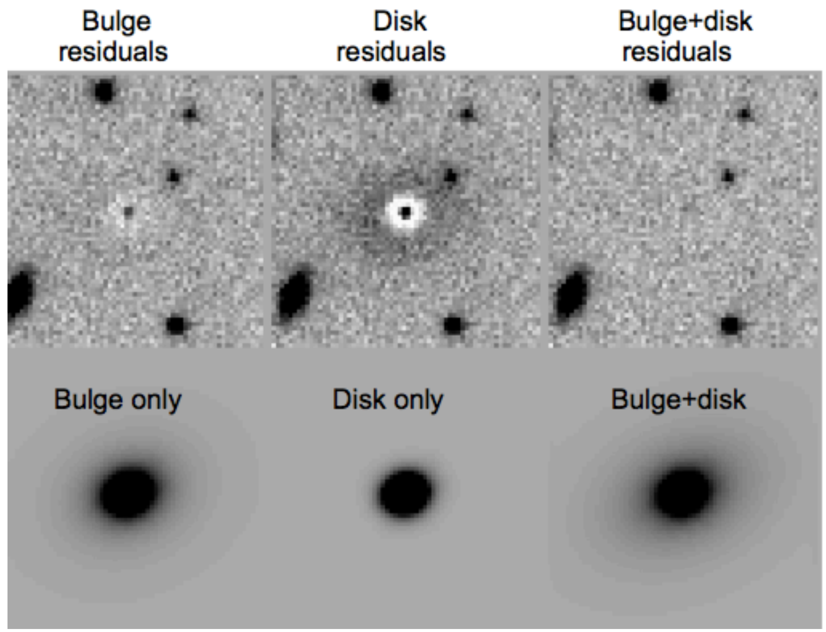
- **Projected catalog sizes:**
  - **18 billion objects** (DR1) → **37 billion** (DR11)
  - **750 billion observations** (DR1) → **30 trillion** (DR11)

Data Release Catalog Contents

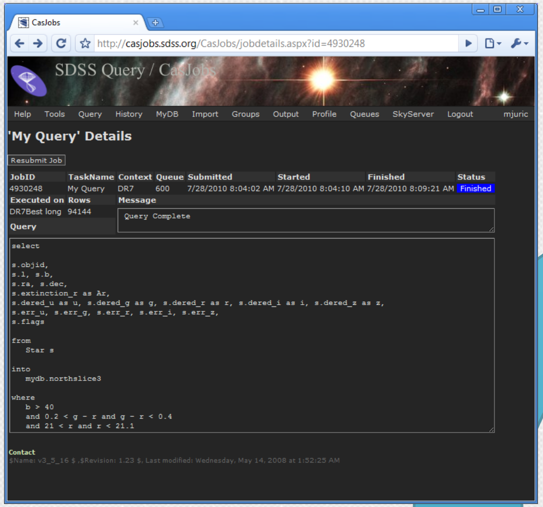
- **Object characterization (models):**
  - Moving Point Source model
  - Double Sérsic model (bulge+disk)
    - Maximum likelihood peak
    - <u>**Samples of the posterior (hundreds)**</u>
- **Object characterization (non-parametric):**
  - Centroid: (α, δ), per band
  - Adaptive moments and ellipticity measures (per band)
  - Aperture fluxes and Petrosian and Kron fluxes and radii (per band)
- **Colors:**
  - Seeing-independent measure of object color
- **Variability statistics:**
  - Period, low-order light-curve moments, etc.

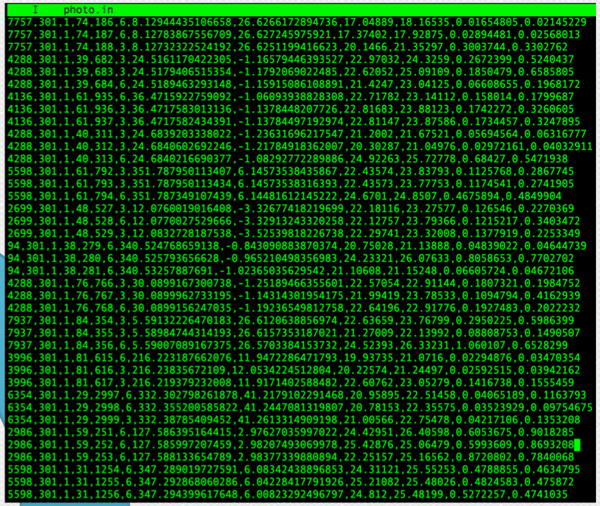


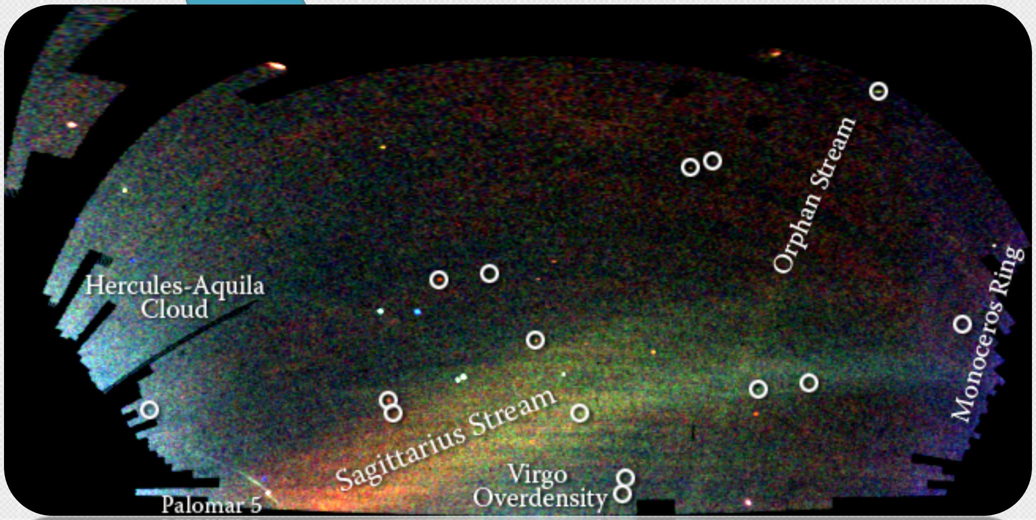
LSST Science Book, Fig. 9.3



Analysis: Subset – Download – Analyze







Data Volumes

	ZTF	LSST
Number of detections	1 trillion	7 trillion
Number of objects	1 billion	37 billion
Nightly alert rate	1 million	10 million
Nightly data rate	1.4 TB	15 TB
Alert latency	< 20 minutes	60 seconds

Science analysis code

~50kb

If the data is big…

… bring the code to the data.

What LSST will Deliver: A Data Stream, a Database, and a (small) Cloud

- A stream of ~10 million time-domain events per night, detected and transmitted to event distribution networks within 60 seconds of observation.
- A catalog of orbits for ~6 million bodies in the Solar System.

Prompt

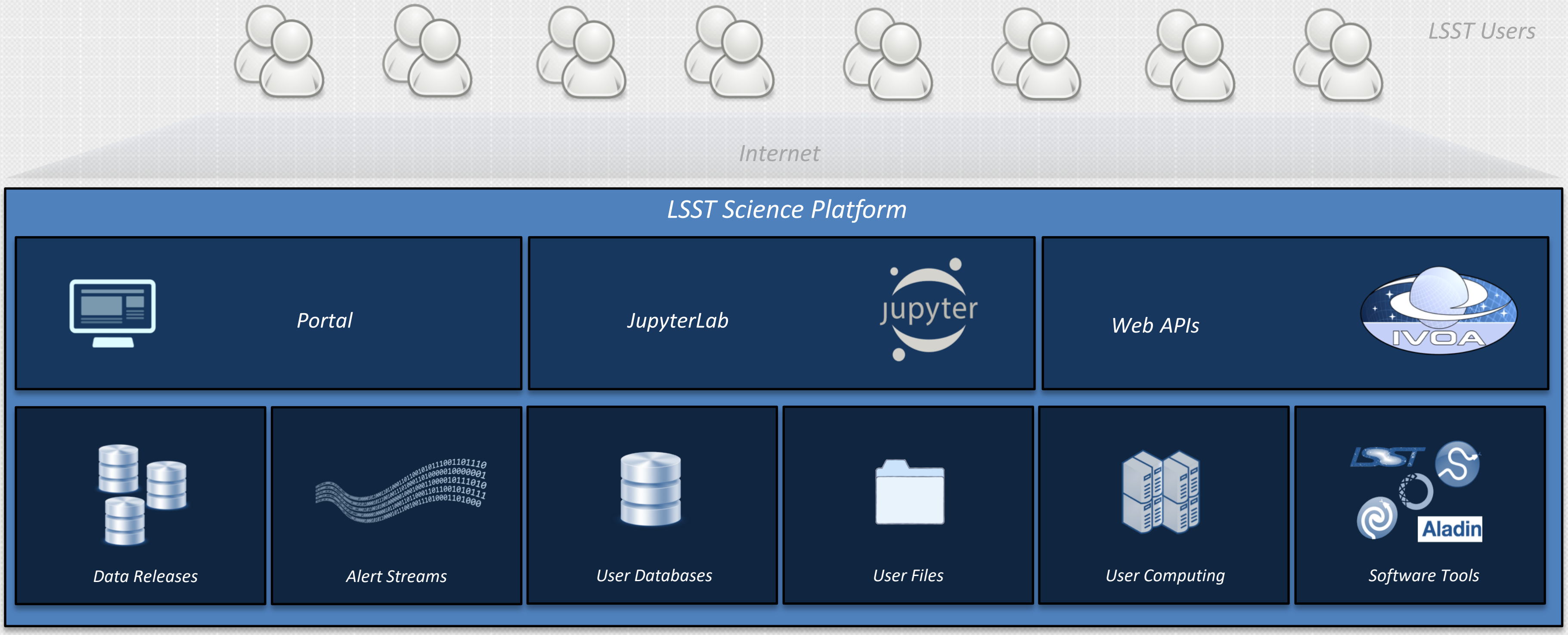
- A catalog of ~37 billion objects (20B galaxies, 17B stars), ~7 trillion single-epoch detections ("sources"), and ~30 trillion forced sources, produced annually, accessible through online databases.
- Deep co-added images.

Data Rel.

- Services and computing resources at the Data Access Centers to enable end-user analysis and generation of more added-value data products.

