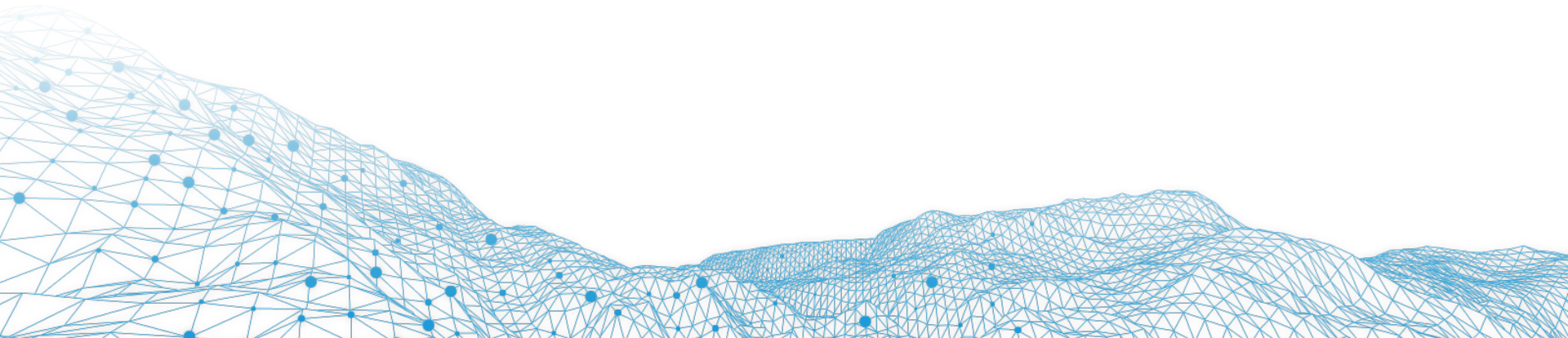
Zeppelin

Interactive Analytics



Overview

- Data
- Analytics
- Interactivity
- Next

Data

Laboratory for Analytic Sciences

National Center for Advancing Translational Sciences

UNC ESHELMAN SCHOOL OF PHARMACY

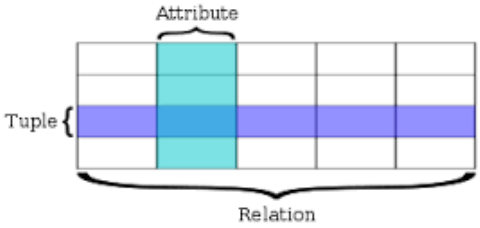
TB → PB

EVRYSCOPE

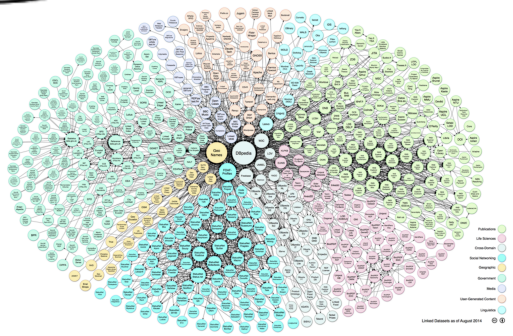
What Data?

Unstructured

Tabular



Graph



Streaming

Analytics

What happens to the data in

- General?
- Particular?
- Interactive collaboration?

