INTRODUCTION TO DASK

METIS

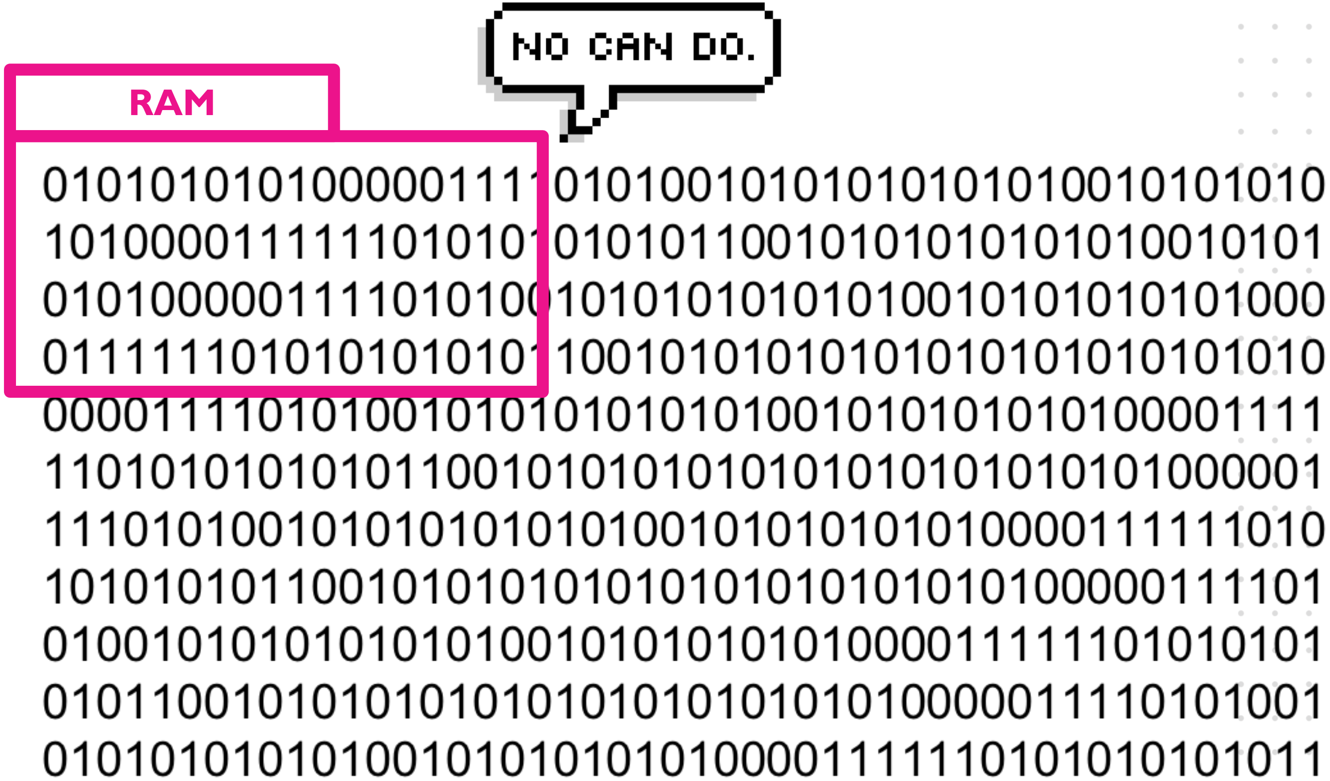
Dask

- Looks a lot like Pandas, but automatically breaks dataframes into smaller chunks (partitions)
- When processing the data, it works on each partition one at a time, storing the active piece in memory and the rest on hard drive

“Hey DASK, load this data for me.”