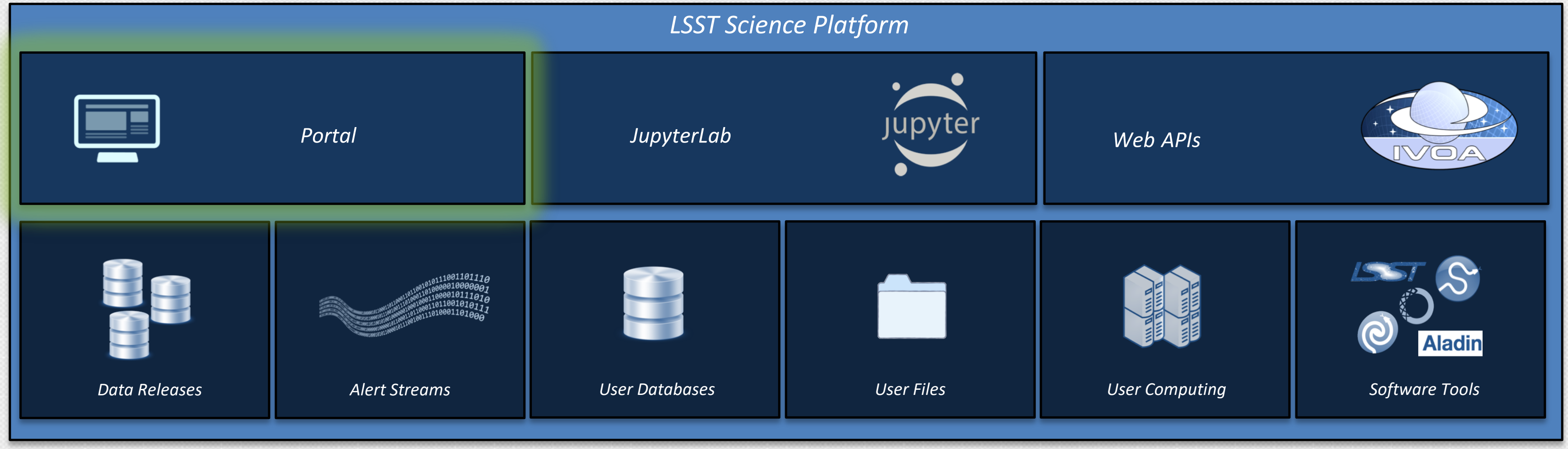
User Gen.

The LSST Science Platform:
Accessing LSST Data and Enabling LSST Science



The **LSST Science Platform** is a set of integrated web applications and services deployed at the LSST Data Access Centers (DACs) through which the scientific community will access, visualize, subset and perform next-to-the-data analysis of the data.

LSST Portal: The Web Window into the LSST Archive

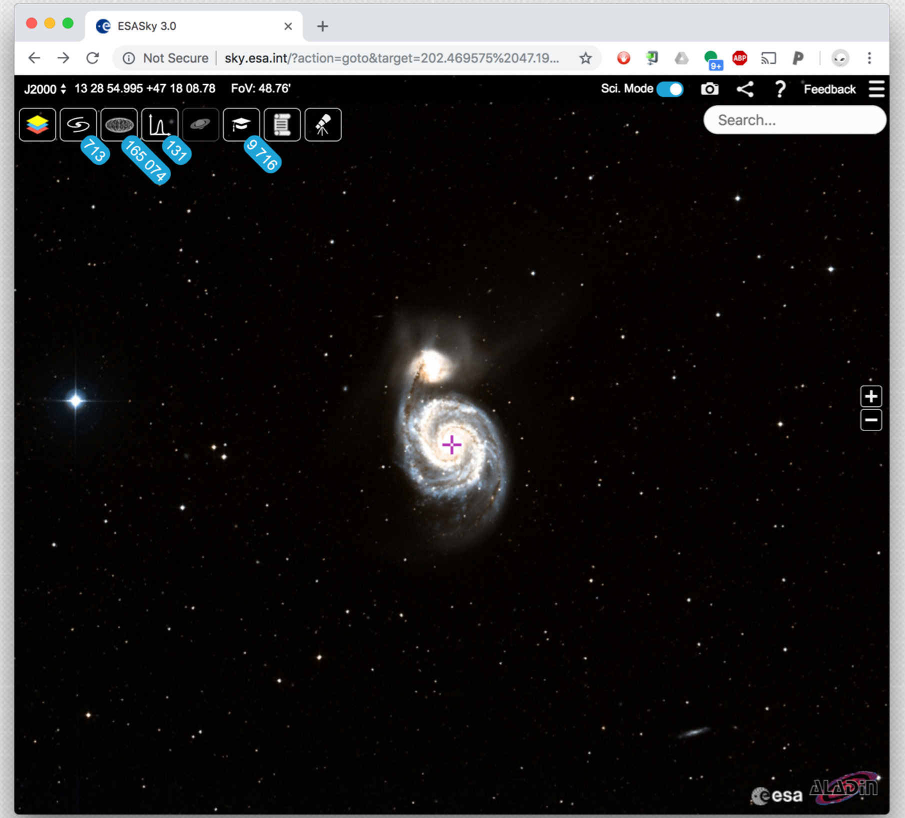


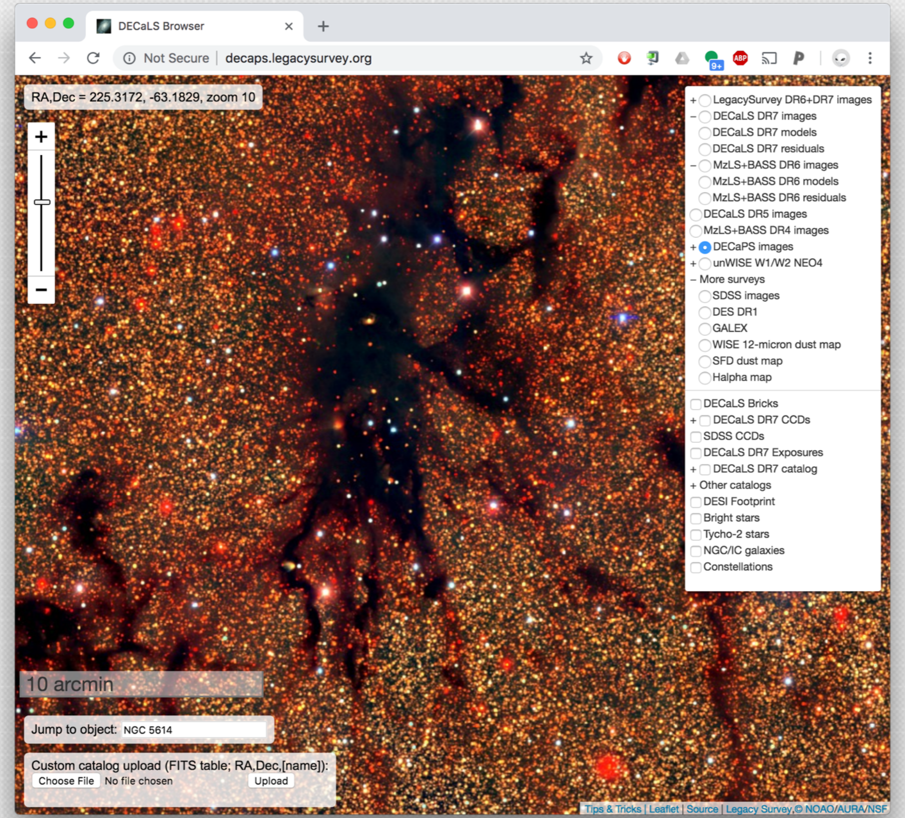
The Web Portal to the archive will enable browsing and visualization of the available datasets in ways the users are accustomed to at archives such as IRSA, MAST, or the SDSS archive, with an added level of interactivity.

Through the Portal, the users will be able to view the LSST images, request subsets of data (via simple forms or SQL queries), construct simple plots, and generally explore the LSST dataset in a way that allows them to identify and access (subsets of) data required by their science case.

This will all be backed by a petascale-capable RDBMS.