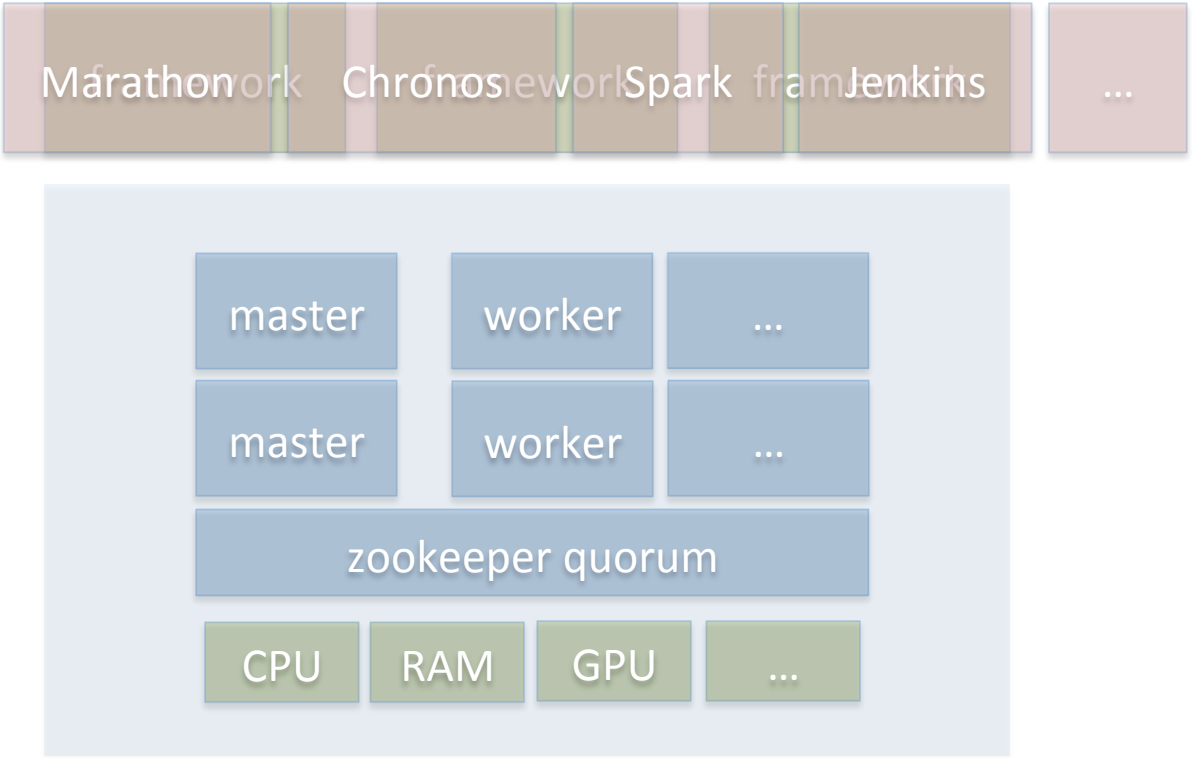
Analytics : In General

What happens to the data?

The same thing that happens to everything else…

Parallelize!

Analytics: MESOS



Analytics : In Particular

What happens?

- **Mapreduce**
- **SQL**
- **GraphFrames**
- **SPARQL**
- **Streaming**
- **And**: Python, Scala, Java, R

Interactivity

Old School:

- **CLI/Interpreter**: Scala or Python
- **Scripts**: Run from the shell
- Nothing wrong with this

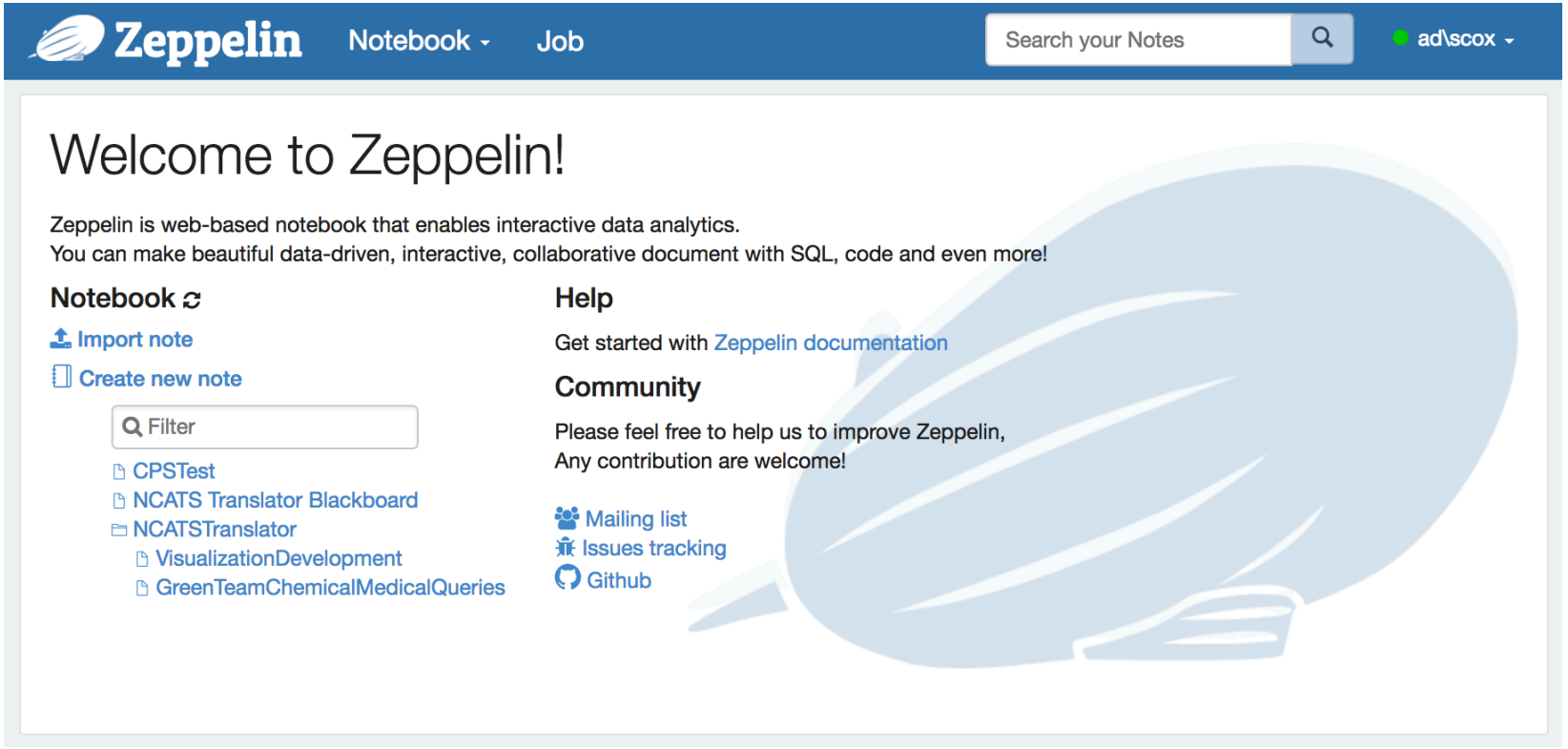
But scalable interactive collaboration is nice too