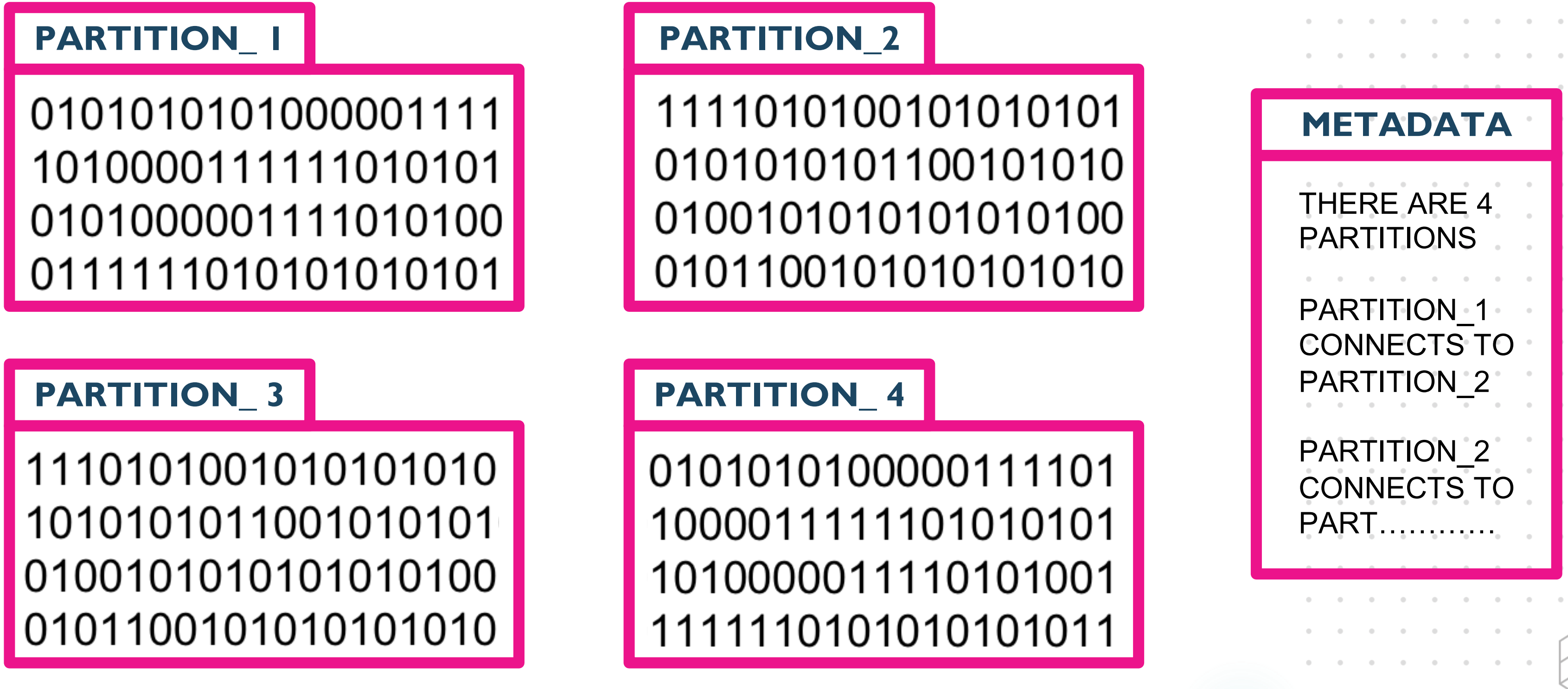
```
010101010100000111101010010101010101010010101010
101000011111101010101010110010101010101010010101
010100000111101010010101010101010010101010101000
011111101010101010110010101010101010101010101010
000011110101001010101010101010010101010101000011 11
110101010101011001010101010101010101010101000001
111010100101010101010100101010101010000111111010
101010101100101010101010101010101010101000000111101
010010101010101010010101010101010000111111010101
010110010101010101010101010101010101000000111101010 01
010101010101001010101010101000011111101010101011
```

“Hey DASK, load this data for me.”



```
01010101010000011110101001010101010101001010101010
10100001111110101010101011001010101010101010010101
01010000011110101001010101010101001010101010101000
01111110101010101011001010101010101010101010101010
00001111010100101010101010100101010101010100001111
11010101010101100101010101010101010101010101000001
11101010010101010101010010101010101000001111111010
10101010110010101010101010101010101010100000111101
010010101010101010010101010101000011111110101010101
0101100101010101010101010101010100000011110101001
01010101010100101010101010100000111111101010101011
```

“Hey DASK, load this data for me.”