What to expect

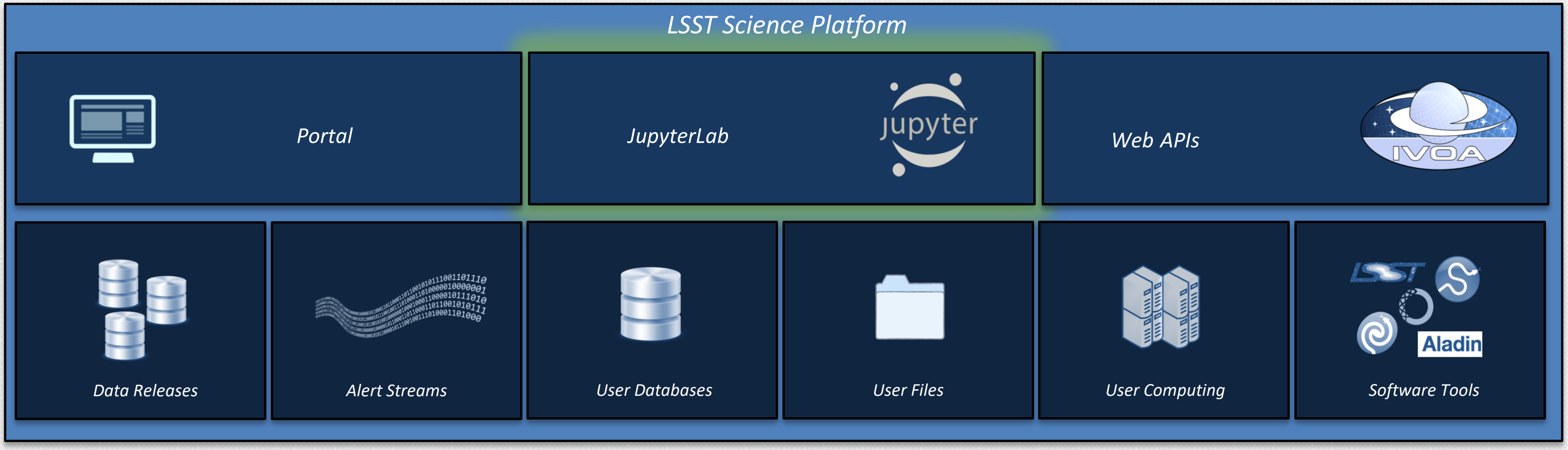




http://sky.esa.int/

http://decaps.legacysurvey.org/

JupyterLab: Next-to-the-data Analysis

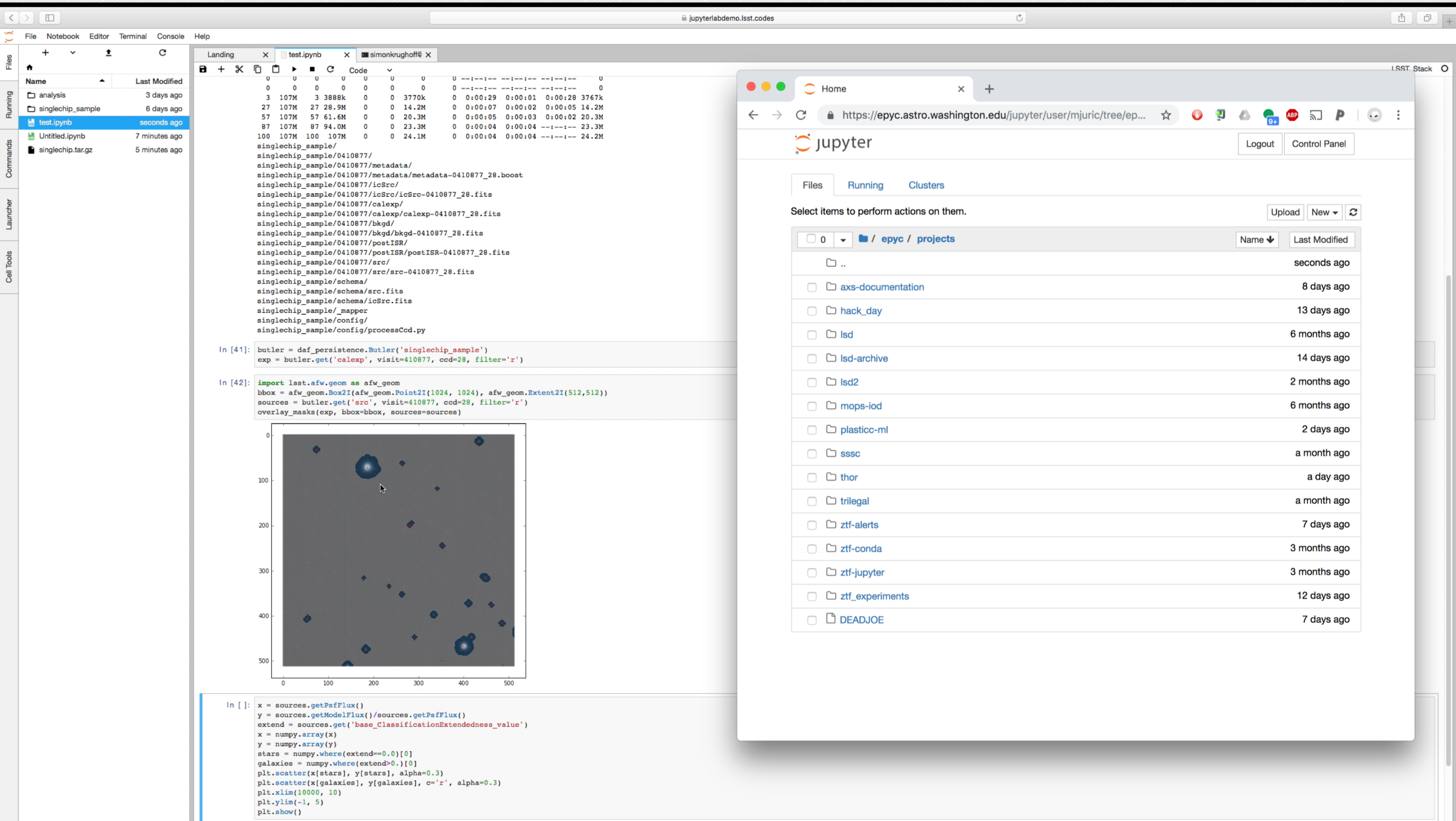


The tools exposed through the Web Portal will permit simple exploration, subsetting, and visualization LSST data. They may not, however, be suitable for more complex data selection or analysis tasks.

To enable that next level of next-to-the-data work, we plan to enable the users to launch their own Jupyter notebooks at our computing resources at the DAC. These will have fast access to the LSST database and files. They will come with commonly used and useful tools preinstalled (e.g., AstroPy, LSST data processing software stack).

This service is similar in nature to efforts such as SciServer at JHU.

JupyterLab: Next-to-the-data Analysis



YouTube demo of the LSST JupyterLab Aspect Demo: http://ls.st/bgt

Why Jupyter is data scientists'

https://www.nature.com/articles/d41586-018-07196-1

nature > toolbox > article

a natureresearch journal

nature
International journal of science

Search
E-alert
Submit
Login

TOOLBOX · 30 OCTOBER 2018

Why Jupyter is data scientists' computational notebook of choice

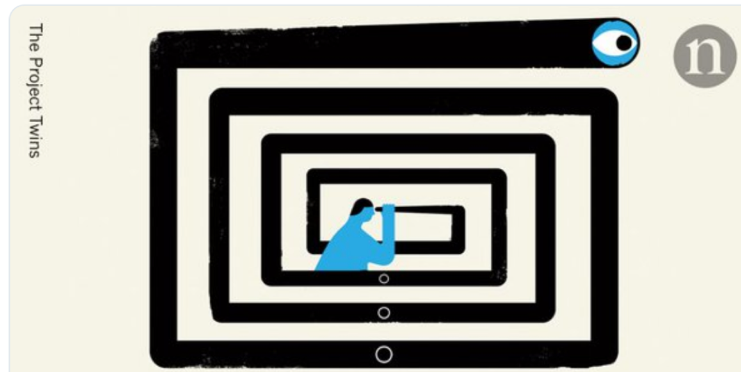
An improved architecture and enthusiastic user base are driving upta
open-source web tool.

Jeffrey M. Perkel

Nature Careers
@NatureCareers

Follow

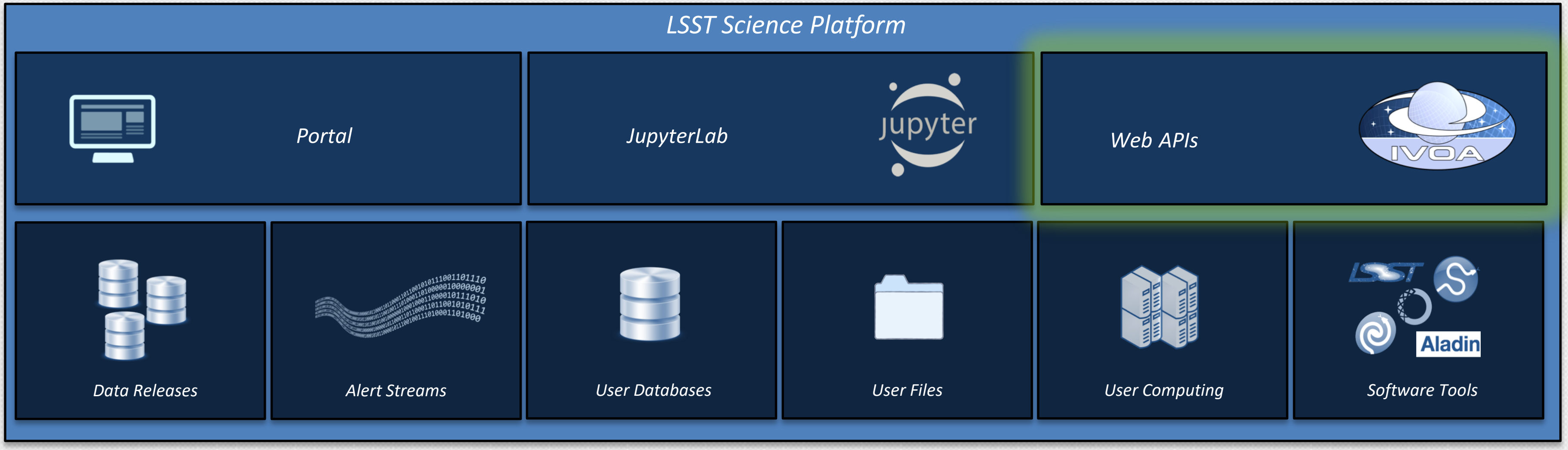
"I've never seen any migration this fast. It's just amazing." -- @mjuric on the rise of @ProjectJupyter in data science



Why Jupyter is data scientists' computational notebook of choice
An improved architecture and enthusiastic user base are driving uptake of the open-source web tool.
nature.com

10:30 PM - 5 Nov 2018

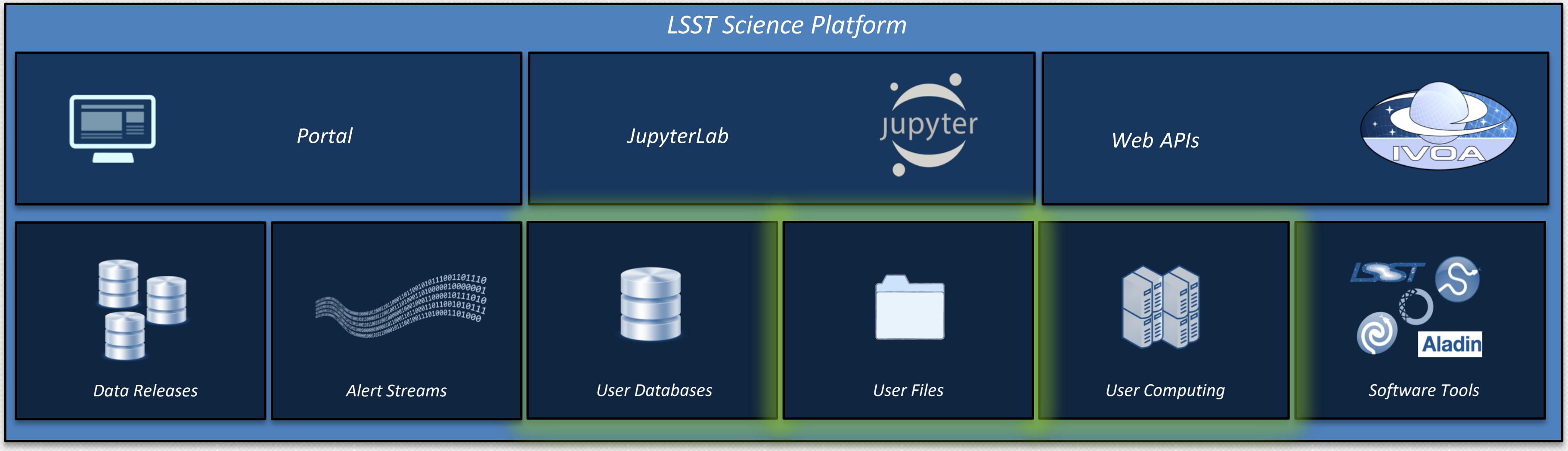
Web APIs: Integrating With Existing Tools



Backend Platform services – such as access to databases, images, and other files – will be exposed through machine-accessible web APIs.

We have a preference for industry standard and/or VO APIs (e.g., WebDAV, TAP, SIA, etc.) – the goal is to support what's broadly accepted within the community. This will allow the discoverability of LSST data products from within the Virtual Observatory, federation of the LSST data set to other archives, and the use of widely utilized tools (eg., TOPCAT or others).

Computing, Storage, and Database Resources



Computing, file storage, and personal databases (the *"user workspace"*) will be made available to support the work via the Portal and within the Notebooks.

An important feature is that no matter how the user accesses the DAC (Portal, Notebook, or VO APIs) they always "see" the same workspace.

How big is the “LSST Science Cloud” (@ DR2)?

- **Computing:**
  - ~2,400 cores
  - ~18 TFLOPs
- **File storage:**
  - ~4 PB
- **Database storage**
  - ~3 PB

This is shared by all users. We’re estimating the number of potential DAC users not to exceed 7500 (relevant for file and database storage).

