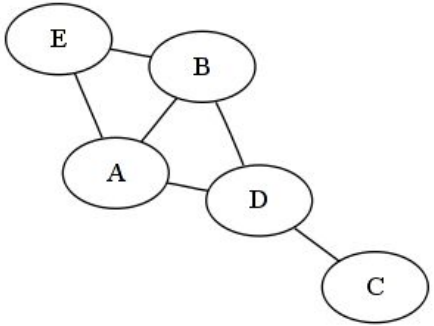
csci-e-29 week 3 section

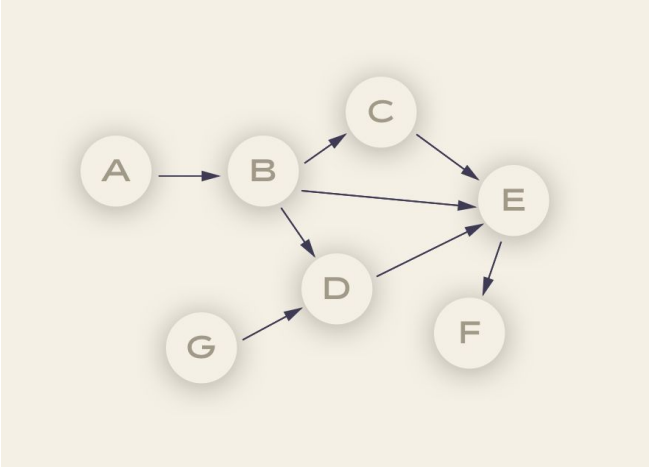
Agenda

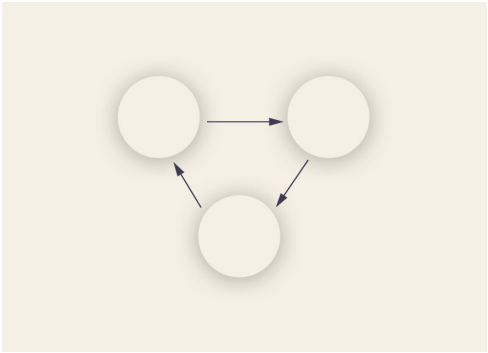
- Directed Acyclic graphs
- Map Reduce
- Task scheduling frameworks
- Pset 2 preview

Directed Acyclic Graphs (DAGs)

- Directed vs undirected graphs
- Acyclic vs cyclic graphs

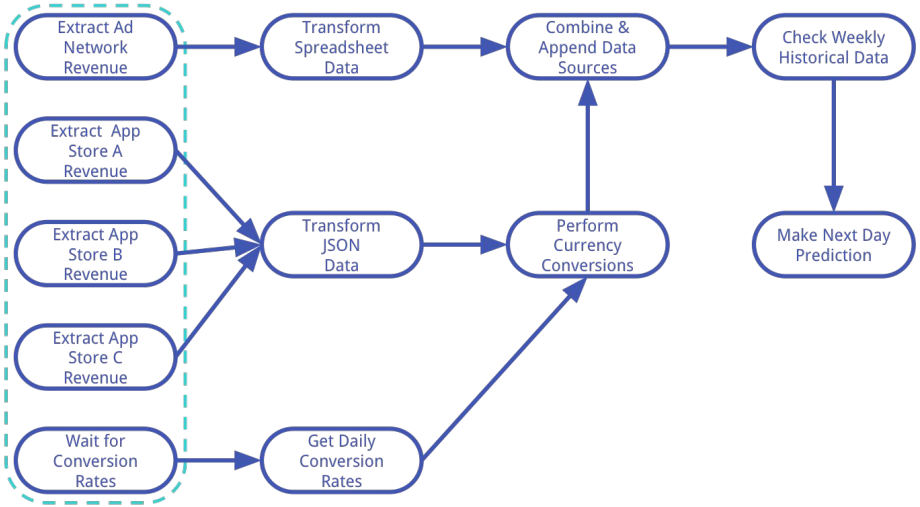






DAGs for representing computation graphs

“do C after A and B, then do D”

