



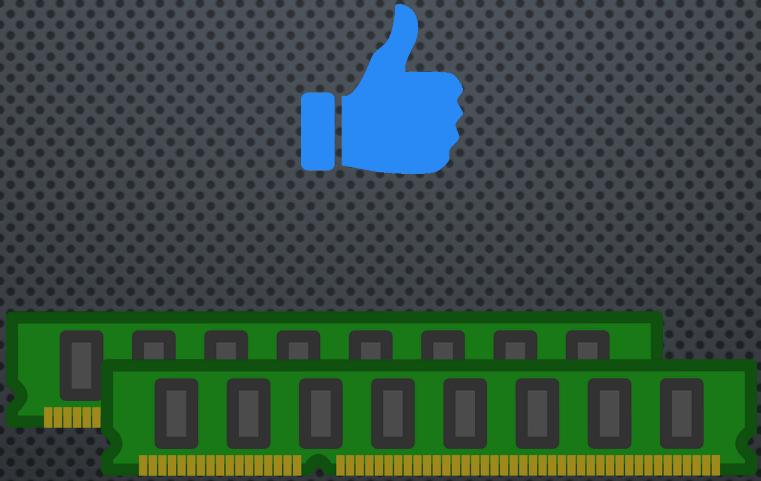
PIP INSTALL DASK FOR DATA SCIENCE

ALICE ADATIVA FERREIRA MENEZES

ROTEIRO

1. MUITOS DADOS PARA POUCA MEMÓRIA.
2. O QUE É O DASK?
3. COLLECTIONS: ARRAY
4. COLLECTIONS: DATAFRAME
5. COLLECTIONS: BAG
6. CONSIDERAÇÕES FINAIS
7. QUERO SABER MAIS. E AGORA?