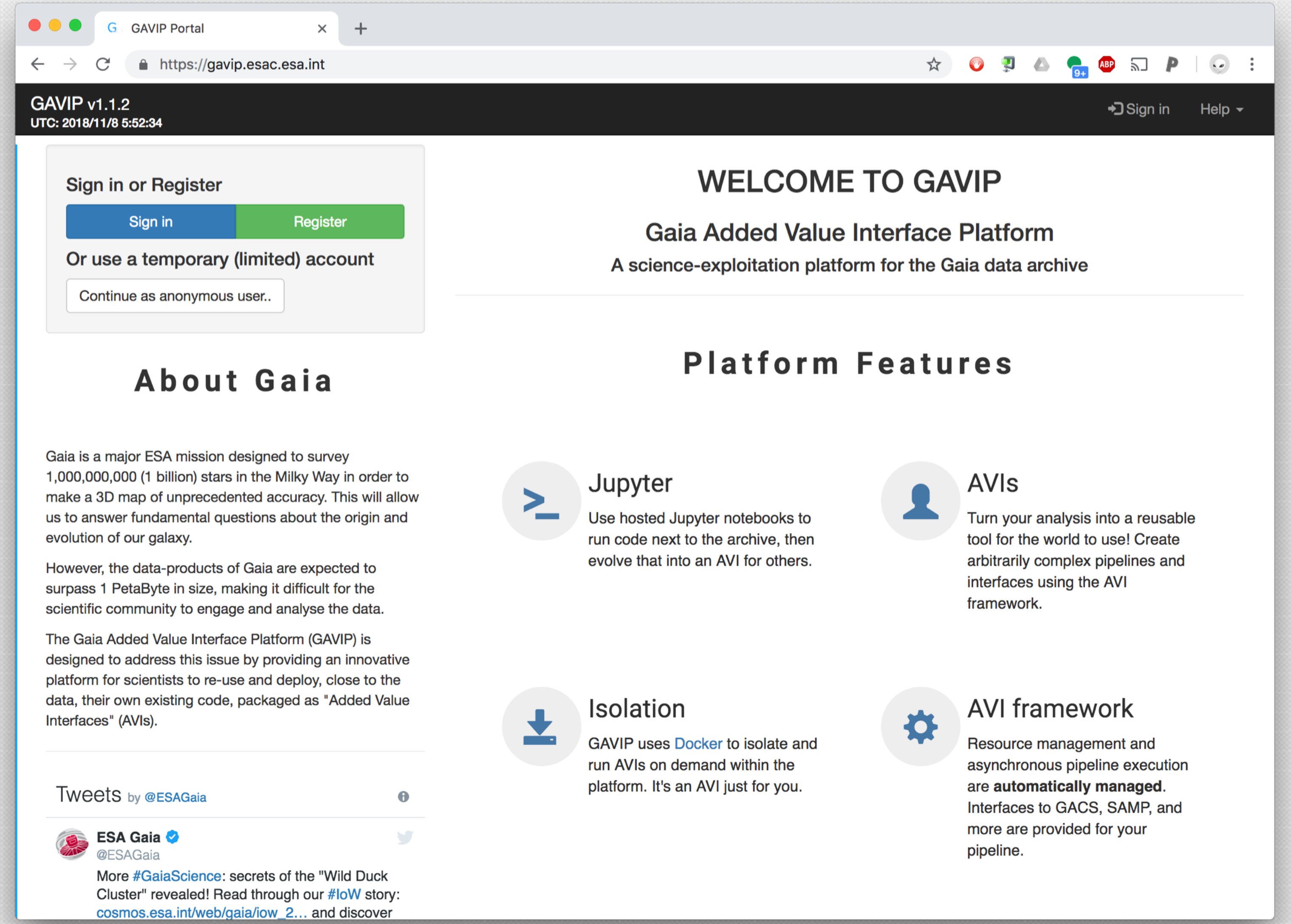
Not all users will be accessing the computing cluster concurrently. **We are estimating on order of a ~100.**

Though this is a relatively small cluster by 2020-era standards, it will be **sufficient to enable preliminary end-user science analyses** (working on catalogs, smaller number of images) and creation of some added-value (Level 3) data products.

Think of this as having your own server with a few TB of disk and database storage, right next to the LSST data, with a chance to use tens to hundreds of cores for analysis.

This kind of approach will become increasingly common for *all* big data archives.

Already Here for Gaia: GAVIP



Challenges (part 1)

Better Together

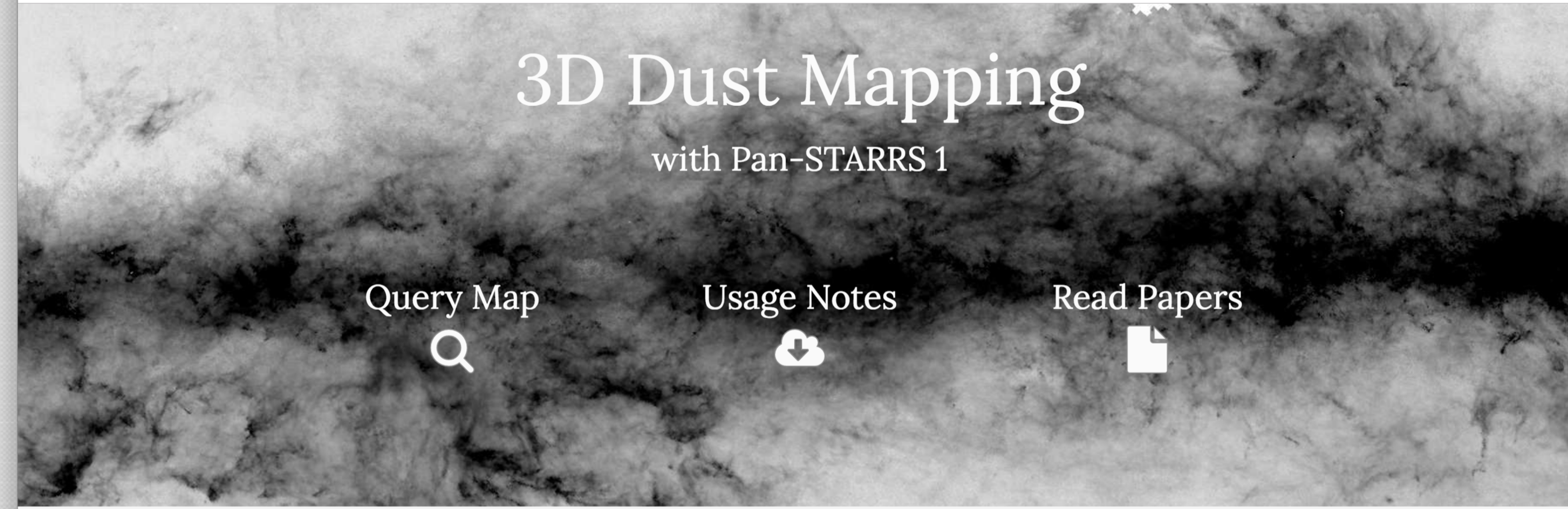
(joining datasets is powerful)

I Want it All

(science demands whole dataset operations)

3D Dust Mapping with Pan-STA ×

Not Secure | argonaut.skymaps.info



The Map

Interstellar dust attenuates ultraviolet, optical and near-infrared light. Because the extent of this attenuation is wavelength-dependent, dust both dims and reddens the light of stars and galaxies before it can reach our telescopes. In many areas of astrophysics, an accurate correction for the effects of interstellar extinction and reddening is critical. Historically, the most widely used maps of dust have been two-dimensional, tracing integrated dust reddening out to infinite distance. Here, we describe three-dimensional maps of interstellar dust reddening, which trace dust reddening both as a function of angular position on the sky and distance. These dust maps are based on Pan-STARRS 1 photometry of 800 million stars, along with 2MASS photometry of 200 million stars.

To read about how to download the map, or how to query it remotely, read our usage notes. To explore our map in the browser, see our interactive query page. To read in detail about our map, read our published papers.

Whole Dataset Operations

- Galactic structure: density/proper motion maps of the Galaxy
  - => forall stars, compute distance, bin, create 5D map
- Galactic structure: dust distribution
  - => forall stars, compute g-r color, bin, find blue tip edge, infer dust distribution
- Near-field cosmology: MW satellite searches
  - => forall stars, compute colors, convolve with spatial filters, report any satellite-like peaks
- Variability: Bayesian classification of transients and discovery of variables
  - => forall stars, get light curves, compute likelihoods, alert if interesting
- …

Challenges (part 2)

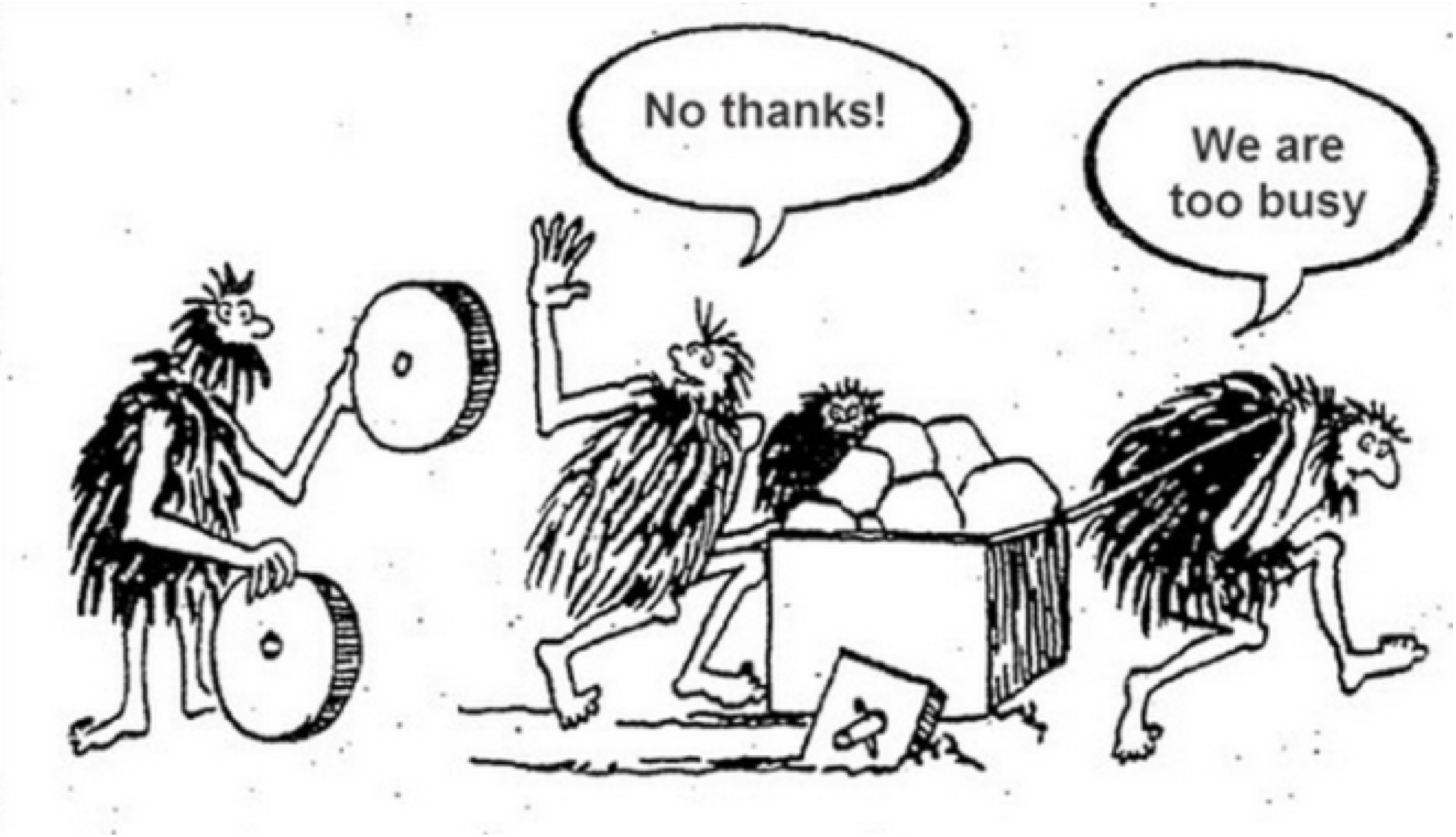
Scalability

(how do I write an analysis code that will scale to petabytes of data?)