PARTITION_ I

0101010101000001111
1010000111111010101
0101000001111010100
0111111010101010101

PARTITION_2

1111010100101010101
0101010101100101010
0100101010101010100
0101100101010101010

PARTITION_ 3

1110101001010101010
1010101011001010101
0100101010101010100
0101100101010101010

PARTITION_ 4

0101010100000111101
1000011111101010101
1010000011110101001
1111110101010101011

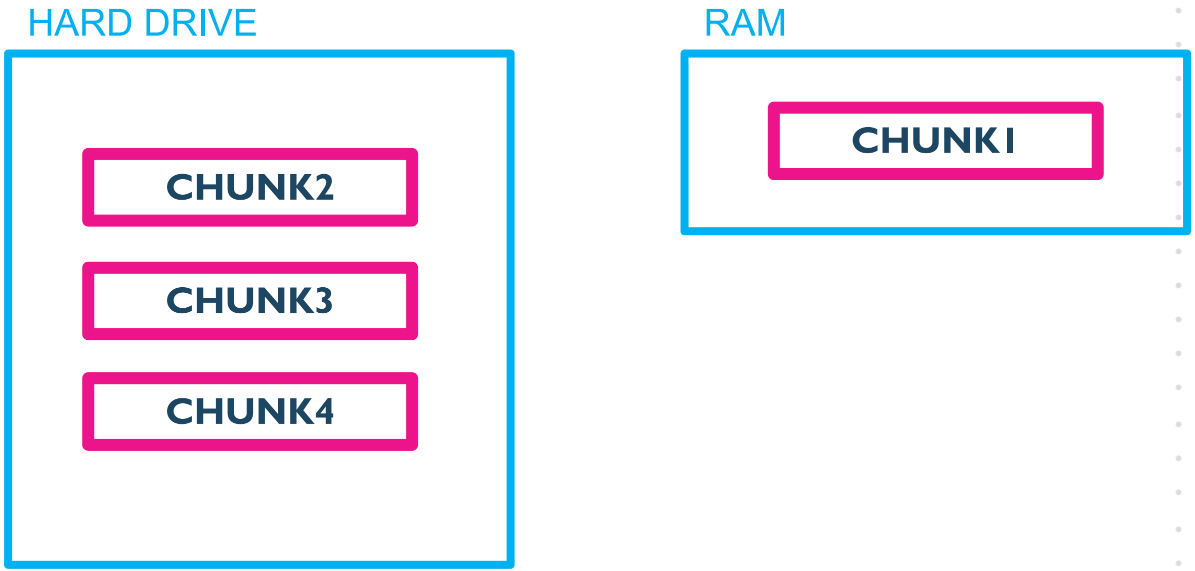
METADATA

THERE ARE 4 PARTITIONS

PARTITION_1 CONNECTS TO PARTITION_2

PARTITION_2 CONNECTS TO PART…………

Let’s see how this helps us by asking a simple question of our data.



HARD DRIVE

CHUNK2

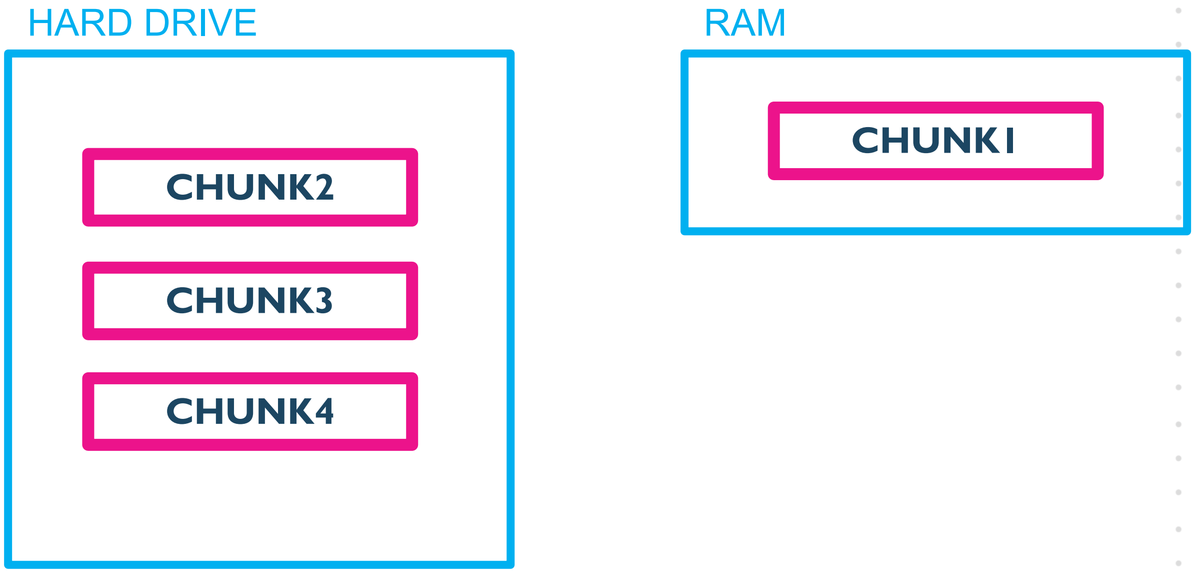
CHUNK3

CHUNK4

RAM

CHUNK1

"I'd like to know the average square footage across the whole dataset."