Interactivity: Notebooks

Rise of the Notebooks:

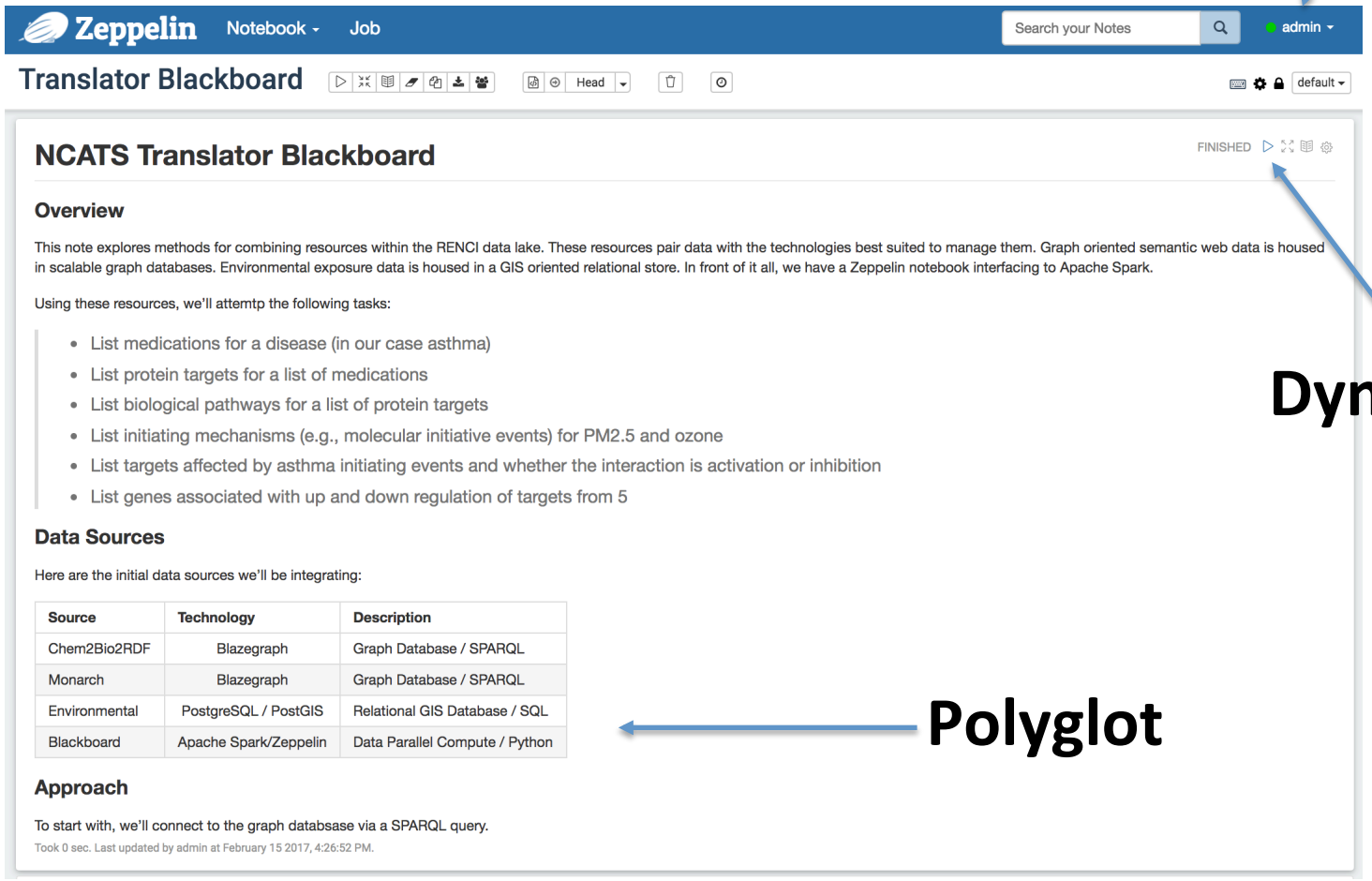
- **Dynamic**: Interpreted languages
- **Collaborate**: Authenticated, authorized
- **Scale**: Via Spark, Mesos, Blazegraph, …
- **Polyglot**: Multiple languages living together
- **Publish**: Reuse notes in other contexts