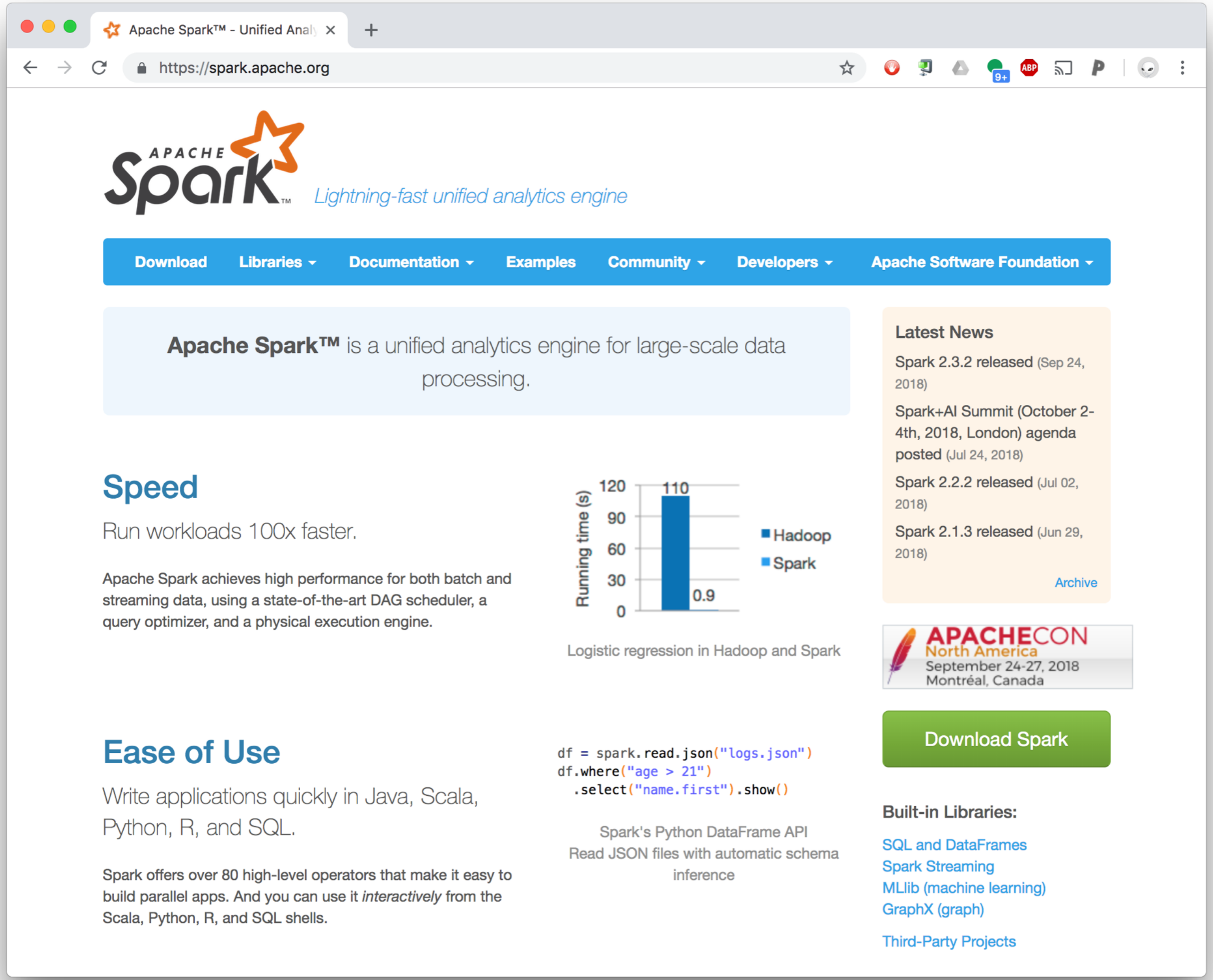
Resources

(where are the resources to run this code?)

Remember Yesterday…



Writing Scalable Applications: MapReduce and Apache Spark



Apache Spark is an open-source distributed general-purpose cluster-computing framework.

Spark provides an interface for programming entire clusters with implicit data parallelism and fault tolerance.

-- Wikipedia

Examples

Pi Estimation

Spark can also be used for compute-intensive tasks. This code estimates π by "throwing darts" at a circle. We pick random points in the unit square ((0, 0) to (1,1)) and see how many fall in the unit circle. The fraction should be π / 4, so we use this to get our estimate.

Python Scala Java

```
def inside(p):
    x, y = random.random(), random.random()
    return x*x + y*y < 1

count = sc.parallelize(xrange(0, NUM_SAMPLES)) \
             .filter(inside).count()
print "Pi is roughly %f" % (4.0 * count / NUM_SAMPLES)
```

https://spark.apache.org/examples.html

Scalability through MapReduce

Dean & Ghemawat (2004)

Map $\{x_i\}$ ---map--> $\{y_i=f(x_i)\}$ Apply a function *f* to every element of dataset *X*, producing dataset *Y*

Reduce $\{ (k_i, v_{ij}) \} \rightarrow \{ y_i=(k_i,f(\{v_{ij}\})) \}$ Apply a function *f* to all values with a common key

Example:

{ (“dog”, 2), (“dog”, 1), (“cat”, 3), (“dog”, 2), (“cat”, 1) }

-> reduce w. *sum()* ->

{ (“dog”, 5), (“cat”, 4) }

Examples

{ (“dog”, 2), (“dog”, 1), (“cat”, 3), (“dog”, 2), (“cat”, 1) }

-> reduce w. *sum()* ->

{ (“dog”, 5), (“cat”, 4) }

Word Count

In this example, we use a few transformations to build a dataset of (String, Int) pairs called `counts` and then save it to a file.

Python Scala Java

```
text_file = sc.textFile("hdfs://...")
counts = text_file.flatMap(lambda line: line.split(" ")) \
             .map(lambda word: (word, 1)) \
             .reduceByKey(lambda a, b: a + b)
counts.saveAsTextFile("hdfs://...")
```

https://spark.apache.org/examples.html

Astronomy Example: Compute Light Curve Features

This works on arbitrarily large datasets!

In [10]:
```
from pyspark.sql.types import ArrayType, FloatType, DoubleType
from pyspark.sql.functions import col, pandas_udf, explode
import pandas as pd

import cesium
from cesium.time_series import TimeSeries
from cesium.featurize import featurize_single_ts, featurize_time_series

##########

features_to_use = ["amplitude", "percent_beyond_1_std", "maximum", "max_slope",
                   "median", "median_absolute_deviation", "percent_close_to_median",
                   "minimum", "skew", "std", "weighted_average"]

ls_features = ["freq1_amplitude1", "freq1_amplitude2", "freq1_amplitude3",
               "freq1_amplitude4", "freq1_freq", "freq1_lambda", "freq1_rel_phase2",
               "freq1_rel_phase3", "freq1_rel_phase4", "freq1_signif", "freq2_amplitude1",
               "freq2_amplitude2", "freq2_amplitude3", "freq2_amplitude4", "freq2_freq",
               "freq2_rel_phase2", "freq2_rel_phase3", "freq2_rel_phase4"]

def featurize_udf(mjd, psfflux):
    feat_outs = []
    for row_mjd, row_psfflux in zip(mjd, psfflux):
        feat_out = featurize_time_series(np.array(row_mjd), np.array(row_psfflux),
                                         features_to_use=features_to_use + ls_features)
        feat_outs.append(feat_out.values.flatten())
    return pd.Series(feat_outs)

##########

feat_udf = pandas_udf(featurize_udf, returnType = ArrayType(DoubleType()))
spark_session.udf.register("FEATURIZE", feat_udf)

pdf = ztf.where("SIZE(mjd)>50").selectExpr("FEATURIZE(mjd, psfflux)").toPandas()
```

Cesium (Naul, 2016), Astronomy eXtensions for Spark (Zecevic+ 2018)