HARD DRIVE

CHUNK2

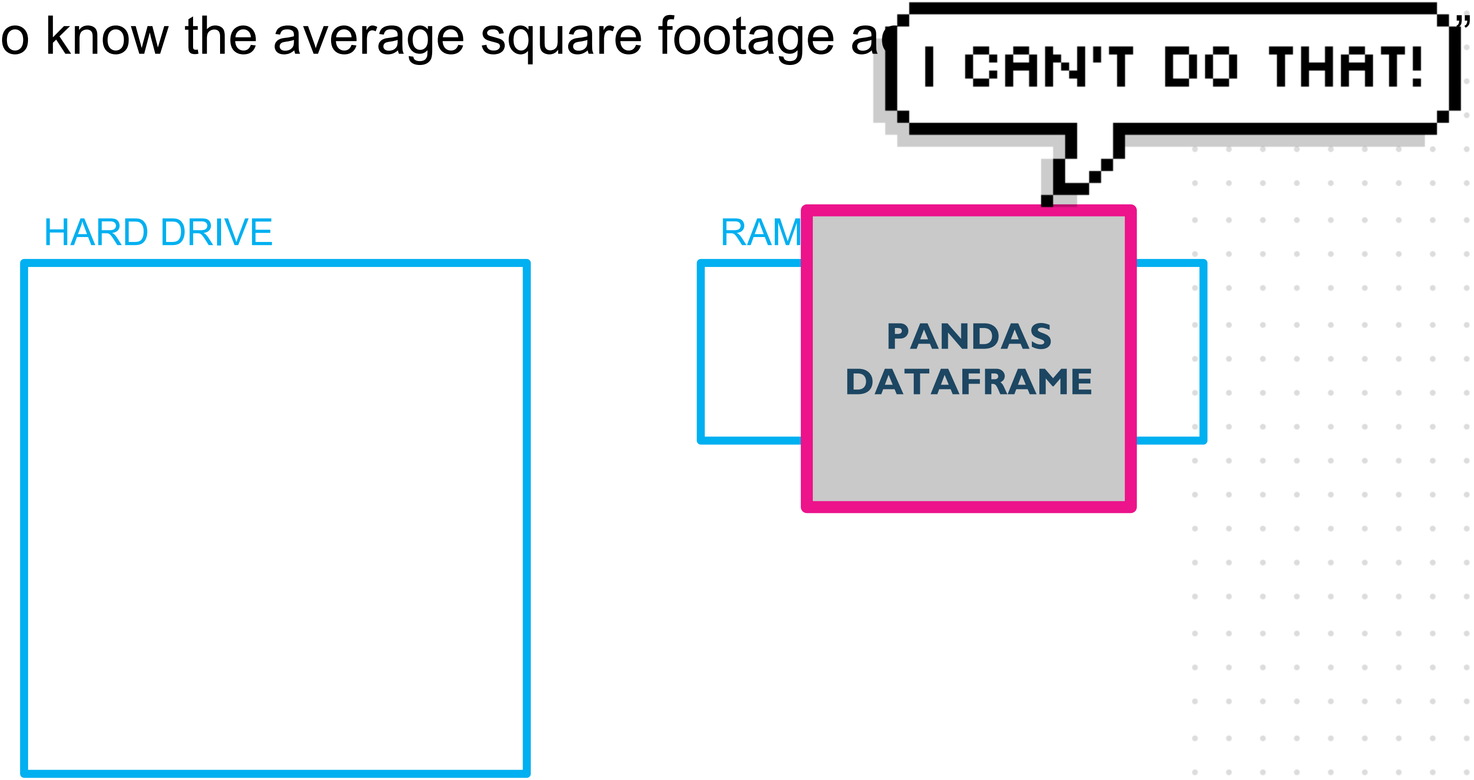
CHUNK3

CHUNK4

RAM

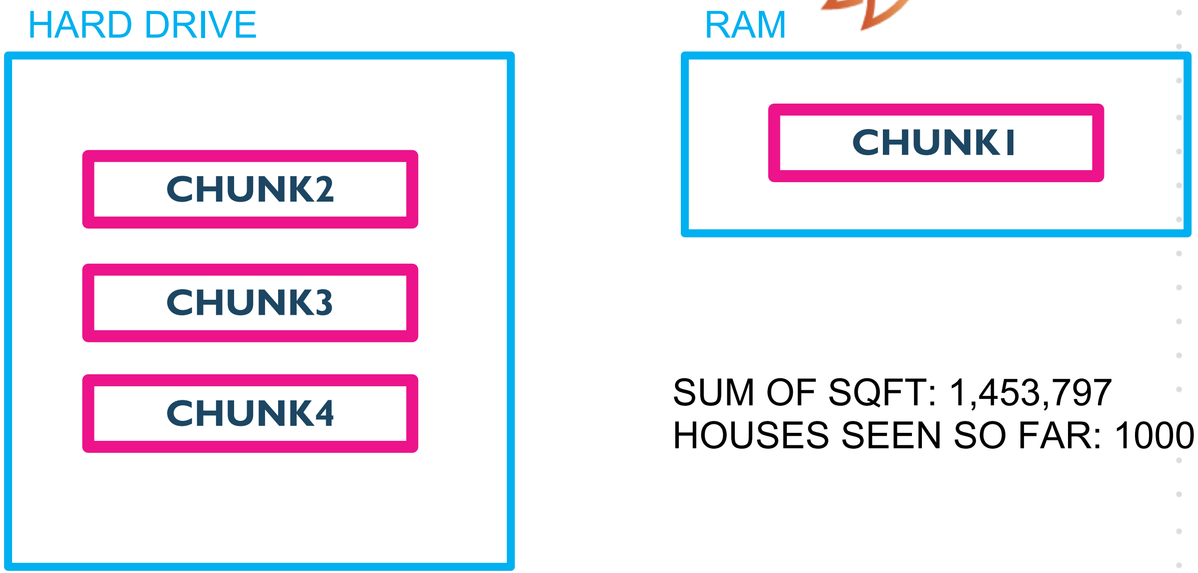
CHUNK1

“I’d like to know the average square footage a ”



“I’d like to know the average square footage across the whole dataset.”