Interactivity: Zeppelin



Zeppelin

Authenticated



Dynamic

Polyglot

Zeppelin

This note is strictly setup

FINISHED

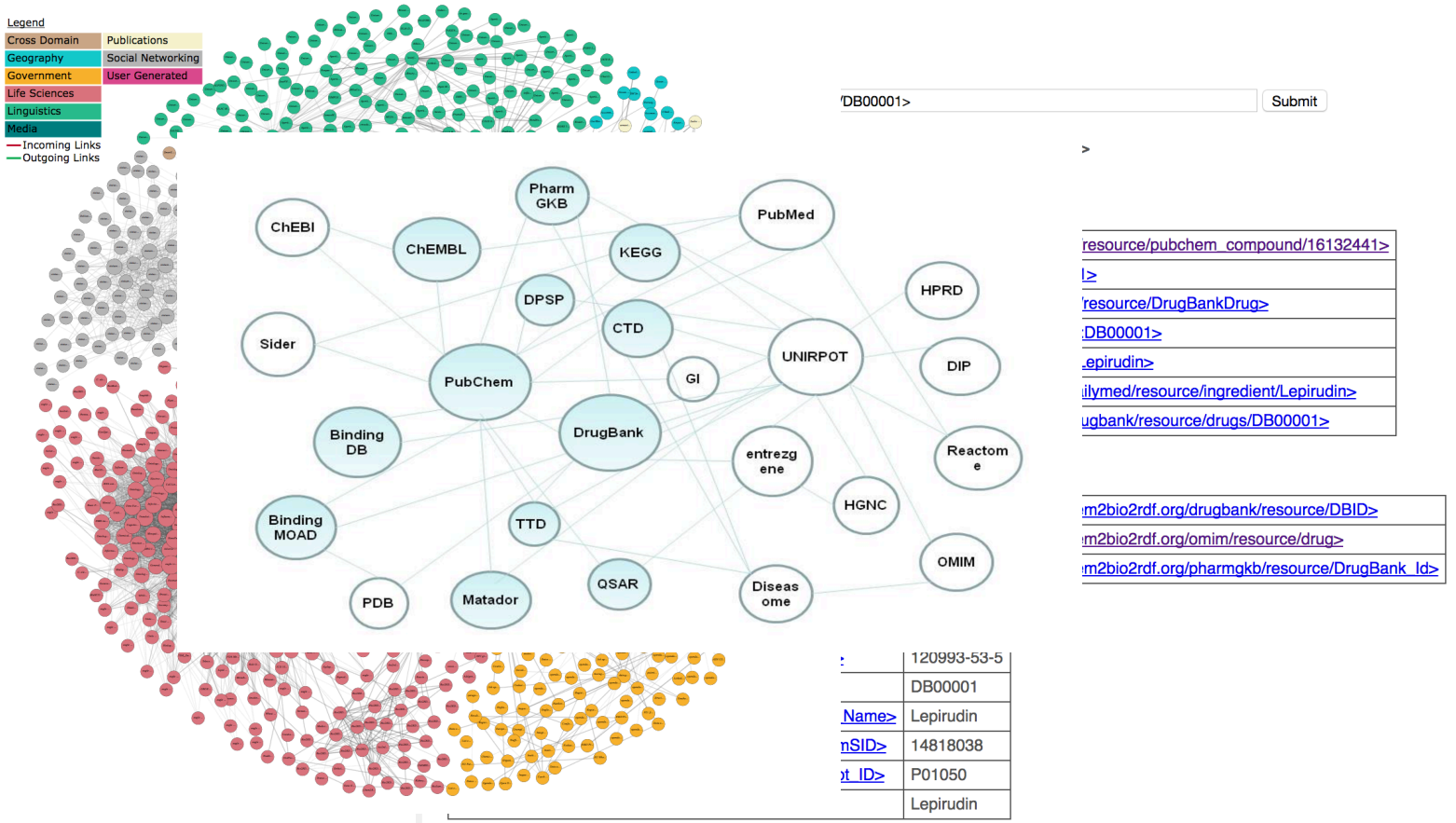
```
%pyspark
from string import Template
from collections import defaultdict
from SPARQLWrapper import SPARQLWrapper, JSON
import matplotlib.pyplot as plt
import seaborn as sns

blazegraph_uri = "http://stars-blazegraph.renci.org/bigdata/sparql"
sparql = SPARQLWrapper(blazegraph_uri)
def query_blaze (query):
    sparql.setQuery (query)
    sparql.setReturnFormat (JSON)
    return sparql.query().convert ()
```