The Result (with apologies for the appallingly poor visualization)

```
In [12]: pdf = ztf.where("SIZE(mjd)>50").limit(10).selectExpr("ADDMJ(mjd, psfflux)").toPandas()

Out[12]: [Row(ADDMJ(mjd, psfflux)=[1925.2211608886719, 0.13978494623655913, 3987.0869140625, None, 344
6.4375, 152.688720703125, 0.7419354838709677, 136.64459228515625, -2.4431908318547433, 631.64
34156713688, 3189.848529118364, 313.63354429378256, 14.639380553238073, 1.3644581456964708,
1.7500900946095723, 1.5124216699725148, 42.94394862399773, 2.5313200999890264, 0.635479246010
1448, -0.5061265295340045, 5.0589783305487925, 297.2971196481153, 49.95079199367568, 5.377033
053881004, 2.9892154859975197, 4.01272141380738, -0.11956697923508663, 0.44717905887839726, -
1.1813749683927528]),
 Row(ADDMJ(mjd, psfflux)=[208.086181640625, 0.17857142857142858, 848.4930419921875, 57922.265
60191954, 561.7611083984375, 21.10723876953125, 0.7142857142857143, 432.3206787109375, 1.7943
474250678484, 58.001638792839145, 561.3877334594727, 41.09281808433971, 3.6566340247461695,
0.484343106761218, 0.3561043472427892, 1.9945387332853752, 36.82950672444141, -2.622502373517
709, 1.901872187975775, 1.3933600656842617, 3.208573392947105, 26.071430553383156, 1.26390776
08450804, 0.47282767919988666, 0.11065610514135481, 23.0800641543819, -2.85326837788731, 1.89
44949539332152, -0.3698982601601857]),
 Row(ADDMJ(mjd, psfflux)=[491.52618408203125, 0.1320754716981132, 1122.197509765625, 32600.79
9800087658, 402.0052490234375, 31.34088134765625, 0.7924528301886793, 139.1451416015625, 2.88
3311815955483, 131.10471926999602, 416.95930855229216, 114.34083688267272, 10.76770241852664
3, 4.481966442155684, 1.2482773020089568, 1.0816044062094867, 24.405764071747054, -1.35716539
37044816, -2.4773879544783286, 2.77126228292522, 3.23567097891298, 60.04996221378995, 5.37096
0993975372, 0.8751468632528988, 0.3325519757277276, 0.6780459096769216, -0.8609512237261738,
-1.5228145978972758, 1.6412947076259528]),
 Row(ADDMJ(mjd, psfflux)=[107.95289611816406, 0.17333333333333334, 303.4608154296875, 15769.0
08109654327, 146.07757568359375, 16.181289672851562, 0.64, 87.55502319335938, 1.9784213046179
848, 33.608624825862975, 152.7379244995117, 14.807230580579672, 0.2625194273012284, 0.1712371
6298087752, 0.1045290010068471, 29.250142510335483, 75.41574345145936, -1.005145004902362, 2.
1760284474899296, 1.1339028285326511, 2.980258224087021, 18.704151766762745, 1.06178361034372
75, 0.283697089687251, 0.18280423388998288, 0.965224836036898, 0.24573780352703498, -1.282736
1427620787, -2.016597738480162]),
 Row(ADDMJ(mjd, psfflux)=[1531.0211791992188, 0.14285714285714285, 4344.10595703125, None, 39
23.2584228515625, 160.582275390625, 0.6587301587301587, 1282.0635986328125, -2.22651166992487
46, 582.6092968330478, 3688.5484958224824, 387.9816297071594, 48.53561353536267, 12.997457710
221171, 6.372125562326946, 3.0098298953254314, 13.648208330625398, 0.4783521707732701, -0.520
8550287735074, 0.9412748273621686, 6.5045045280108695, 236.76237021479648, 3.799074459364612,
```

Want to try it out?

```
conda install -c conda-forge pyspark
```

Scaling with Spark

Today, Spark is being adopted by major players like Amazon, eBay, and Yahoo! Many organizations run Spark on clusters with thousands of nodes. According to the Spark FAQ, the largest known cluster has over 8000 nodes. Indeed, Spark is a technology well worth taking note of and learning about.

But where do I find 8000 nodes?

https://www.toptal.com/spark/introduction-to-apache-spark

Cloud services

- Essentially, companies who rent computers (or a few million of them)
  - The same for storage
- Pay only for what you use (by the second/minute/hour)
- Scalable: ask for 1000 machines, get a 1000 machines
- Becoming cost effective (TCO)
  - Especially “spot” pricing

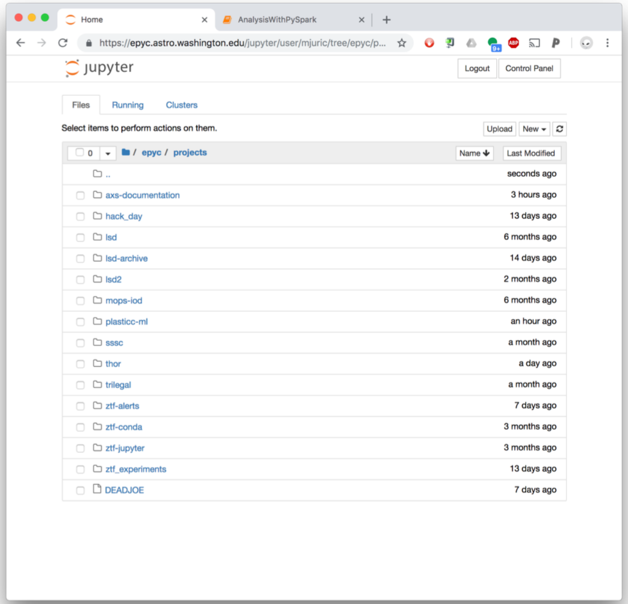
Meeting the Challenges

Dataset Storage

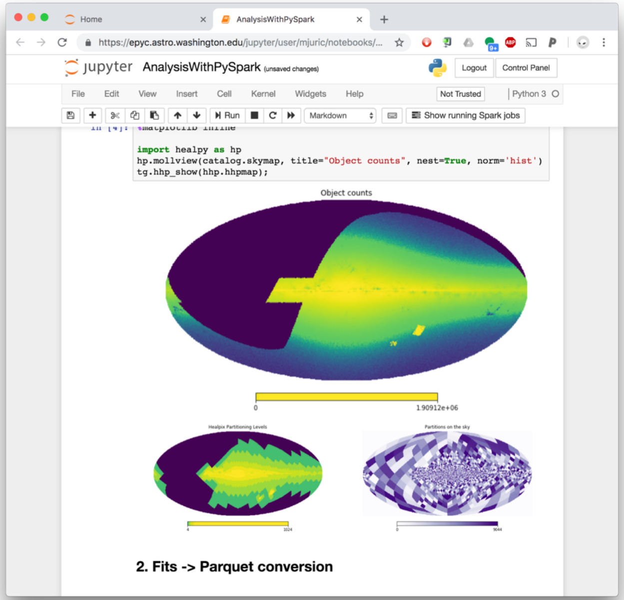
Resources

Scalable Analysis Code

Interface



"Analysis 2025"



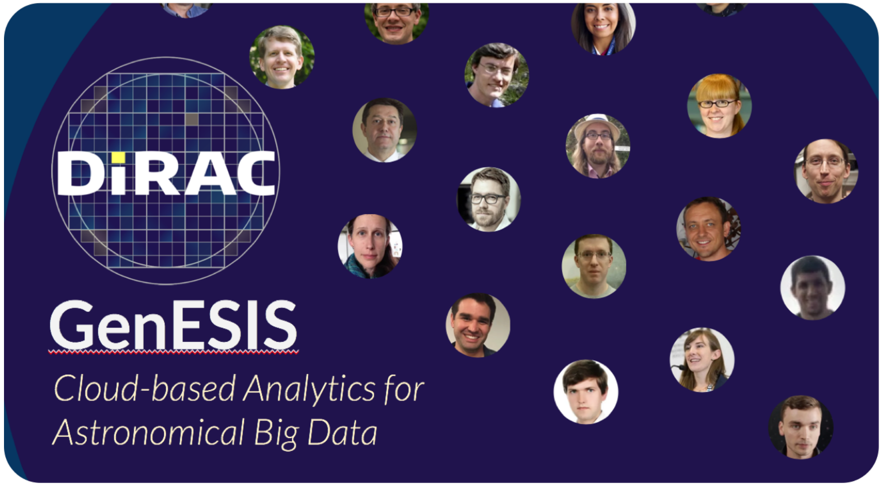
A Number of Projects are Working to Make this Happen

PANGEO

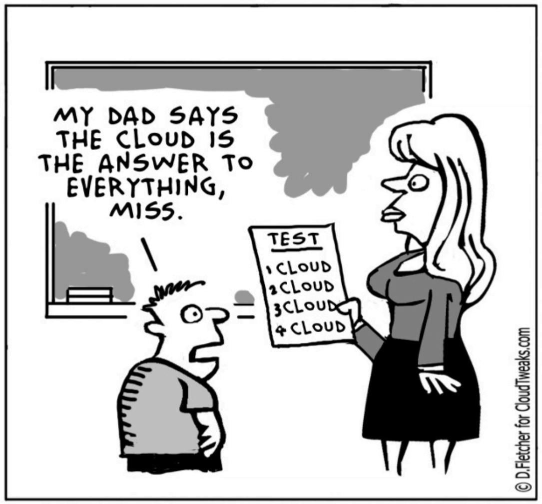
A community platform for Big Data geoscience

http://pangeo.io/

Coming soon w. ZTF !



Some Words of Caution



Just like with machine learning / A.I., there's no need to throw cloud at everything.

Small datasets?
Large-ish datasets?

But the *programming model* works across all scales.

"What if we don't change at all ... and something magical just happens?"

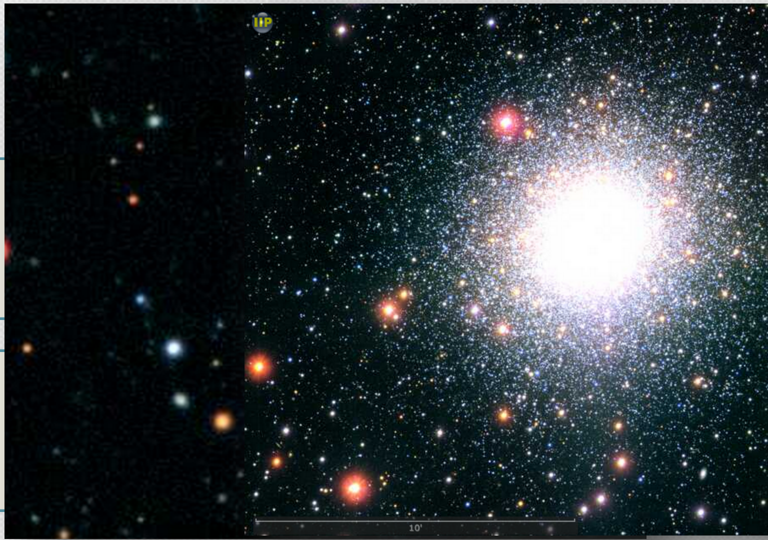
The implementation of these technologies is still in its infancy. They change incredibly quickly.

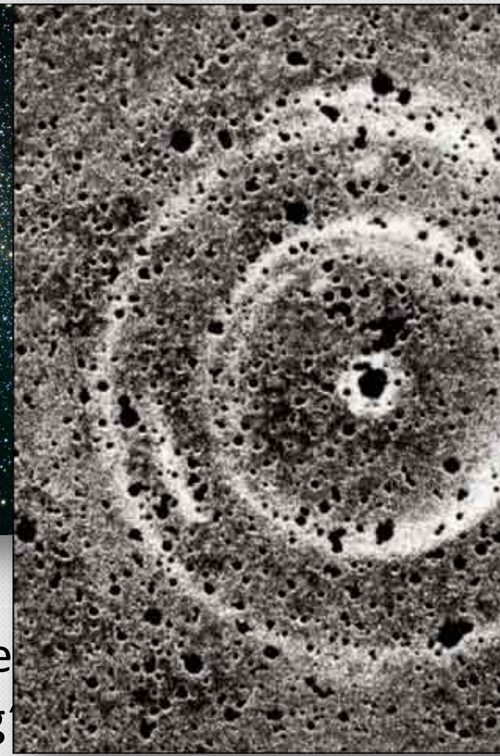
Expect you may need to shift from framework to framework (e.g., Spark → Dask).

That said, the *programming models* change on a much longer timescale (e.g., MR 2004 ->).