DASK



HARD DRIVE

CHUNK2

CHUNK3

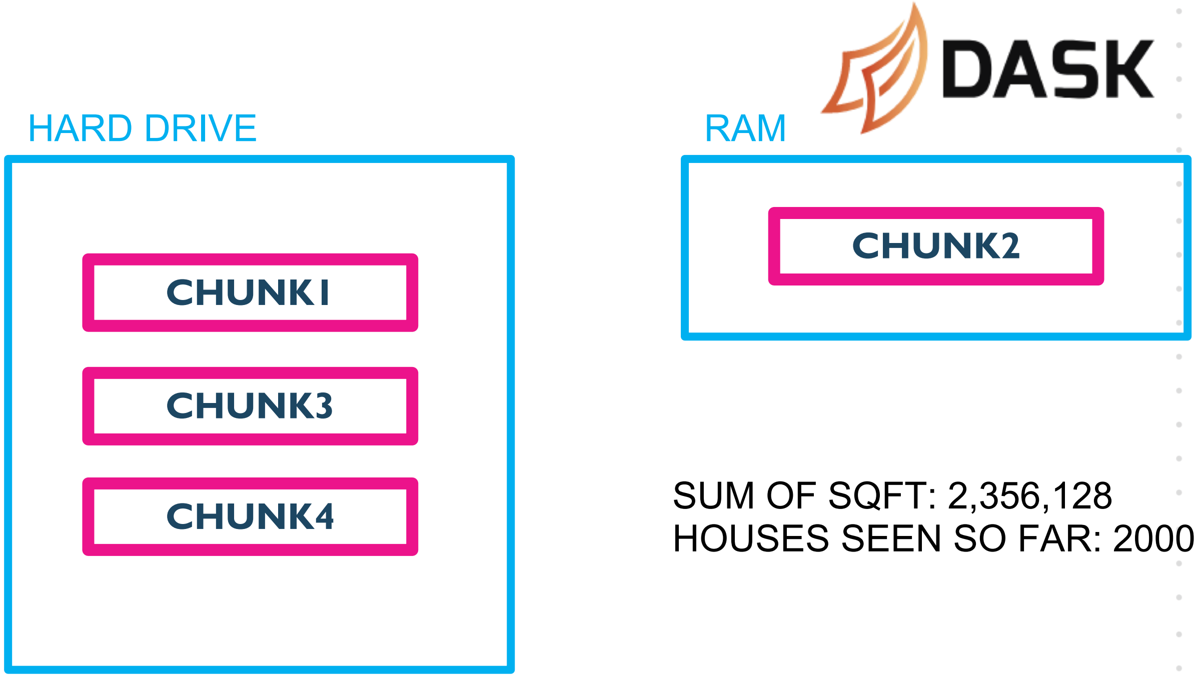
CHUNK4

RAM

CHUNK1

SUM OF SQFT: 1,453,797
HOUSES SEEN SO FAR: 1000

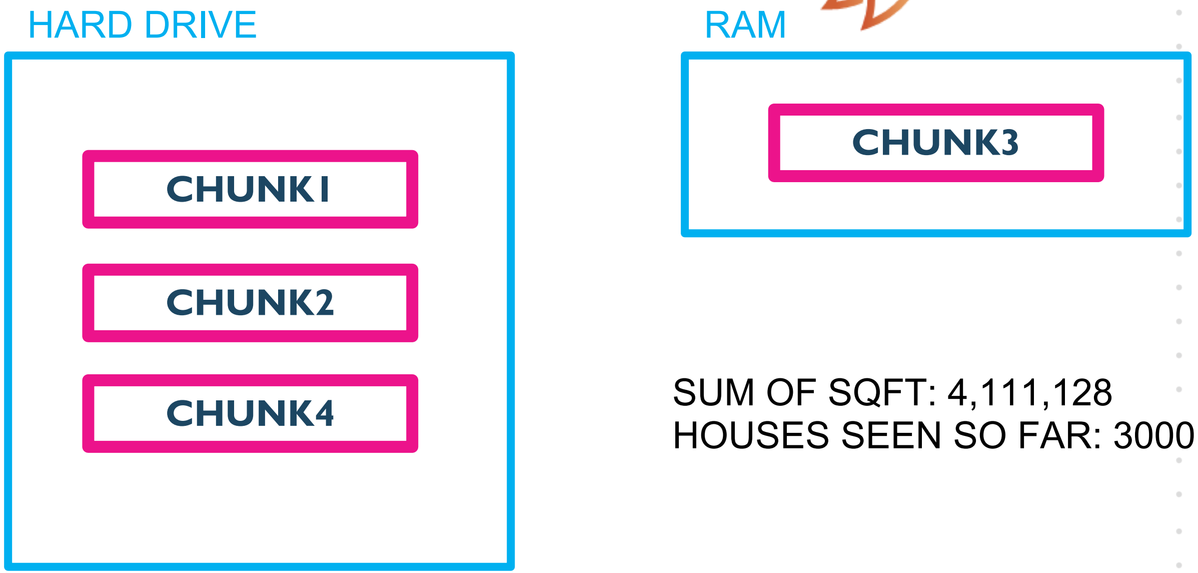
“I’d like to know the average square footage across the whole dataset.”



SUM OF SQFT: 2,356,128
HOUSES SEEN SO FAR: 2000

“I’d like to know the average square footage across the whole dataset.”