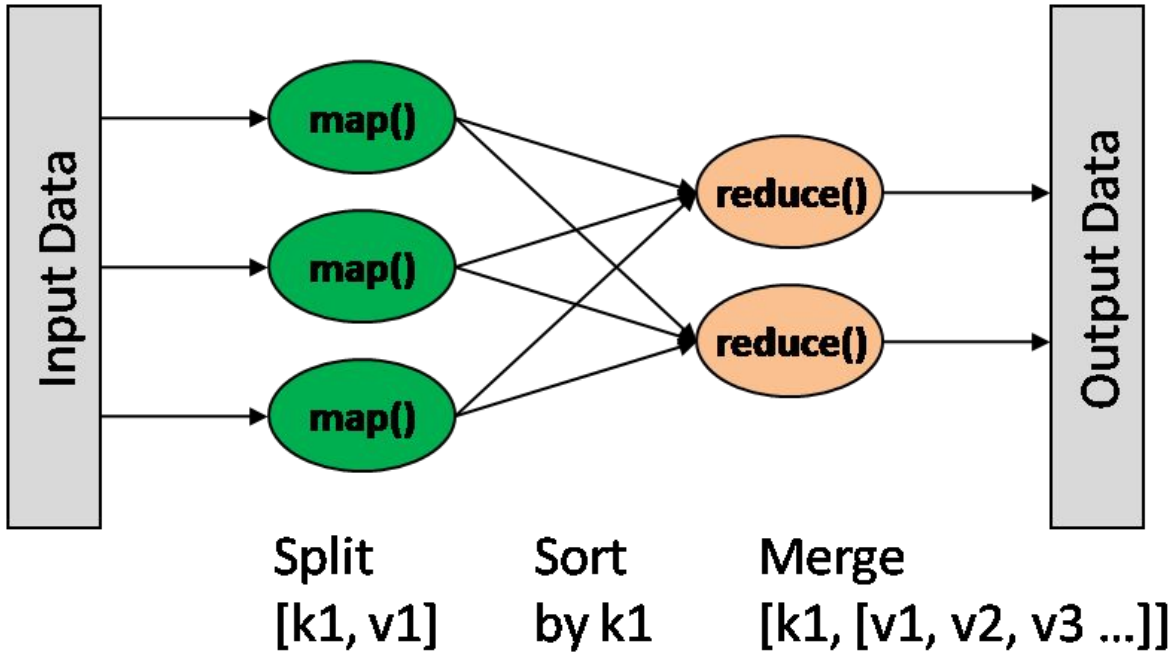


Example

Spotify:

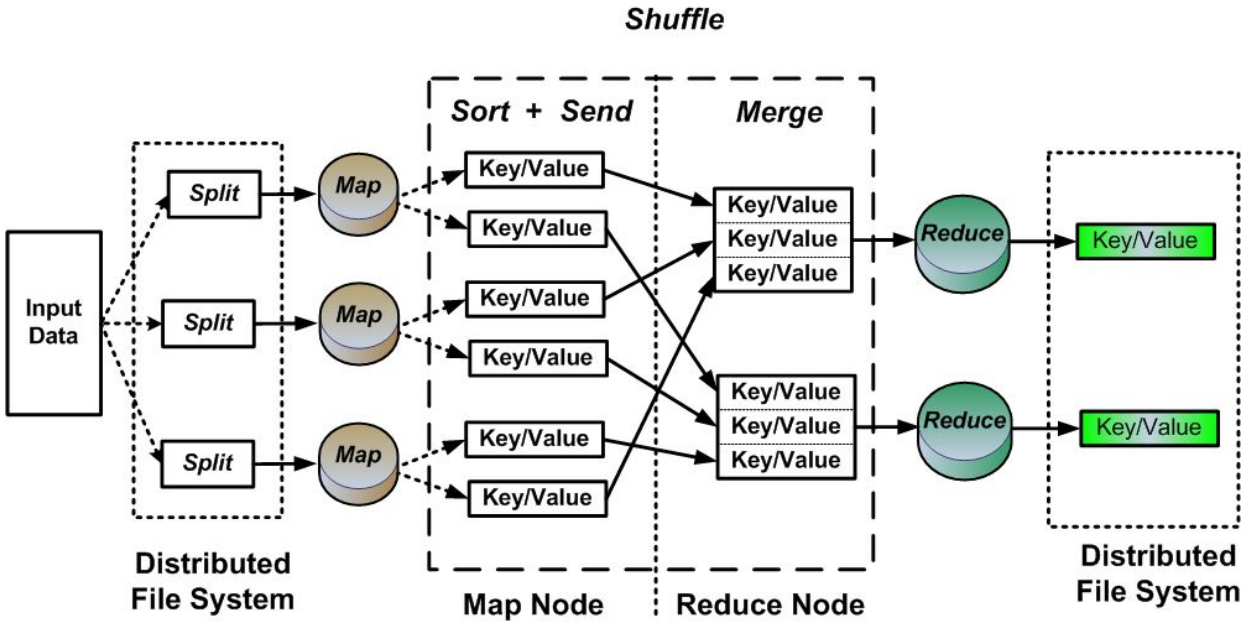
- What were the top 10 most popular artists in 2017 (i.e. most number of song plays)
- Stream data is stored in log text files, one for each day. Each row represents a stream (timestamp, artist, track)

Map reduce


Input Data
map()
map()
map()
reduce()
reduce()
Output Data
Split
[k1, v1]
Sort
by k1
Merge
[k1, [v1, v2, v3 ...]]

Map reduce

Usually in a distributed context (input, output, compute nodes)

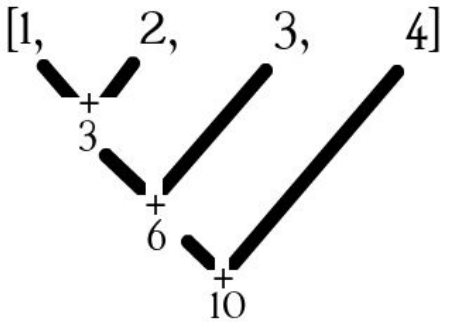


“map reduce” in python

map(fun:x, iterable): apply function to each element of iterable

reduce(fun:x,y, iterable): apply function pairwise to produce a single result

```
from functools import reduce
a = [1,2,3,4]
map(lambda x: x**2, a)
reduce(lambda x,y: x+y, 1)
```



“map reduce” in pandas

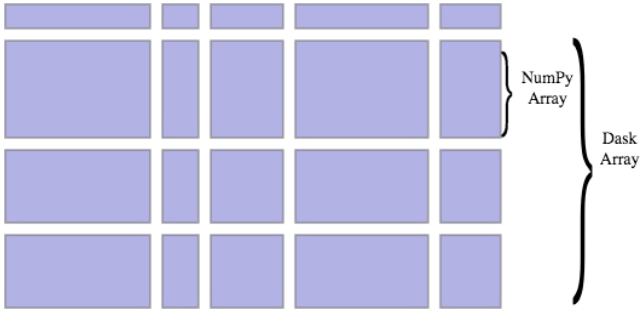
pandas groupby / apply

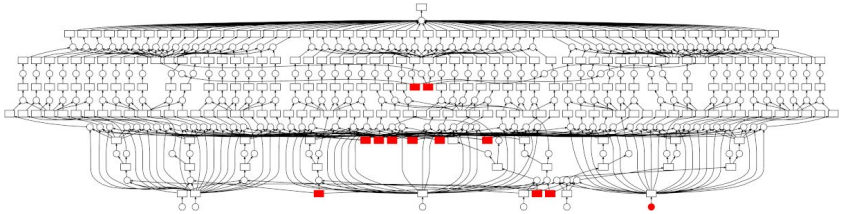
```
import pandas as pd
df = pd.DataFrame(...)
grouped = df.groupby(‘A’)
grouped.apply(function)
```