Looking Ahead: Leave no Information Behind

(or why software and services are even more important than we think)





- As our measurements become
  occurs in the “Data Processing
- Sometimes, an assumption or an algorithmic choice th been made th
  may introduce a systematic that drowns out the signal (or eliminates it).
- For optimal inference, one wants to design measurements that directly probe the relevant aspects of the *original (imaging data)*, and not the (lossy-compressed) catalog.
  - Or derive more appropriate catalogs/feature sets/etc.

Pushing the Boundaries of Optimal Inference

Model ← *inference* – **Data**

Model ← *inference* – **Catalog** ← Data Processing – **Data**

- **Reasons we don't do this today:**
  1. Computationally (and I/O) intensive
  2. Sociologically difficult
     - Expertize in statistics, applied math, and software engineering is often not there
     - Catalogs are too often taken as "God given", fundamental, result of a survey

- **Things are changing**
  - Big data problems are becoming computationally tractable (see prev. discussion)
  - Average astronomer in the 2020s will grow up with an expectation of being well versed in Stats, SE, Appl. Math.
  - A concerted effort is under way, primarily driven by people in large survey and telescope projects, to create the necessary software to make this possible.

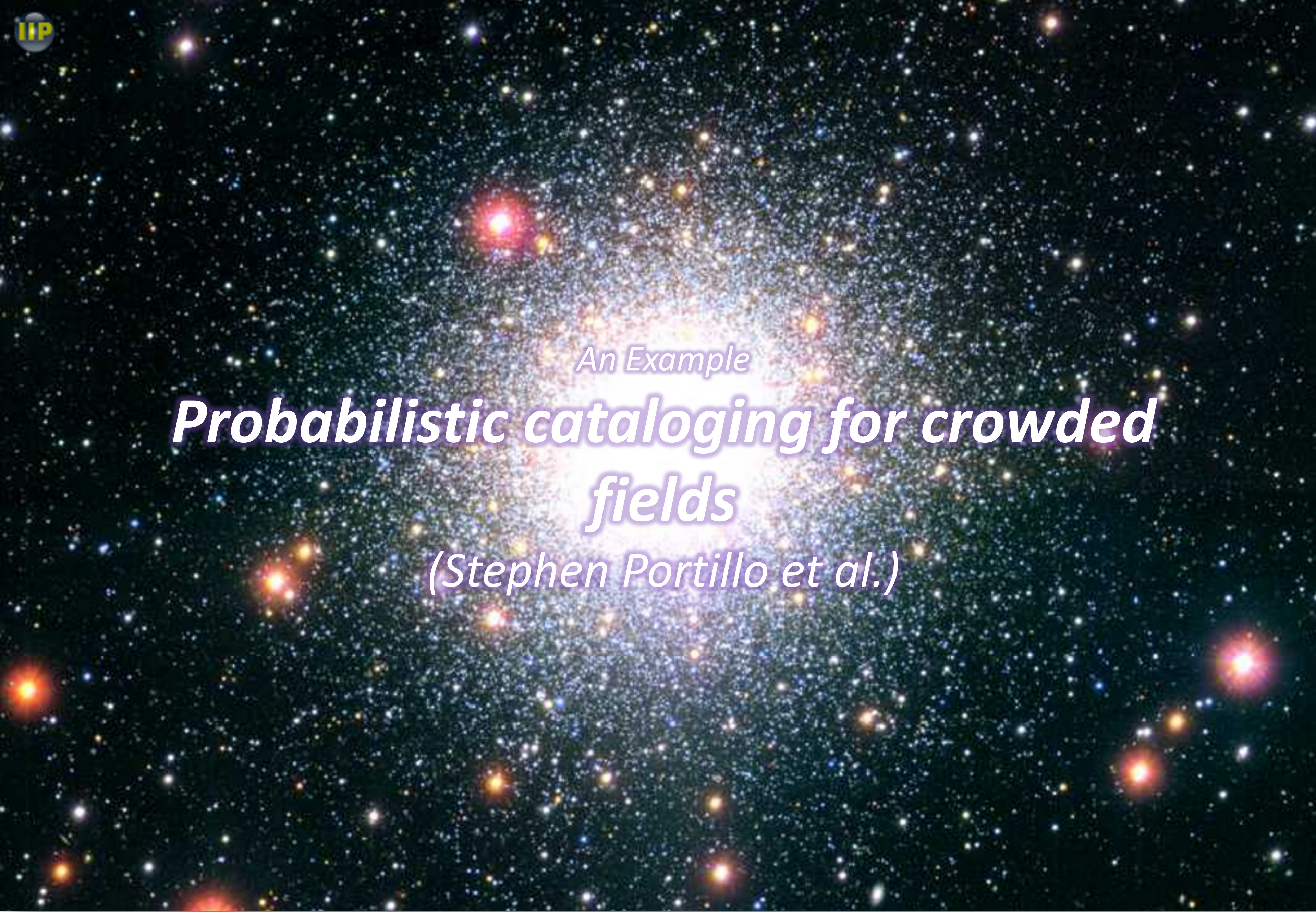
Astronomy 2025: “Personalized Medicine”

- In the next decade, it may be possible for any one of you to re-reduce large datasets for optimally your science case.

- You will be able to do this because the software building blocks (AstroPy, LSST stack, etc.) will be there, with frameworks and cloud resources for large-scale computation.

- Right now, we see the data releases as the key product of a survey. By the end of the next decade, I wouldn’t be surprised if we saw **the software as the key product**, with hundreds specialized (and likely ephemeral) catalogs being generated by it.

- The official “data releases” will just be some of those catalogs, designed to be more broadly useful than others, and retained for a longer period of time.



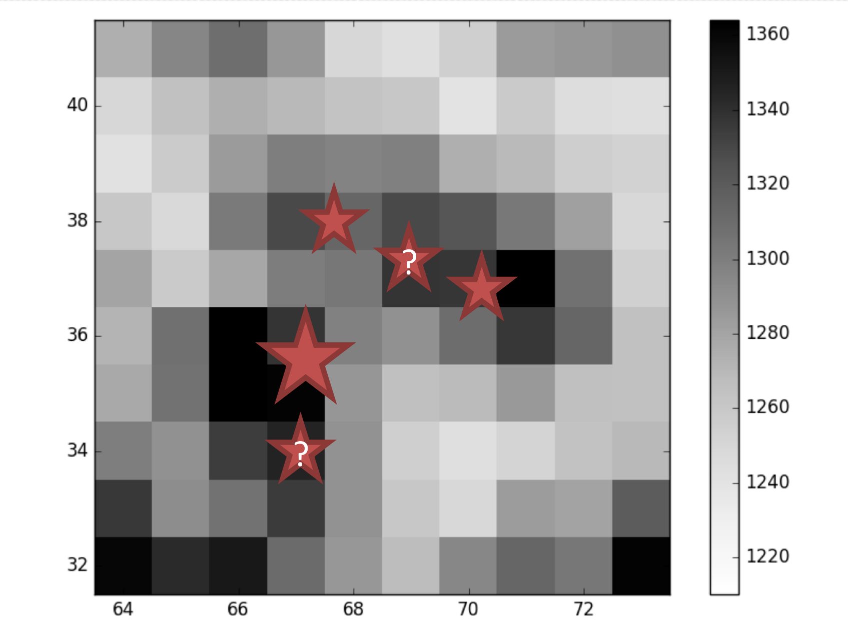
An Example

Probabilistic cataloging for crowded fields

(Stephen Portillo et al.)

Crowded Field Cataloguing

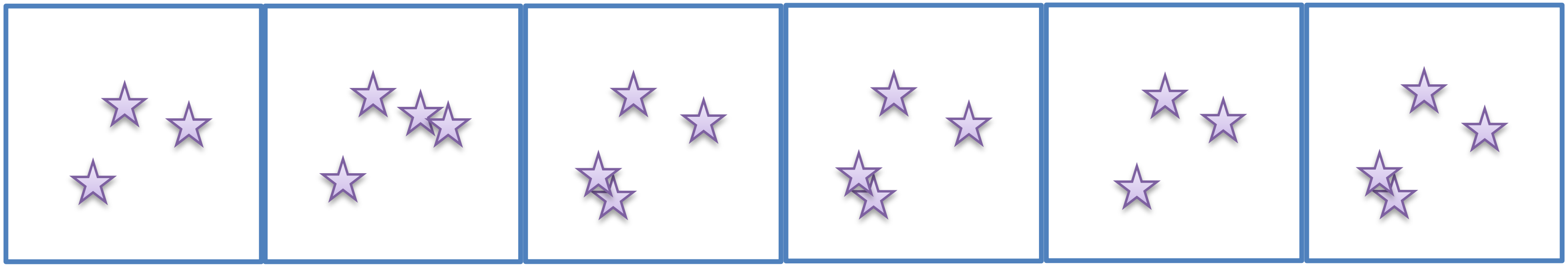
- Neighbouring sources are covariant
- Deblending can be difficult or even ambiguous
- The inferred properties depend on how the image is deblended



SDSS DR14

Probabilistic Cataloguing

- Instead of having one catalog, produce an ***ensemble of catalogs*** (each with an associated probability of occurrence)
- Naturally handles deblending ambiguities and source-source covariance in crowded fields

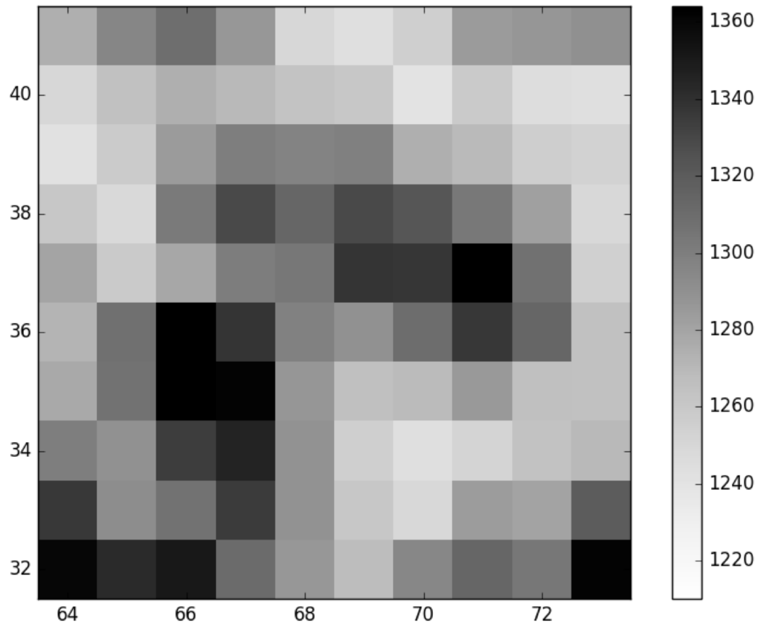


- Space of possible catalogues is ***transdimensional***

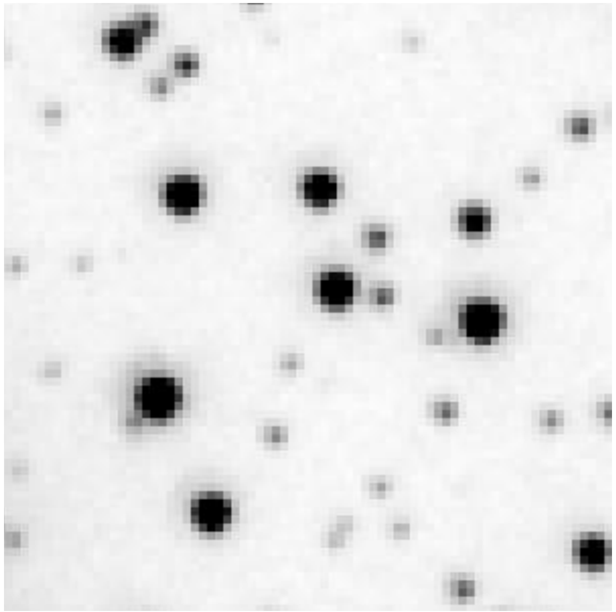
Brewer, Foreman-Mackey, and Hogg (2013)