DASK



HARD DRIVE

CHUNK1

CHUNK2

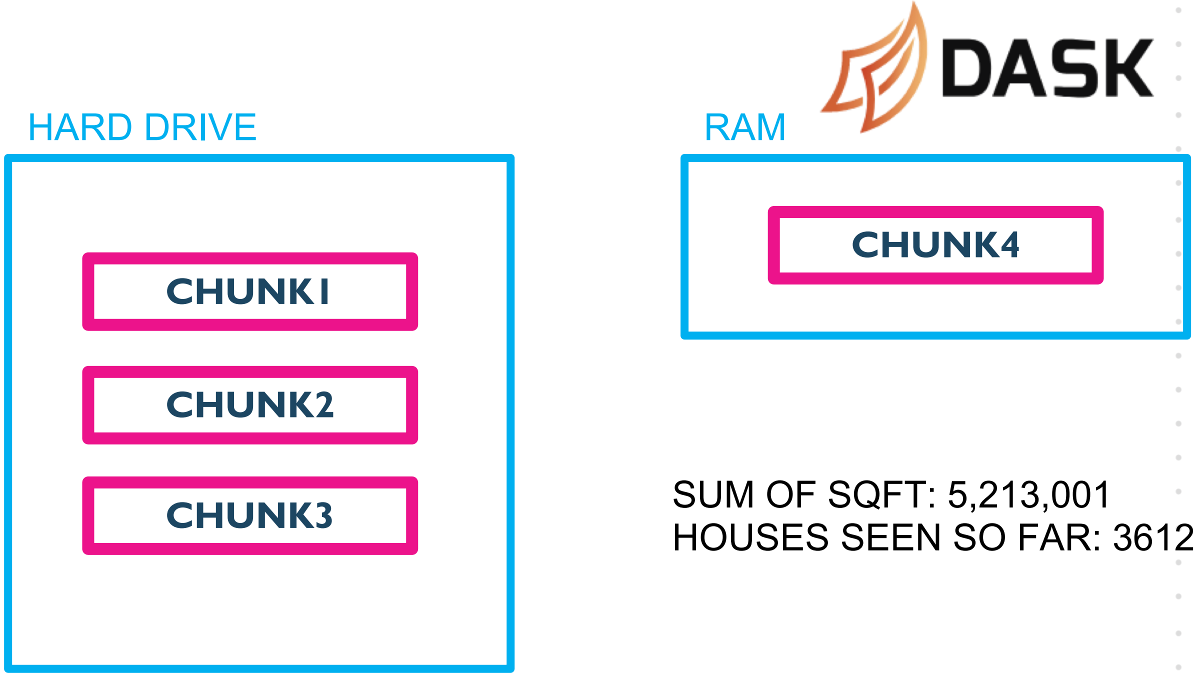
CHUNK4

RAM

CHUNK3

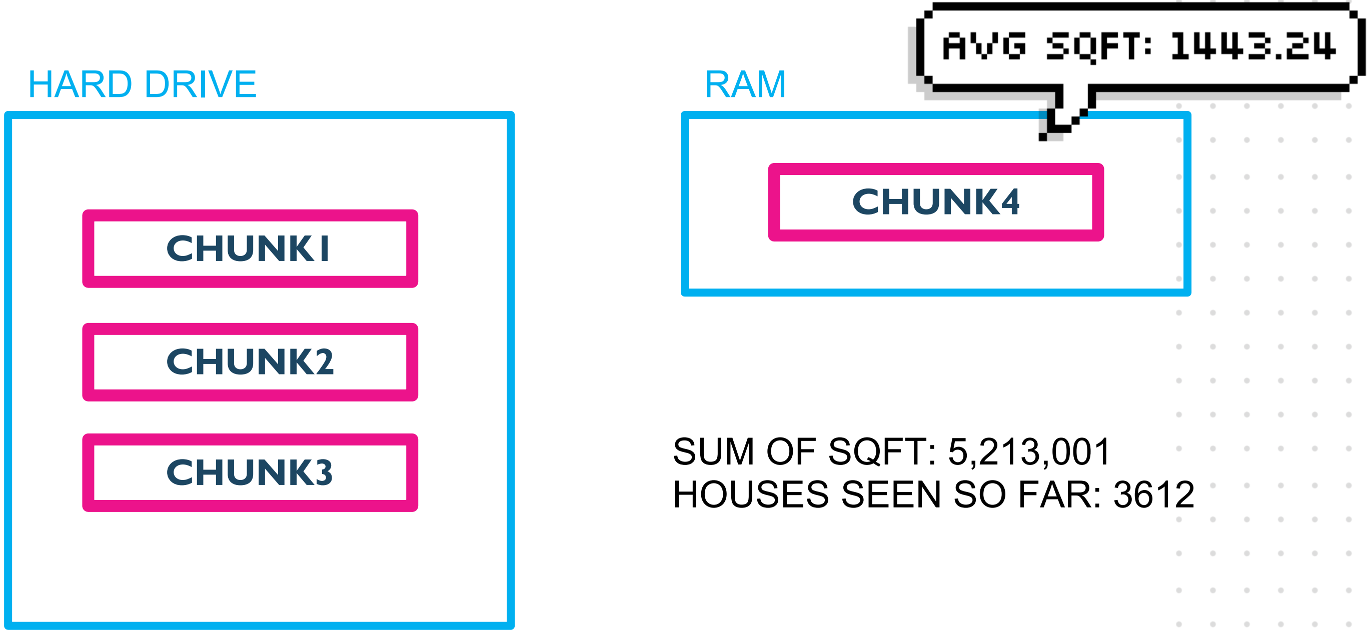
SUM OF SQFT: 4,111,128
HOUSES SEEN SO FAR: 3000

“I’d like to know the average square footage across the whole dataset.”



SUM OF SQFT: 5,213,001
HOUSES SEEN SO FAR: 3612

“I’d like to know the average square footage across the whole dataset.”



SUM OF SQFT: 5,213,001
HOUSES SEEN SO FAR: 3612

HOW DOES DASK DO ALL THAT?

Lazy Evaluation

- One of the key differences between Pandas and Dask is that Dask considers how to do things before doing them.
- Dask makes a plan of action before doing anything, and only executes that plan once the result is required.
- This means Dask can pre-plan how to move the data around, what values to record from each partition, and what order to do all the operations.

Let’s imagine we want to…

- Multiply the length of a building by it’s width to compute the square footage (our dataset has length and width)
- Find the mean square footage of all buildings in our sample.

This will take 2 steps. So let’s see how this would be handled in Pandas and Dask.

Pandas

Me: Hey pandas, I’d like to multiply these two columns together and…

Pandas: DID IT RIGHT NOW YES I’M AWESOME

Me: Oh, okay. Uhh. Well, can we calculate the mean of that?

Pandas: THE MEAN OF WHAT? WHAT ARE YOU TALKING ABOUT.