Dask

https://www.youtube.com/watch?v=hiPvmeLhInw

https://mybinder.org/v2/gh/dask/dask-examples/master?urlpath=lab





“map reduce” in dask

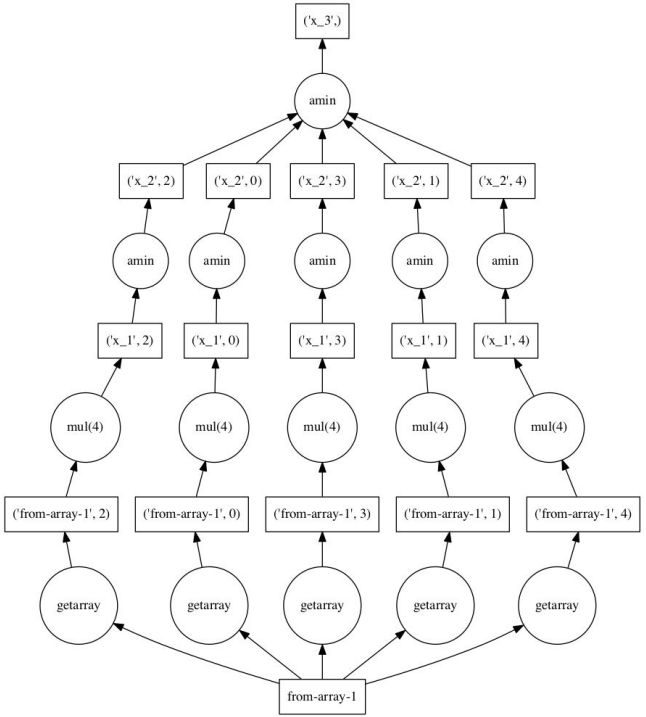
In [3]:
```
import numpy as np

# create an array of normally-distributed random numbers
a = np.random.randn(1000)

# multiply this array by a factor
b = a * 4

# find the minimum value
b_min = b.min()
print(b_min)
```

```
-11.4051061336
```

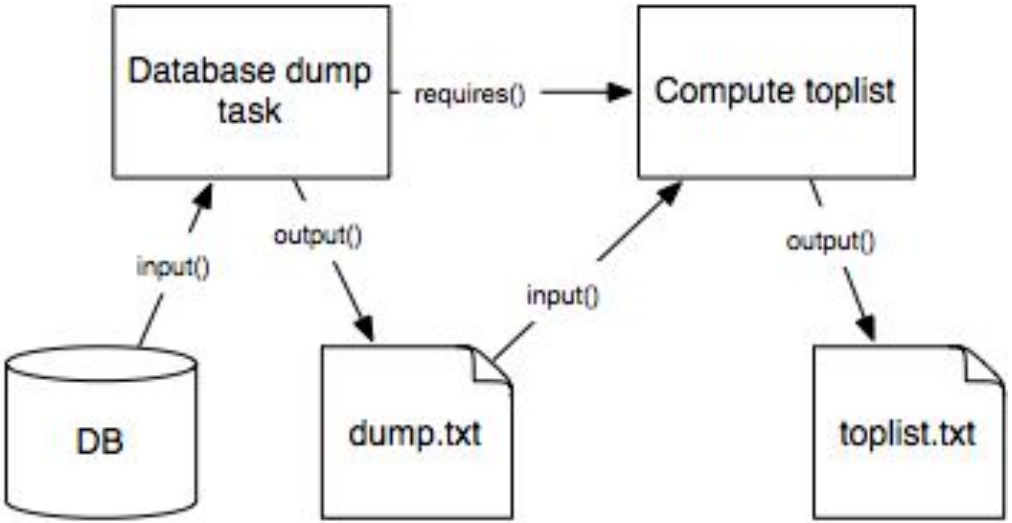
In [4]:
```
import dask.array as da

# create a dask array from the above array
a2 = da.from_array(a, chunks=200)

# multiply this array by a factor
b2 = a2 * 4

# find the minimum value
b2_min = b2.min()
print(b2_min)
```

```
dask.array<x_3, shape=(), chunks=(), dtype=float64>
```

