

crossteam

A Python package
for joining large datasets
across servers
efficiently

BERT WAGNER

Data Scientist, YouTuber, Blogger

Code and slides:
bertwagner.com/crosstream

Contact:
bertwagner.com
bertwagner@bertwagner.com
[@bertwagner](https://twitter.com/bertwagner)

DATA WITH BERT



MOVING MOUNTAINS

Joining datasets across servers is not easy

- Linked servers, PolyBase, etc...
- Move small data to the large data
- ETL servers



A large shark, likely a great white, is shown from a side-on perspective swimming through deep blue ocean water. The shark's body is dark, contrasting with the lighter blue of the water above it. Sunlight filters down from the surface in bright rays, creating a dramatic play of light and shadow on the shark's skin and illuminating the water around it.

WARNING

You should use the previous options if
they are available

What follows is a great option when there
is nothing else you can do

CROSSTREAM

Install from PyPI:

```
pip install crosstream
```

Or from source:

```
git clone https://github.com/bertwagner/crosstream.git
cd crosstream
pip install .
```

The screenshot shows the GitHub repository page for 'crosstream' by 'bertwagner'. The repository is public and has 1 contributor. The README.md file contains the following text:

crosstream   

This Python package helps join large datasets across servers efficiently. It accomplishes this by streaming the data, allowing it to:

- read each dataset only once
- not need to store the complete datasets in memory

This is particularly helpful when your datasets are larger than the available memory on your machine or when you want to minimize network reads.

This package supports CSV and pyodbc datasources.