Dask

Me: Hey dask, I’d like to multiply these two columns together and…

Dask: Uh-huh

Dask’s Checklist

- Multiply columns 1 & 3

Dask

Me: Hey dask, I’d like to multiply these two columns together and…

Dask: Uh-huh

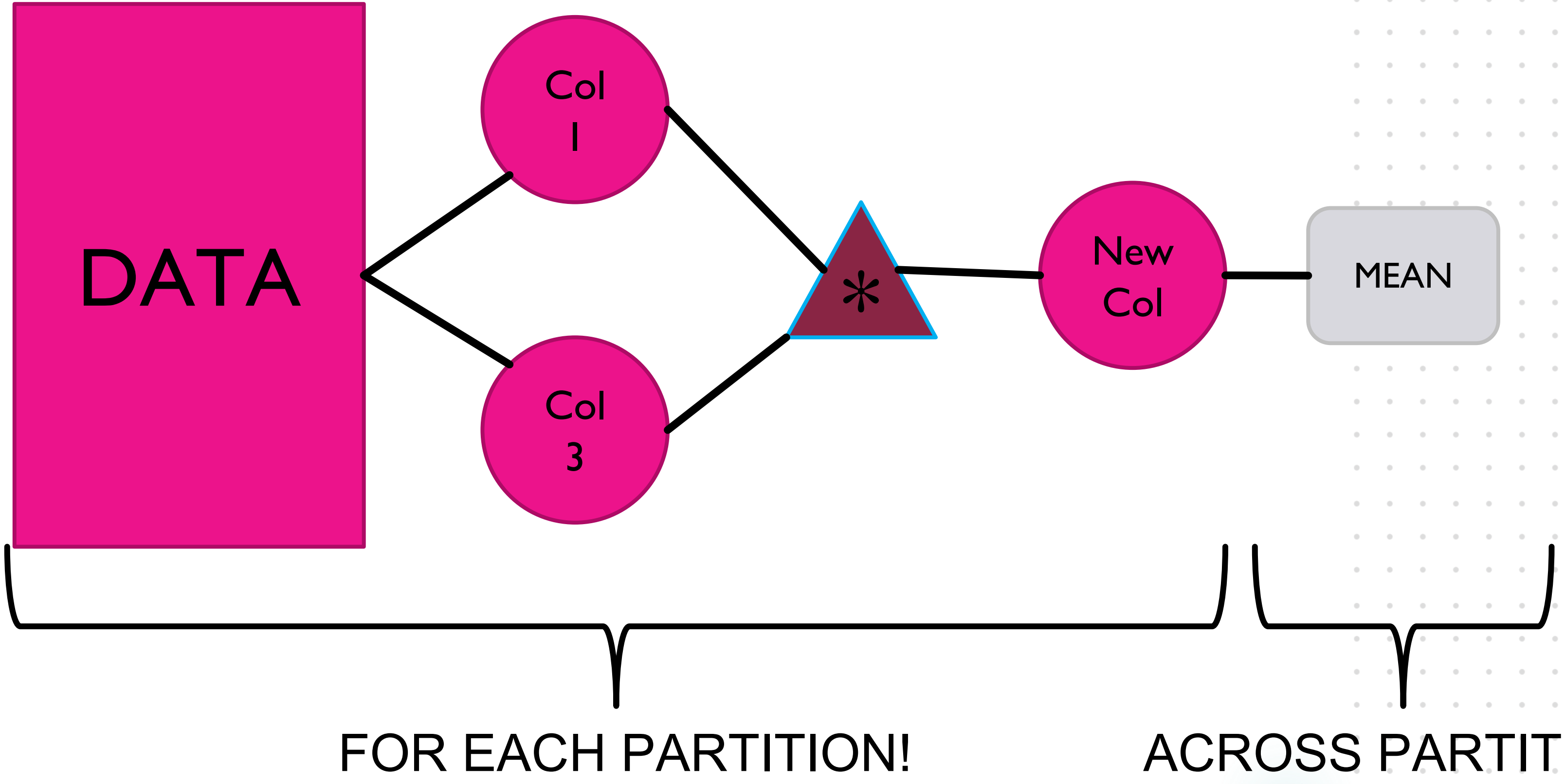
Me: Then let’s take the mean of that new thing.

Dask: Sure thing. Want the answer now?

Dask’s Checklist

- Multiply columns 1 & 3
- Mean of new column
- Return answer back to user

Dask – Making a Plan



DAGs

- Dask creates plans like that one by creating Directed Acyclic Graphs (DAGs) which are task graphs that flow data through processes.
- These DAGs can then be parallelized by different cores as available, and also work as a single plan to optimize calculation across big data.
- We'll talk more about DAGs during our Spark discussion.

Lazy Evaluation

- To make DAGs, Dask has two different types of behaviors: Transformations and computations.
- So if we wanted to filter some data we’d do something like:

```
df2 = df[df[‘sqft’] > 1000]
# df2 is currently a plan of action

df2.compute() # now it’s a dataframe
```

Dask also…

- … integrates directly with SkLearn/Numpy
- … can manage parallelization through it’s own cluster creation method
- … allows streaming data
- … can do parallelized machine learning

Dask and ML

- SkLearn will convert Dask data into a single numpy array behind the scenes, which means if your data is too big for RAM, you might not be able to SkLearn
- If you want to do ML on the partitions you can either manually load each partition and use a partial fit, or use Dask_ML.

Dask - Summary

- Extends the ”workable data” size from fits in RAM to fits on disk, by partitioning data
- Optimizes calculations via DAGs and lazy evaluation
- Maintains a similar API and feel to Pandas
- Provided an easy parallelization tool for Pandas-able data

QUESTIONS?

Cites

- Pixel speech bubbles: https://pixelspeechbubble.com/