Took 0 sec. Last updated by admin at February 19 2017, 3:55:25 PM. (outdated)

Zeppelin

Semantic Web, chem2bio2rdf, and Blazegraph:



Zeppelin

Find asthma drugs:

```
%pyspark
query = Template ("""
PREFIX db_resource: <http://chem2bio2rdf.org/drugbank/resource/>
PREFIX omim:        <http://chem2bio2rdf.org/omim/resource/>
SELECT DISTINCT ?disease ?generic_name
WHERE {
  ?drug        db_resource:Generic_Name ?generic_name .
  ?disease_rec omim:drug                ?drug ;
               omim:Name                ?disease .
  FILTER regex(?disease, "${disease}", "i")
}""")
results = query_blaze (query.substitute (disease="asthma"))
drugs = []
for r in results["results"]["bindings"]:
    disease_name = r['disease']['value']
    drug_name = r['generic_name']['value']
    if not drug_name in drugs:
        drugs.append (drug_name)
print ("drugs: {0}".format (drugs))
```

```
drugs: [u'Dyphylline', u'Hydrocortisone', u'Salmeterol', u'Formoterol', u'Theophylline', u'Bec
 u'Fluticasone Propionate', u'Omalizumab', u'Isoetharine', u'Nedocromil', u'Zileuton', u'Momet
```

Took 0 sec. Last updated by admin at February 19 2017, 12:44:29 PM.

Definitions

Query

Parse

Output

Zeppelin

Graph protein targets by drug with Seaborn:

2. List of Gene Targets for a set of drugs:

FINISHED

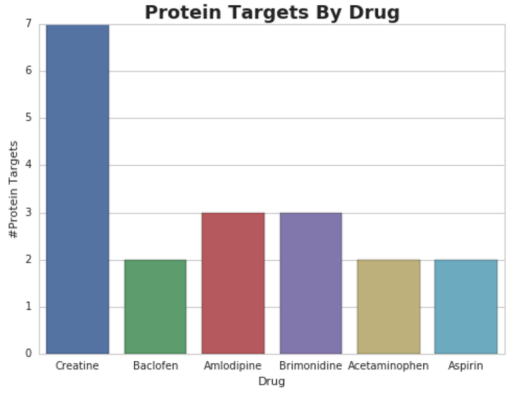
This is a count of protein targets by drug. It's generated by a Zeppelin notebook running on RENCI's Stars big data infrastructure.
The Zeppelin notebook is connected to an Apache Spark Python interpreter running on the Stars distributed Mesos compute fabric.

This text is generated by a Zeppelin markdown note. The graphic below is generated by a second Zeppelin PySpark note. The PySpark note creates a SPARQL query which it executes on a Blazegraph graph database instance.

The Blazegraph instance contains several Semantic Web artifacts from the Chem2Bio2RDF project.
Protein targets associated with each drug are counted and rendered in the following histogram using Seaborn and Matplotlib.

Took 0 sec. Last updated by admin at February 16 2017, 10:34:03 PM.