Installation

```
git clone https://github.com/bertwagner/crosstream.git
```

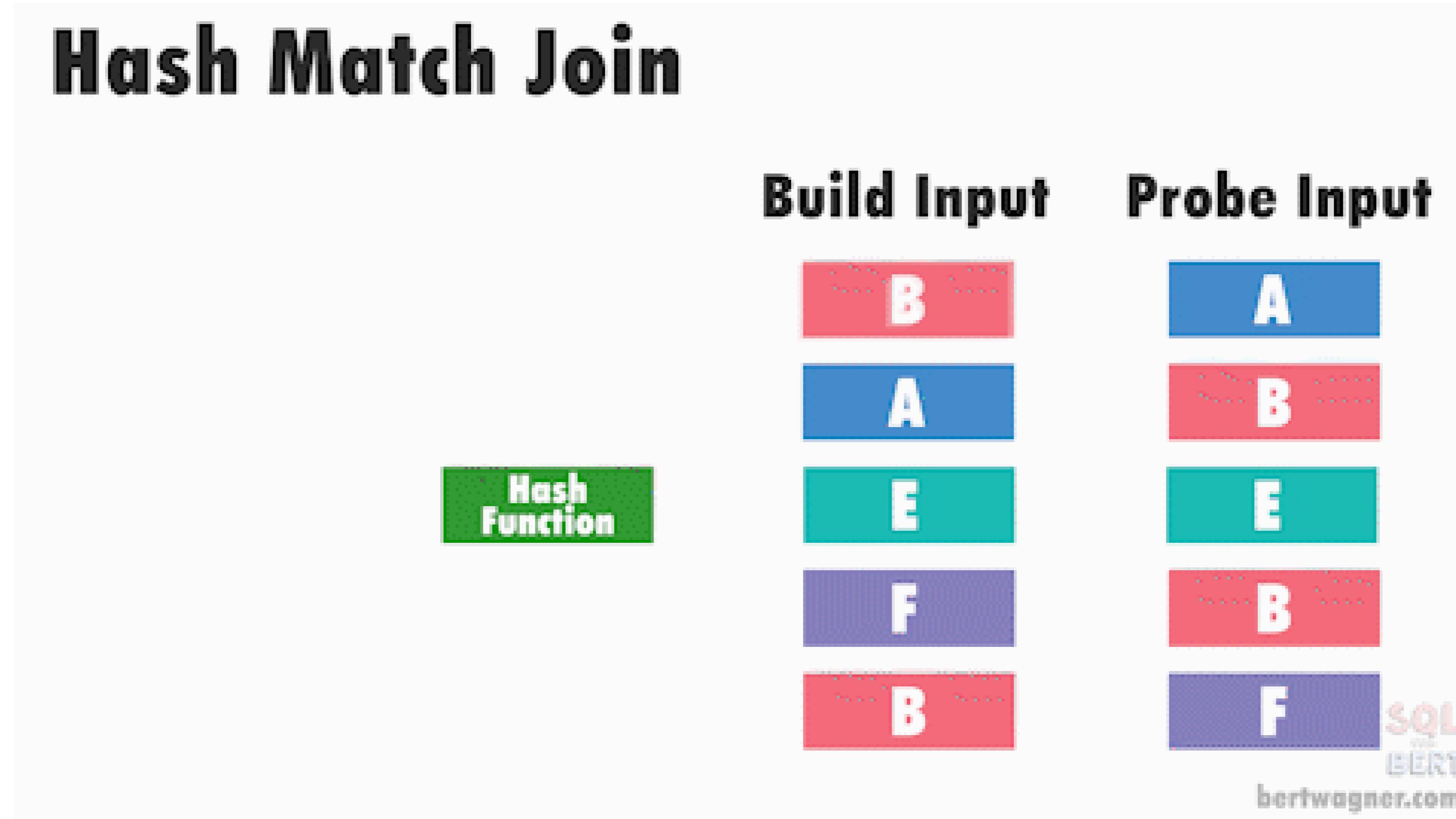
github.com/bertwagner/crosstream

FEATURES

- Reads each dataset only once
- Does not store full datasets in memory or on disk
- Works on CSV files or anything ODBC
- Allows data transformations before equality comparisons
- Allows data transformations after matching

INTERNAL

Hash Match Join



<https://bertwagner.com/posts/hash-match-join-internals/>

DEMO:

BASIC USAGE

```
from crosstream.hash_join import HashJoin
from crosstream.data_types import CSVData,QueryData
import csv

file1 = 'small_dataset.csv'
file2 = 'large_dataset.csv'

# join using column indexes or column names
c1=CSVData(file1,True,[0,1])
c2=CSVData(file2, True, ['col1','col2'])

# initialize the type of join to perform.
h=HashJoin()

# specify the output file
with open('joined_output.csv', 'w') as f:
    w =csv.writer(f)

    # write header column names
    w.writerow(c1.column_names + c2.column_names)

    for row_left,row_right in h.inner_join(c1,c2):
        # write matched results to our joined_output.csv
        w.writerow(row_left + row_right)
```

DEMO:

BASIC USAGE

```
from crosstream.hash_join import HashJoin
from crosstream.data_types import CSVData,QueryData
import csv

file1 = 'small_dataset.csv'
file2 = 'large_dataset.csv'

# join using column indexes or column names
c1=CSVData(file1,True,[0,1])
c2=CSVData(file2, True, ['col1','col2'])

# initialize the type of join to perform.
h=HashJoin()

# specify the output file
with open('joined_output.csv', 'w') as f:
    w =csv.writer(f)

    # write header column names
    w.writerow(c1.column_names + c2.column_names)

    for row_left,row_right in h.inner_join(c1,c2):
        # write matched results to our joined_output.csv
        w.writerow(row_left + row_right)
```

DEMO:

BASIC USAGE

```
from crosstream.hash_join import HashJoin
from crosstream.data_types import CSVData,QueryData
import csv

file1 = 'small_dataset.csv'
file2 = 'large_dataset.csv'

# join using column indexes or column names
c1=CSVData(file1,True,[0,1])
c2=CSVData(file2, True, ['col1','col2'])

# initialize the type of join to perform.
h=HashJoin()

# specify the output file
with open('joined_output.csv', 'w') as f:
    w =csv.writer(f)

    # write header column names
    w.writerow(c1.column_names + c2.column_names)

    for row_left,row_right in h.inner_join(c1,c2):
        # write matched results to our joined_output.csv
        w.writerow(row_left + row_right)
```

DEMO:

BASIC USAGE

```
from crosstream.hash_join import HashJoin
from crosstream.data_types import CSVData,QueryData
import csv

file1 = 'small_dataset.csv'
file2 = 'large_dataset.csv'

# join using column indexes or column names
c1=CSVData(file1,True,[0,1])
c2=CSVData(file2, True, ['col1','col2'])

# initialize the type of join to perform.
h=HashJoin()

# specify the output file
with open('joined_output.csv', 'w') as f:
    w =csv.writer(f)

    # write header column names
    w.writerow(c1.column_names + c2.column_names)

    for row_left,row_right in h.inner_join(c1,c2):
        # write matched results to our joined_output.csv
        w.writerow(row_left + row_right)
```

DEMO: CUSTOM JOIN KEYS

```
# define a function for joining on criteria that is modified  
before insert into hash table  
def custom_join_key(row,indices):  
    # calculate the hash of join values  
    join_values = []  
    for col_index in indices:  
        # here we transform our join key, removing spaces  
        join_values.append(str(row[col_index]).replace(' ',''))  
    join_key = ''.join(join_values)  
  
    return join_key  
  
...  
for row_left, row_right in  
h.inner_join(c1,c2,override_build_join_key=custom_join_key):  
    ...
```

DEMO: CUSTOM JOIN KEYS

```
# define a function for joining on criteria that is modified  
before insert into hash table  
def custom_join_key(row,indices):  
    # calculate the hash of join values  
    join_values = []  
    for col_index in indices:  
        # here we transform our join key, removing spaces  
        join_values.append(str(row[col_index]).replace(' ', ''))  
    join_key = ''.join(join_values)  
  
    return join_key  
  
...  
for row_left, row_right in  
h.inner_join(c1,c2,override_build_join_key=custom_join_key):  
    ...
```

DEMO: CUSTOM JOIN KEYS

```
# define a function for joining on criteria that is modified  
before insert into hash table  
def custom_join_key(row,indices):  
    # calculate the hash of join values  
    join_values = []  
    for col_index in indices:  
        # here we transform our join key, removing spaces  
        join_values.append(str(row[col_index]).replace(' ', ''))  
    join_key = ''.join(join_values)  
  
    return join_key  
  
...  
for row_left, row_right in  
h.inner_join(c1,c2,override_build_join_key=custom_join_key):  
    ...
```

DEMO: CUSTOM MATCH PROCESSING

```
# define a function for performing additional transformations
# or adding additional outputs before the columns are returned
def custom_process_matched_hashes(bucket_row,probe_row,
bucket_join_column_indexes, probe_join_column_indexes):
    # adding a new column indicating the weights of these
    matches are equal to 1
    weight=1.0
    return tuple(bucket_row),tuple(probe_row),(weight,)

...
for row_left,row_right,weight in
h.inner_join(c1,c2,override_process_matched_hashes=custom_proce
ss_matched_hashes):
    ...
```

DEMO: CUSTOM MATCH PROCESSING

```
# define a function for performing additional transformations
# or adding additional outputs before the columns are returned
def custom_process_matched_hashes(bucket_row,probe_row,
bucket_join_column_indexes, probe_join_column_indexes):
    # adding a new column indicating the weights of these
    matches are equal to 1
    weight=1.0
    return tuple(bucket_row),tuple(probe_row),(weight,)

...
for row_left,row_right,weight in
h.inner_join(c1,c2,override_process_matched_hashes=custom_proce
ss_matched_hashes):
    ...
```

DEMO: CUSTOM MATCH PROCESSING

```
# define a function for performing additional transformations
# or adding additional outputs before the columns are returned
def custom_process_matched_hashes(bucket_row,probe_row,
bucket_join_column_indexes, probe_join_column_indexes):
    # adding a new column indicating the weights of these
    matches are equal to 1
    weight=1.0
    return tuple(bucket_row),tuple(probe_row),(weight,)

...
for row_left,row_right,weight in
h.inner_join(c1,c2,override_process_matched_hashes=custom_proce
ss_matched_hashes):
    ...
```

THANK YOU!



@bertwagner



bertwagner.com



youtube.com/DataWithBert



bertwagner@bertwagner.com