MUITOS DADOS PARA POUCA MEMÓRIA



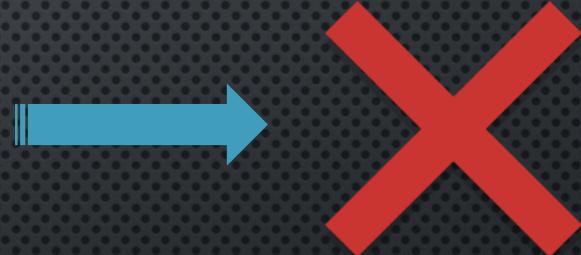
16 GB



MUITOS DADOS PARA POUCA MEMÓRIA



24 GB



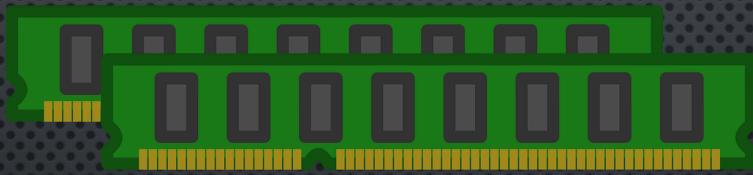
MUITOS DADOS PARA POUCA MEMÓRIA



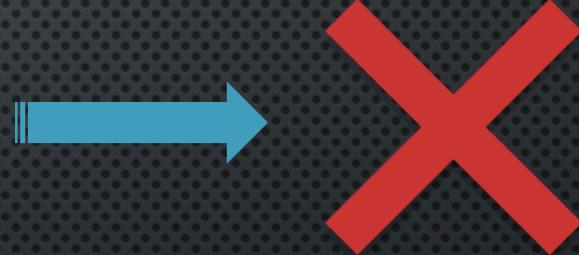
10 GB



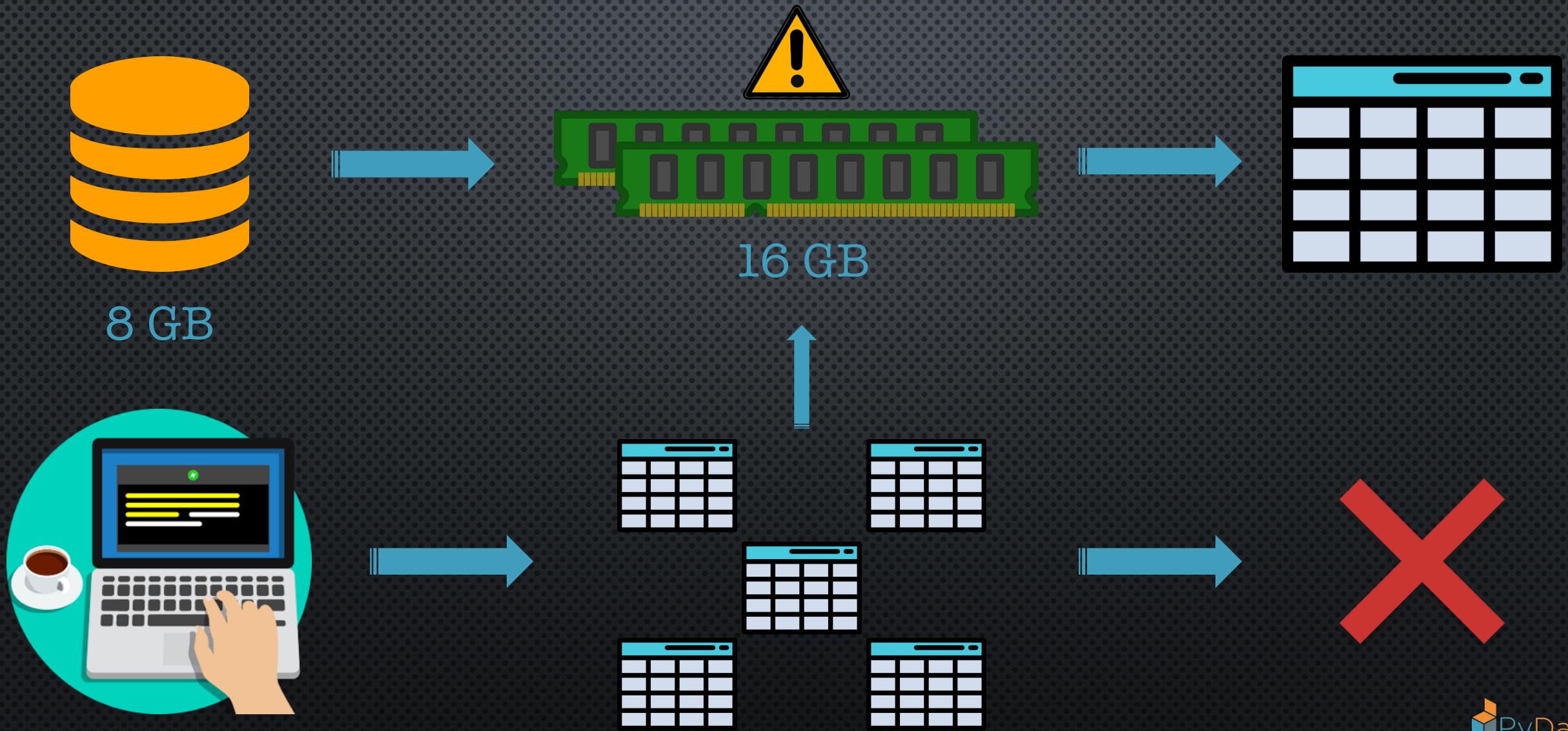
10 GB



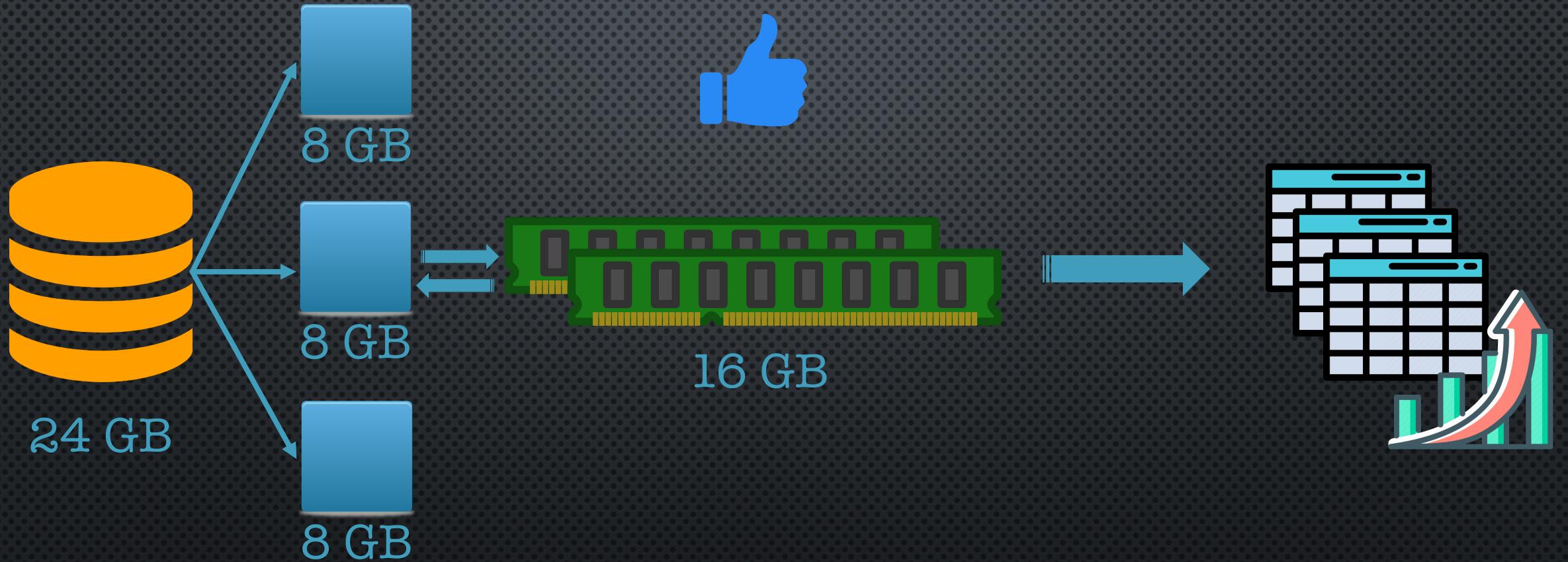
16 GB



MUITOS DADOS PARA POUCA MEMÓRIA



PARALELIZANDO O PROCESSAMENTO

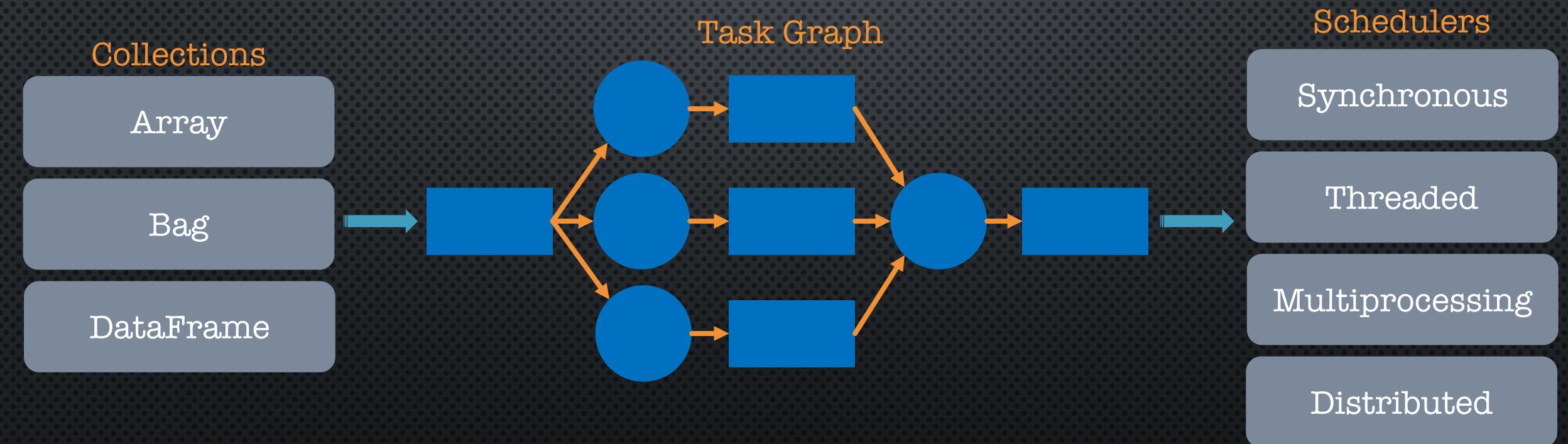