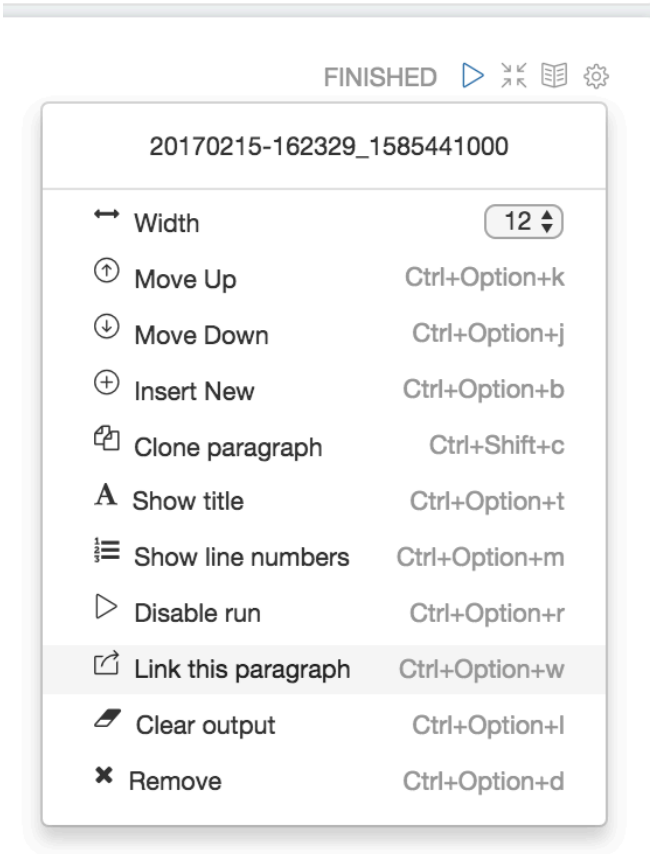
```
%pyspark
query = Template ("""
PREFIX db_resource: <http://chem2bio2rdf.org/drugbank/resource/>
SELECT ?generic_name ?protein
WHERE {
  ?interaction db_resource:SwissProt_ID ?protein .
  ?interaction db_resource:DBID         ?drug .
  ?drug        db_resource:Generic_Name ?generic_name .
  FILTER regex(?generic_name, "${drugs}", "i")
}""")
drug_list = "Creatine|Baclofen|Amlodipine|Brimonidine|Acetaminophen|Aspirin"
results = query_blaze ( query.substitute (drugs=drug_list))
drug_protein = defaultdict(list)
protein_drug = defaultdict(list)
for r in results["results"]["bindings"]:
    drug = r['generic_name']['value']
    protein = r['protein']['value'].rsplit('/', 1)[-1]
    if protein != 'Not_available':
        drug_protein[drug].append (protein)
        protein_drug[protein].append (drug)
x = drug_list.split ("|")
data={ 'x' : x, 'y' : map (lambda k : len(drug_protein[k]), x) }
sns.set_style("whitegrid")
sns.plt.title('Protein Targets By Drug', weight='bold').set_fontsize('18')
ax = sns.barplot(x="x", y="y", data=data). \
    set(xlabel='Drug', ylabel='#Protein Targets')
```

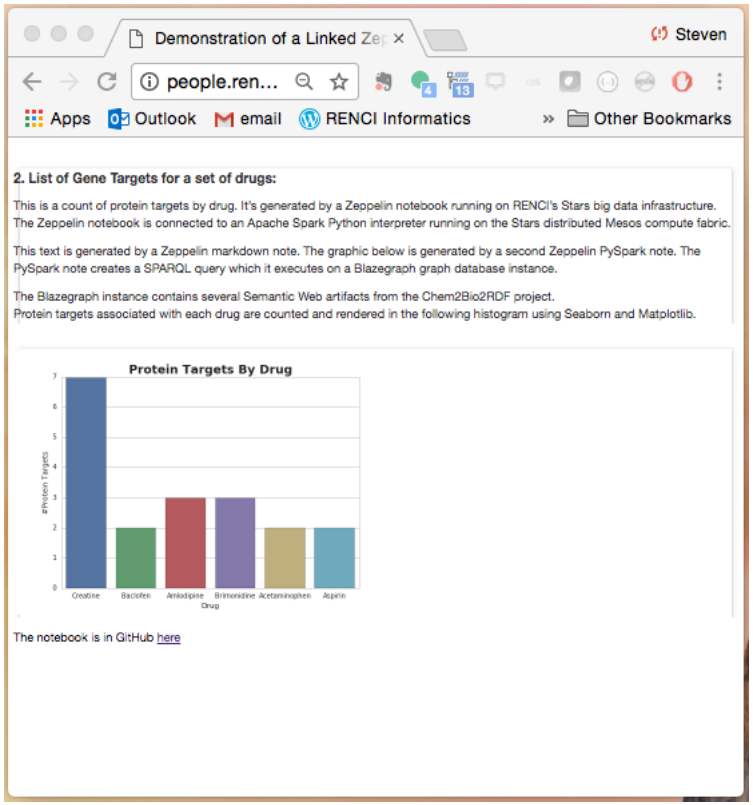


Took 1 sec. Last updated by admin at February 16 2017, 10:34:08 PM.