b2_min.compute()



DAG for computation across systems and languages

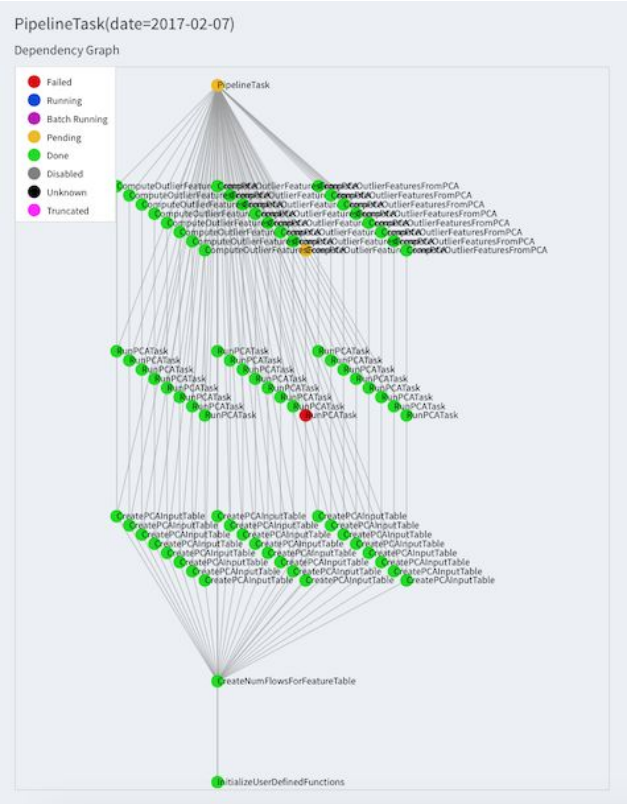


Task scheduling frameworks

- Luigi
- Airflow

Advantages of frameworks

- UI
- Cron scheduling
- Detect task failures
- Notifications / emails on task failures



Activity

Think of a complex data pipeline or process at your organization, or an organization you find interesting. How would you express it as a DAG?

Pset 2

Extending / Overriding classes

```
from atomicwrites import atomic_write as _backend_writer, AtomicWriter

# You probably need to inspect and override some internals of the package
class SuffixWriter(AtomicWriter):

    def get_fileobject(self, dir=None, **kwargs):
        # Override functions like this
        ...

@contextmanager
def atomic_write(file, mode='w', as_file=True, new_default='asdf', **kwargs):

    # You can override things just fine...
    with _backend_writer(some_path, writer_cls=SuffixWriter, **kwargs) as f:
        # Don't forget to handle the as_file logic!
        yield f
```

Pset 2 - environment variables

Environment variables

Bash:

- export AWS_ACCESS_KEY_ID=AKIAXXXXXX

Python:

- import os
- os.environ.get(‘AWS_ACCESS_KEY_ID’)

Pset 2 - environment variables

Getting environment variables into containers:

- at runtime: docker-compose -e AWS_ACCESS_KEY_ID=XX app python test.py
- specify env variables in docker-compose
  - save values in .env file in same directory as docker-compose.yml
  - can pass through environment variables from host

https://docs.docker.com/compose/environment-variables/#substitute-environment-variables-in-compose-files

https://docs.docker.com/compose/env-file/