PARALELIZANDO O PROCESSAMENTO



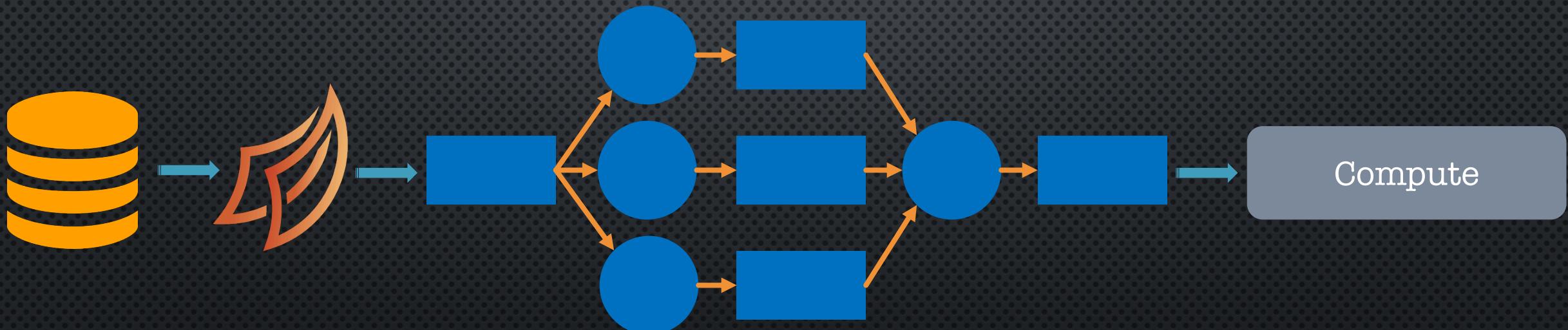
O QUE É O DASK?

- BIBLIOTECA OPEN SOURCE PARA PARALELIZAR COMPUTAÇÃO EM PYTHON.



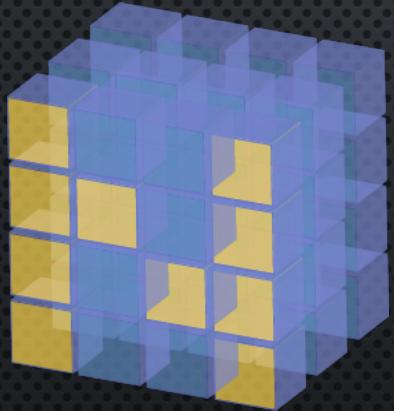
O QUE É O DASK?

- TRABALHA COM PIPELINES (LAZY EVALUATION).



O QUE É O DASK?

- COMPATÍVEL COM OUTRAS BIBLIOTECAS (NUMPY, PANDAS E SCIKIT-LEARN).



O QUE É O DASK?

- VOCÊ NÃO PRECISA REESCREVER SEU CÓDIGO COMPLETAMENTE.

```
import pandas as pd  
df = pd.read_csv('2019-01-01.csv')  
df.groupby(df.user_id).value.mean()
```



```
import dask.dataframe as dd  
df = dd.read_csv('2019-*-*csv')  
df.groupby(df.user_id).value.mean().compute()
```

```
import numpy as np  
f = h5py.File('myfile.hdf5')  
x = np.array(f['/small-data'])  
x - x.mean(axis=1)
```



```
import dask.array as da  
f = h5py.File('myfile.hdf5')  
x = da.from_array(f['/big-data'],  
                  chunks=(1000, 1000))  
x - x.mean(axis=1).compute()
```

O QUE É O DASK?