Zeppelin

Publish note output; embed in web apps:





Zeppelin : Graph + Mapreduce

Find biological pathways associated with a protein:

```
%pyspark
wikipathways_iri = "http://sparql.wikipathways.org/"
wikipathways = SPARQLWrapper2 (wikipathways_iri)
query = Template ("""
PREFIX wp:      <http://vocabularies.wikipathways.org/wp#>
PREFIX rdfs:    <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dcterms: <http://purl.org/dc/terms/>
SELECT DISTINCT ?pathway str(?label) as ?geneProduct
WHERE {
    ?geneProduct a wp:GeneProduct .
    ?geneProduct rdfs:label ?label .
    ?geneProduct dcterms:isPartOf ?pathway .
    ?pathway a wp:Pathway .
    FILTER regex(str(?label), "${protein}").
}""").substitute (protein="CYP")
results = query_sparql (query = query, service = wikipathways)
```

Zeppelin : Graph + Mapreduce

Spark mapreduce: Top 10 pathways by gene count

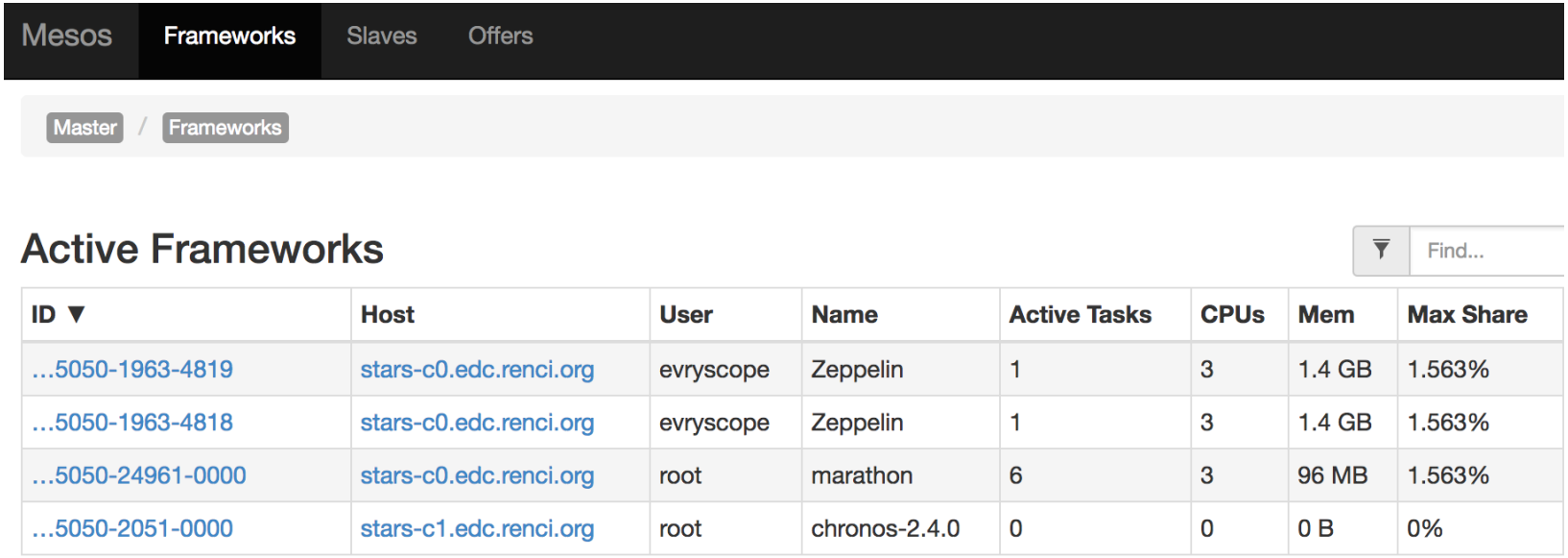
```
path_gene = []
for binding in results.bindings:
    path_gene.append ([ binding["pathway"].value, binding["geneProduct"].value])

s_path_gene = sc.parallelize (path_gene). \
    map (lambda p : ( p[0].split('/')[-1], 1 )). \
    reduceByKey (lambda x,y: x+y). \
    sortBy (lambda p : p[1], ascending=False). \
    take (10)
```

```
Top 10 gene count:
    -- (u'WP43_r84102', 60)
    -- (u'WP702_r73516', 55)
    -- (u'WP1077_r80775', 52)
    -- (u'WP1006_r80841', 51)
    -- (u'WP3248_r85056', 24)
    -- (u'WP981_r80762', 20)
    -- (u'WP3264_r80908', 16)
    -- (u'WP3131_r89204', 15)
    -- (u'WP299_r89331', 13)
    -- (u'WP970_r80700', 12)
```

