Usage:

singularity pull shub://pvelesko/singularity\_files:datadog

singularity run ./datadog <lat> <lon> <range>

Problem Approach

Traversal Methods:

1. Random traversal

Pros:

easy to scale

no additional mem requirements

Cons:

for larger radius have to run for a while

2. Random traversal of nearest neighbors

Pros:

no training time

easy to scale

Cons:

Might fail for larger datasets

Have to compute neighbor table

3. Q-Learning

Pros:

4. Deep Q-Learning

Pros:

Allows for effective exploration of larger spaces

No need to store Q-table

Cons:

Needs to be trained

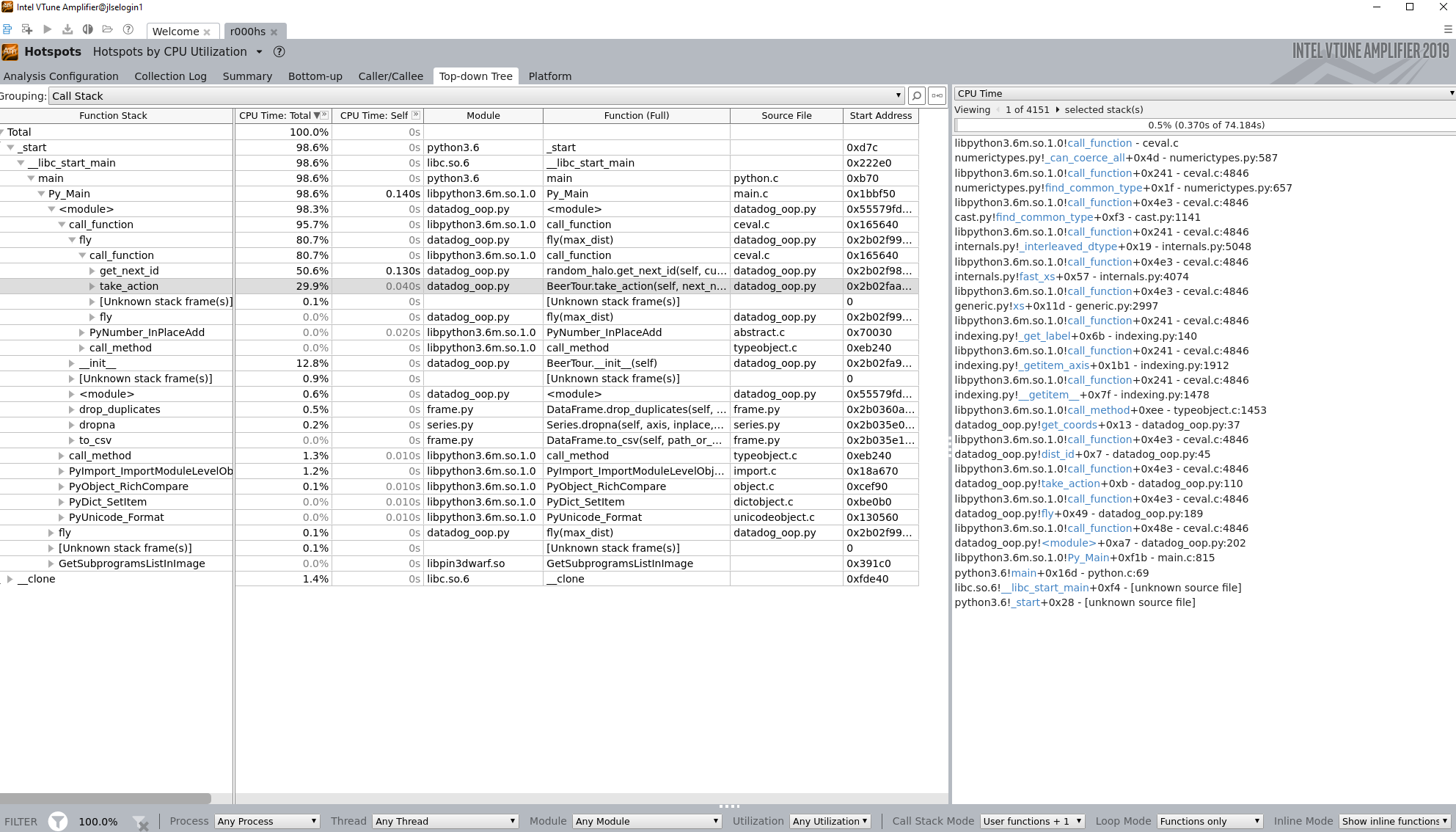
Optimizations:

1. Reduced potential breweries to max round trip

2. Neighbor list precompute (memory requirements O(n^2))

Optimization

I profiled the code using Vtune Amplifier 2019:



250s to complete 500 iterations.