- FOI LANÇADO EM JANEIRO DE 2015 (VERSAO 0.2.0).
- VERSAO ATUAL: 1.1.0 (18/01/2019)

COLLECTIONS: ARRAY

- O DASK ARRAY É UMA EXTENSÃO DO NUMPY ARRAY.
- MUITOS DOS ATRIBUTOS, AGREGAÇÕES, TRANSFORMAÇÕES DE ARRAYS E OPERAÇÕES MATEMÁTICAS DO NUMPY TAMBÉM PODEM SER UTILIZADOS NO DASK.

COLLECTIONS: ARRAY

- ATRIBUTOS: SHAPE, NDIM, NBYTES, DTYPE, SIZE, ETC.
- AGREGAÇÕES: MAX, MIN, MEAN, STD, SUM, ETC.
- TRANSFORMAÇÕES: RESHAPE, REPEAT, STACK, FLATTEN, TRANSPOSE, T, ETC.
- OPERAÇÕES MATEMÁTICAS: ROUND, REAL, CONJ, DOT, ETC.

COLLECTIONS: ARRAY

```
import numpy as np
```

```
In: n = np.random.randn(10000)
n.shape
Out: (10000,)
```

```
In: n.sum()
Out: -9.939841125094565
```

```
import dask.array as da
```

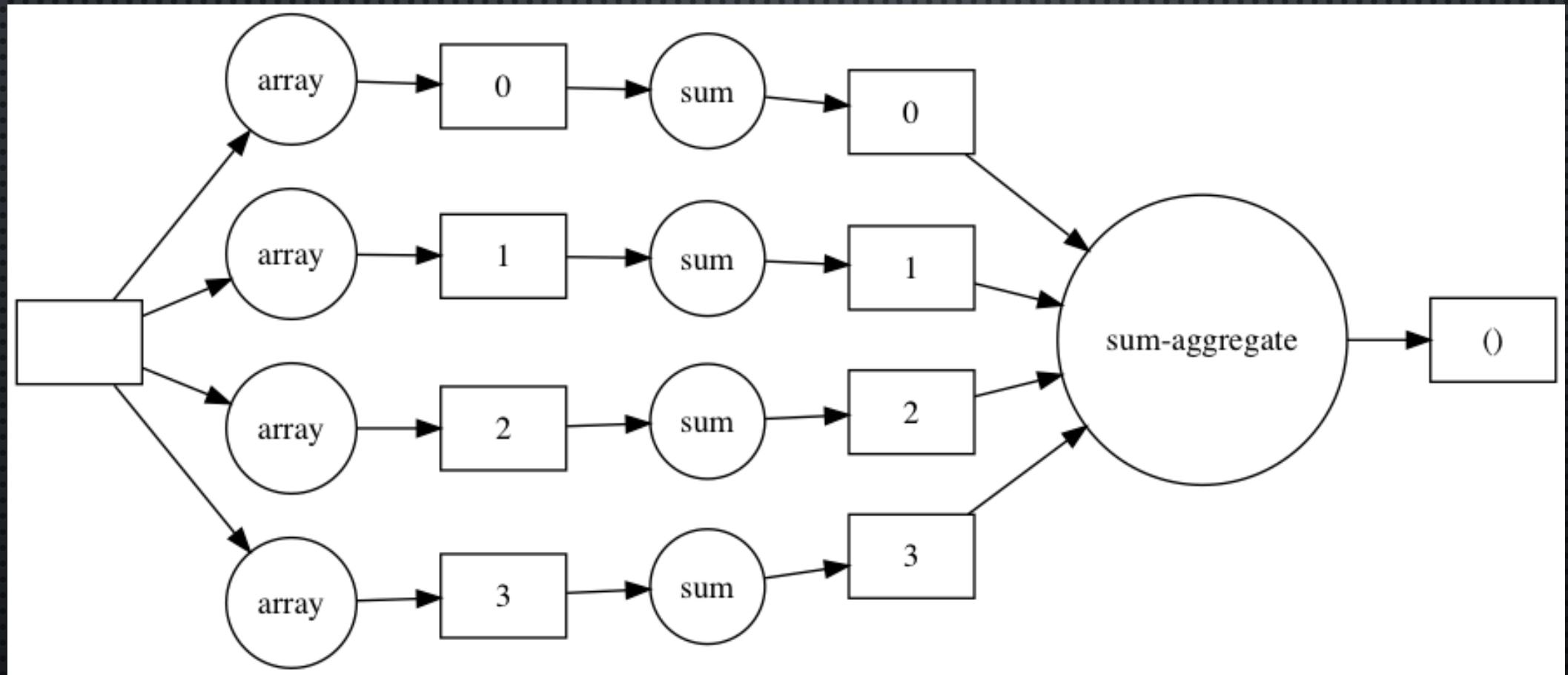
```
In: n_chunks = 4
dask_array = da.from_array(n, chunks = len(n) // n_chunks)
dask_array.chunks
Out: ((2500, 2500, 2500, 2500),)
```

```
In: result = dask_array.sum()
result
Out: dask.array<sum-aggregate, shape=(), dtype=float64, chunksizer>
```

```
In: result.compute()
Out: -9.939841125094565
```