Zeppelin : Scale

The interpreter scales to the Mesos cluster:

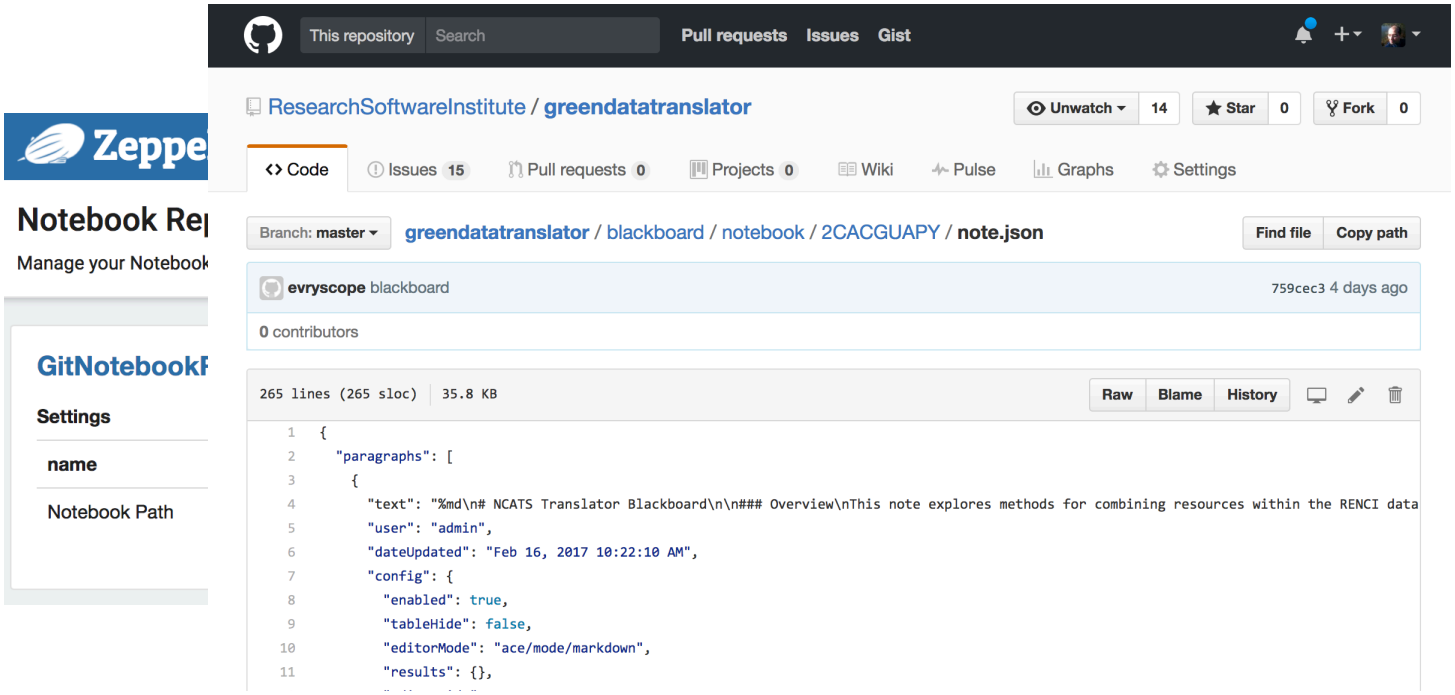


ID ▼	Host	User	Name	Active Tasks	CPUs	Mem	Max Share
...5050-1963-4819	stars-c0.edc.renci.org	evryscope	Zeppelin	1	3	1.4 GB	1.563%
...5050-1963-4818	stars-c0.edc.renci.org	evryscope	Zeppelin	1	3	1.4 GB	1.563%
...5050-24961-0000	stars-c0.edc.renci.org	root	marathon	6	3	96 MB	1.563%
...5050-2051-0000	stars-c1.edc.renci.org	root	chronos-2.4.0	0	0	0 B	0%

Zeppelin : SCM

Notebooks pushed to GitHub @ 15min

- But w/o per-user audit trail



Next

- Visualization
- Authorization
- Spark
- GPU

STARS

ABOUT DASHBOARD

Interactive Analytics

by stevencox

Data Science Notebooks

We've shown a few ways RENCI's Stars cluster can be applied to sizable data sets. But running any of these systems involves logging in to a relatively low level interface. We SSH into the cluster and run scripts. And there's nothing wrong with that.

But recently, data science has seen the rise of interactive, graphical notebooks.