Application: Deblending the Cluster M2

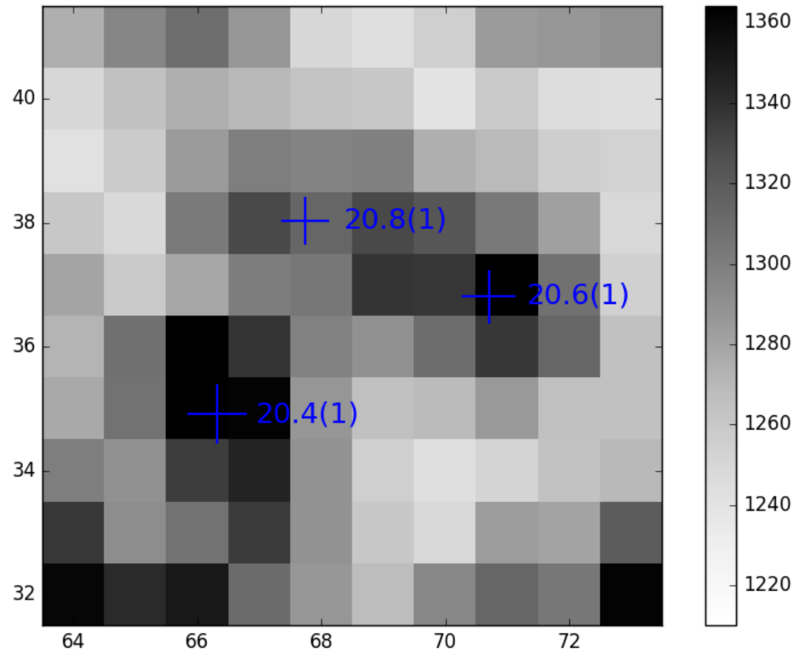
- Sloan Digital Sky Survey



- Hubble Space Telescope



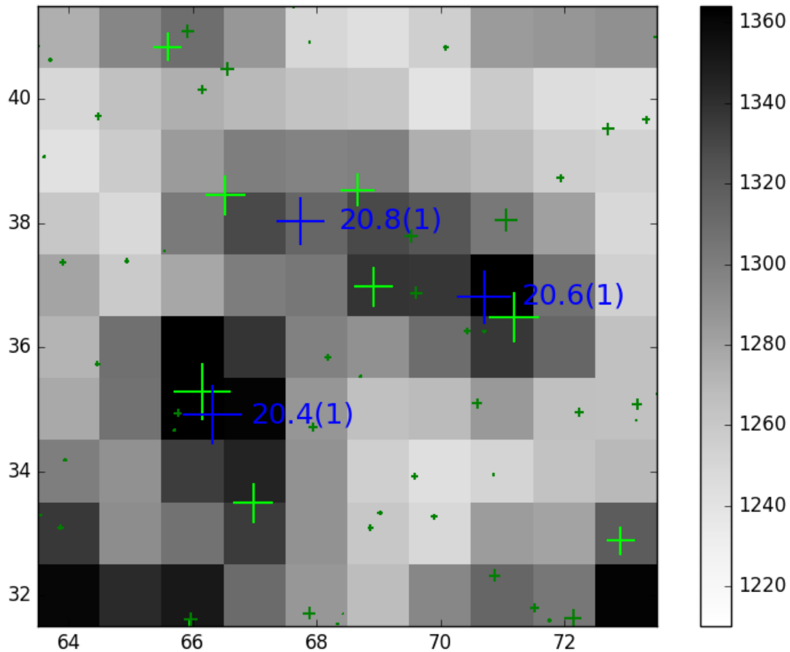
Traditional Catalogue



DAOPhot

SDSS DR14
An et al. (2008)

Compared to Hubble

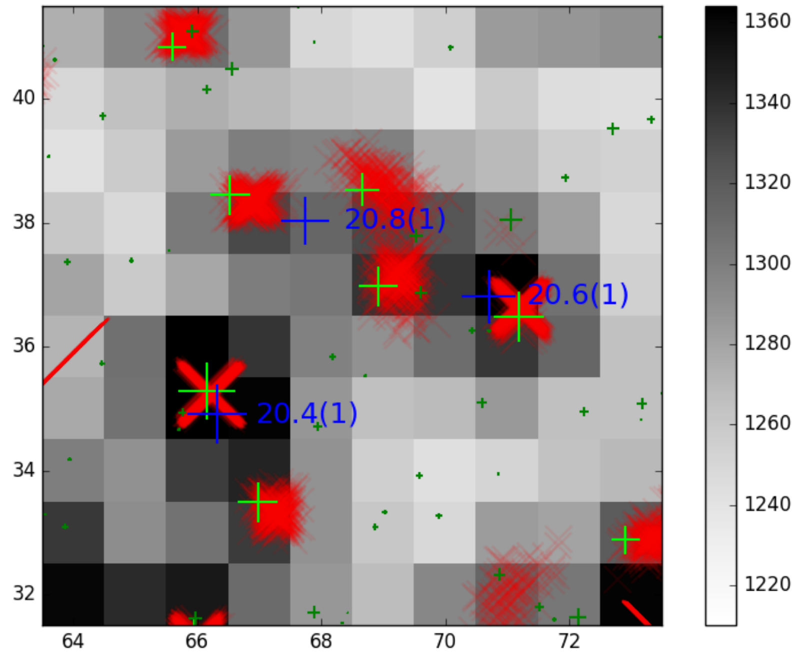


DAOPhot

HST

SDSS DR14
An et al. (2008)
Sarajedini et al. (2007)

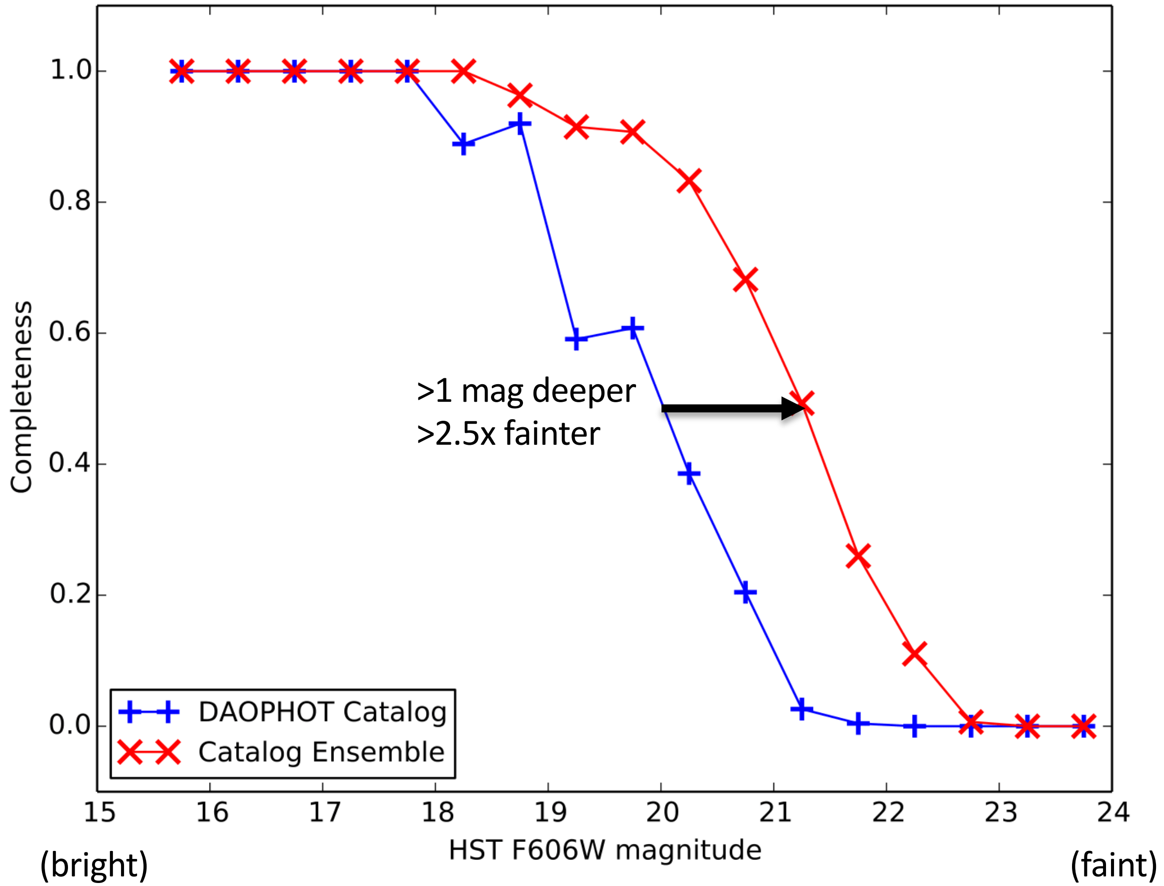
Stacked Catalogue Ensemble



DAOPhot
HST
PCat

Completeness

$$\frac{\text{true positives}}{\text{all real sources}}$$



Where is all this going

- The data is big, but not unmanageable. But technologies exist (in the industry) to meet the challenges.

- Two changes in paradigms:
  - New programming models (and frameworks): MapReduce (Spark)
  - Analysis on cloud services, rather than on local machines

- This is an ***opportunity***: we'll soon be able to take data analysis one level closer to the *images* (and therefore extract more data). Or devise custom, complex, analyses over entire datasets.
  - E.g., crowded field codes.

- Adding ML/AI in the mix, sky's the limit….

Machine learning	Statistics
network, graphs	model
weights	parameters
learning	fitting
generalization	test set performance
supervised learning	regression/classification
unsupervised learning	density estimation, clustering
large grant = $1,000,000	large grant = $50,000
nice place to have a meeting: Snowbird, Utah, French Alps, Tenerife	nice place to have a meeting: Las Vegas in August

Some important differences between machine learning and statistics.