COLLECTIONS: ARRAY

```
In: result.visualize(rankdir = 'LR')
```



COLLECTIONS: DATAFRAME

- UMA VERSÃO DO PANDAS DATAFRAME COM LAZY EVALUATION.
- PARA ALGUMAS OPERAÇÕES, O PARALELISMO NÃO É NECESSÁRIO (NÃO É NECESSÁRIO UTILIZAR O .COMPUTE()).
 - DF.HEAD()
 - DF.TAIL()

COLLECTIONS: DATAFRAME

- ALGUMAS FUNCIONALIDADES DOS PANDAS QUE ESTÃO PRESENTES NO DASK:
 - INDEXAÇÃO, SELEÇÃO E REINDEXAÇÃO.
 - AGREGAÇÕES: SUM(), MEAN(), STD(), MIN(), MAX(), ETC.
 - AGRUPAMENTO DOS DADOS (GROUPBY()).
 - CONVERSÕES E OPERAÇÕES DE DATA E HORA (TO_DATETIME()).

COLLECTIONS: DATAFRAME

- ALGUMAS FUNCIONALIDADES DOS PANDAS QUE NÃO ESTÃO PRESENTES NO DASK:
 - NÃO SÃO SUPORTADOS ALGUNS FORMATOS DE ARQUIVO (.XLS, .ZIP, .GZ).
- E SE EU PRECISAR DO PANDAS PARA REALIZAR ALGUMA OPERAÇÃO QUE NÃO É POSSÍVEL NO DASK?
 - USE O DECORATOR @DELAYED DO DASK.

COLLECTIONS: DATAFRAME

```
import glob
import pandas as pd
import dask.dataframe as dd
import dask.delayed as delayed
```

```
In: @delayed
    def read_files(filename):
        df = pd.read_csv(filename, parse_dates=['DATE'])
        return df
```

```
In: my_filenames = glob.glob('data/my_files-[1-5].csv')
```

```
In: for filename in my_filenames:
    dataframes = []
    dataframes.append(read_files(filename))
dataframes
```

```
Out: [Delayed('read_files-c4246eac-9f8e-4730-be35-80a30e2d0c19')]
```

```
In: df_dask = dd.from_delayed(dataframes)
df_dask['VALUE'].mean().compute()
```

COLLECTIONS: DATAFRAME

- COMO SABER SE O DASK OU PANDAS É APROPRIADO PARA O MEU PROBLEMA?
 - QUAL É O TAMANHO DA SUA BASE DE DADOS?
 - QUANTO DE MEMÓRIA RAM VOCÊ TEM DISPONÍVEL?
 - AS FUNCIONALIDADES QUE VOCÊ PRECISA TEM NO DASK?
 - ESTÁ OCORRENDO MUITO PROCESSAMENTO EM PARALELO?