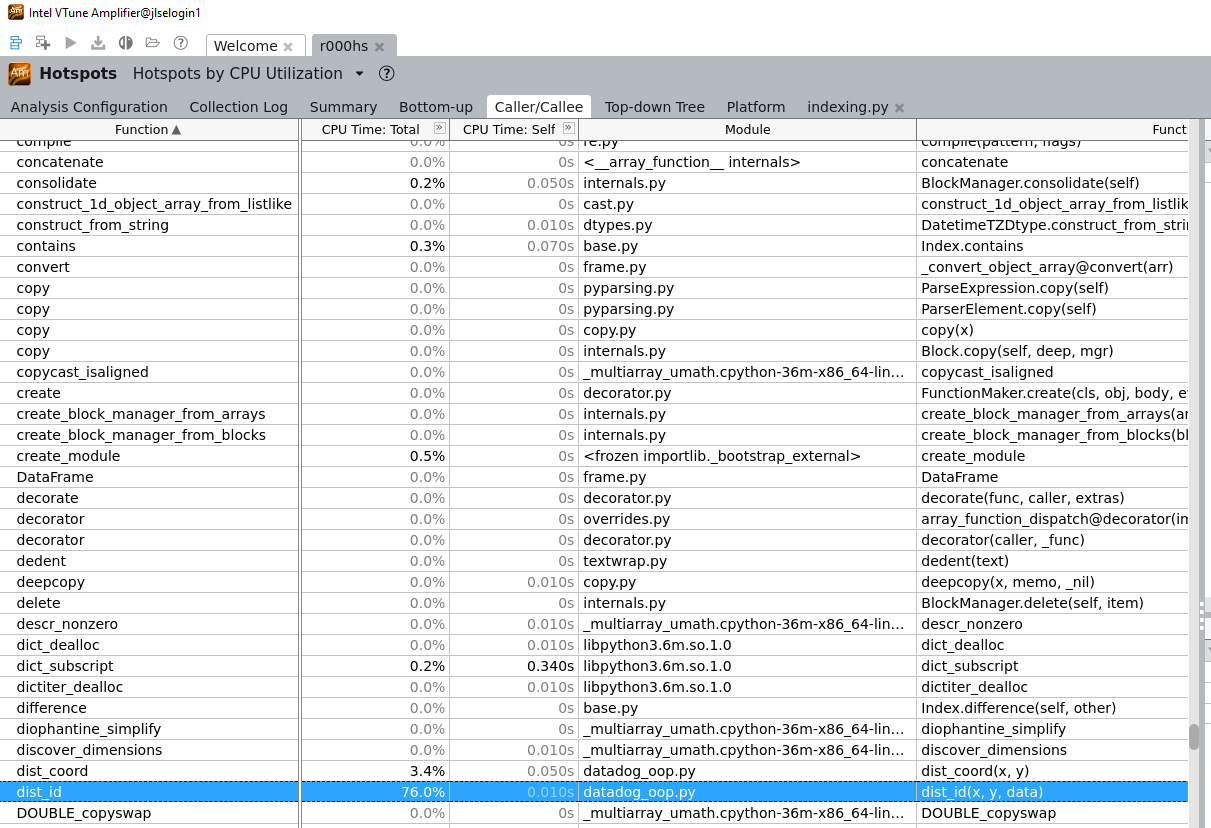
Pathfinding is taking 80% of the time for 500 iterations, of that 50% is get\_next\_id and 30% take\_action

Get\_next\_id:

Most of the time is spent in pandas/python layer retrieving the actual coordinates for dist\_id. Seems like we must convert our dataframes to numpy arrays since those are vectorized and implemented in C.

Take action:

Looks like bound by dist\_id as well..

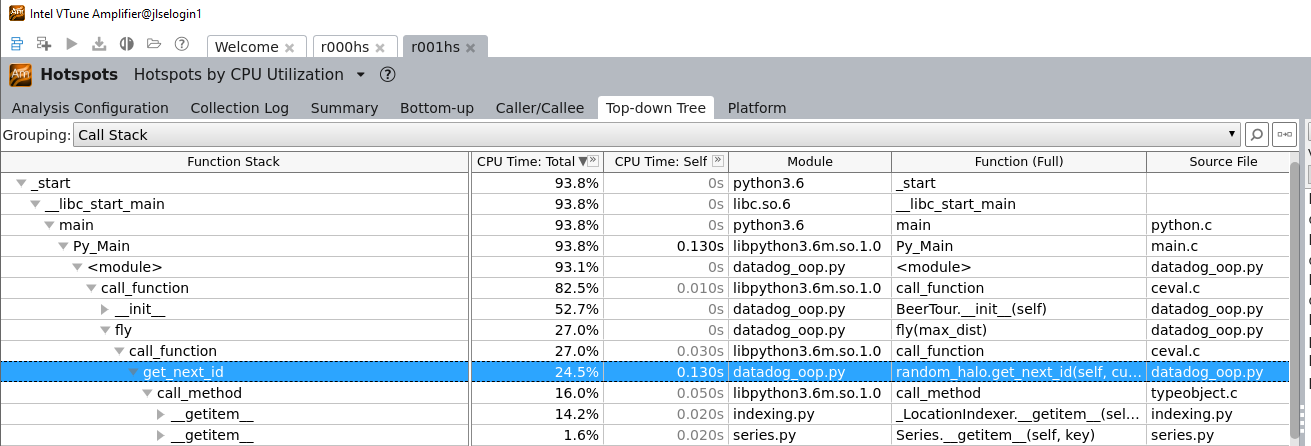


Call/callee view shows dist\_id taking 76% of overall runtime.

Optimization: Store coordinates as numpy arrays instead of dataframes

Result: Time for iterating through 500 iterations 200s -> 15s = 13x speedup.

2nd Profiling Run:



Get\_next\_id is still taking a while since I’m storing neighbor lists as dataframes. Seems like pandas has no high performance back-end.

Optimization: convert to neighbor list to numpy