COLLECTIONS: DATAFRAME

```
In: %ls -lh data/yellow_taxis_2017/  
Out: yellow_tripdata_2017-01.csv  
      yellow_tripdata_2017-02.csv  
      yellow_tripdata_2017-03.csv  
      yellow_tripdata_2017-04.csv  
      yellow_tripdata_2017-05.csv  
      yellow_tripdata_2017-06.csv  
      yellow_tripdata_2017-07.csv  
      yellow_tripdata_2017-08.csv  
      yellow_tripdata_2017-09.csv  
      yellow_tripdata_2017-10.csv  
      yellow_tripdata_2017-11.csv  
      yellow_tripdata_2017-12.csv
```



COLLECTIONS: DATAFRAME

```
import glob
import pandas as pd

In: %%time
    df_pandas = pd.concat([pd.read_csv(f) for f in
                           glob.glob('data/yellow_taxis_2017/*.csv')], ignore_index = True)
Out: Wall time: 9min 16s
```

```
In: %%time
    trip_mean = df_pandas['trip_distance'].mean()
Out: Wall time: 1.11 s
```

```
In: %%time
    trip_sum = df_pandas['trip_distance'].sum()
Out: Wall time: 1.26 s
```

COLLECTIONS: DATAFRAME

```
import dask
import dask.dataframe as dd

In: %%time
    df_dask = dd.read_csv('data/yellow_taxis_2017/*.csv')
Out: Wall time: 382 ms
```

```
In: %%time
    trip_mean = df_dask['trip_distance'].mean()
Out: Wall time: 24.6 ms
```

```
In: %%time
    trip_sum = df_dask['trip_distance'].sum()
Out: Wall time: 3.61 ms
```

```
In: %%time
    dask.compute(df_dask['trip_distance'].mean(),
                  df_dask['trip_distance'].sum())
Out: Wall time: 3min 38s
(2.928659862649609, 332393739.42)
```

COLLECTIONS: BAG

- DASK ARRAYS E DASK DATAFRAMES TRABALHAM COM DADOS ESTRUTURADOS.
- O DASK BAG É PARA DADOS NÃO ESTRUTURADOS QUE PODEM ESTAR CONTIDOS EM ARQUIVOS DE TEXTO OU JSON.
- SUPORTA DADOS HETEROGENEOS.

COLLECTIONS: BAG

- UTILIZA FUNÇÕES NO LUGAR DE LOOPS (MAP, FILTER, REDUCTION).
- AO IMPORTAR TEXTOS (DASK.BAG.READ_TEXT('MEU_TEXTO.TXT')), PODE UTILIZAR FUNÇÕES PARA TRANSFORMAR STRINGS (UPPER(), SPLIT()).
- PODE SER UTILIZADO PARA PARALELIZAR ALGUMAS OPERAÇÕES EM PANDAS E NUMPY.

COLLECTIONS: BAG

```
import dask.bag as db

In: def squared(x):
     return x ** 2

In: def is_even(x):
     return x % 2 == 0

In: numbers = db.from_sequence([1, 2, 3, 4, 5, 6])

In: squares = numbers.map(squared)
    squares.compute()
Out: [1, 4, 9, 16, 25, 36]

In: evens = numbers.filter(is_even)
    evens.compute()
Out: [2, 4, 6]
```

CONSIDERAÇÕES FINAIS

- DASK PARALELIZA E OTIMIZA O PROCESSAMENTO DE DADOS POR MEIO DE PIPELINES E LAZY EVALUATION.
- EFICIENTE PARA TRABALHAR COM APENAS UM COMPUTADOR OU EM CLUSTERS.
- COMPATIBILIDADE COM OUTRAS BIBLIOTECAS DE PYTHON.

CONSIDERAÇÕES FINAIS

- PORÉM...
 - RECOMENDA-SE A UTILIZAÇÃO COM GRANDES BASES DE DADOS E EM LOCAIS COM GRANDE QUANTIDADE DE PROCESSAMENTO DE DADOS.
 - PARA BASES DE DADOS MENORES, RECOMENDA-SE O USO DAS BIBLIOTECAS TRADICIONAIS, QUE PODEM FAZER TODAS AS OPERAÇÕES NECESSÁRIAS EM MEMÓRIA RAM.



OBRIGADA!



@ADATIVAALICE



ALICE ADATIVA