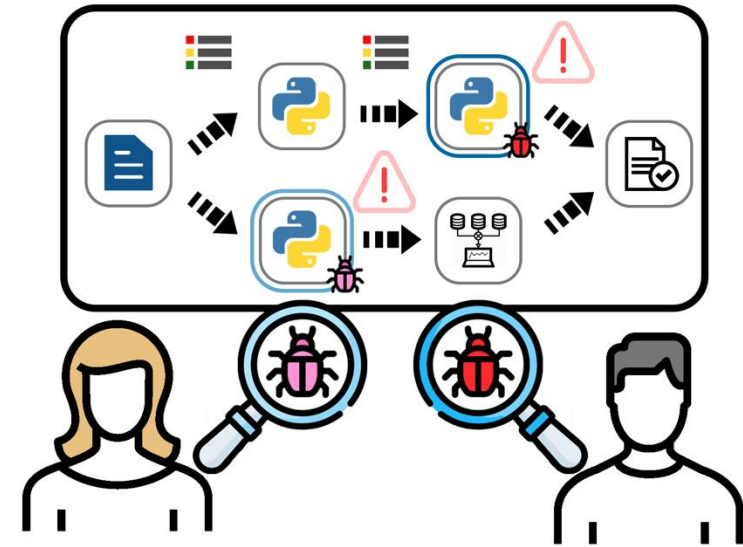


Udon: Efficient Debugging of User-Defined Functions in Big Data Systems with Line-by-Line Control

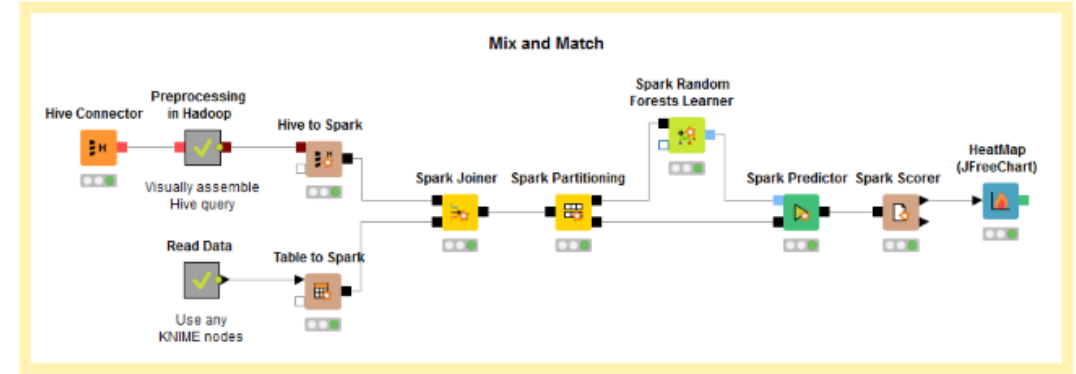
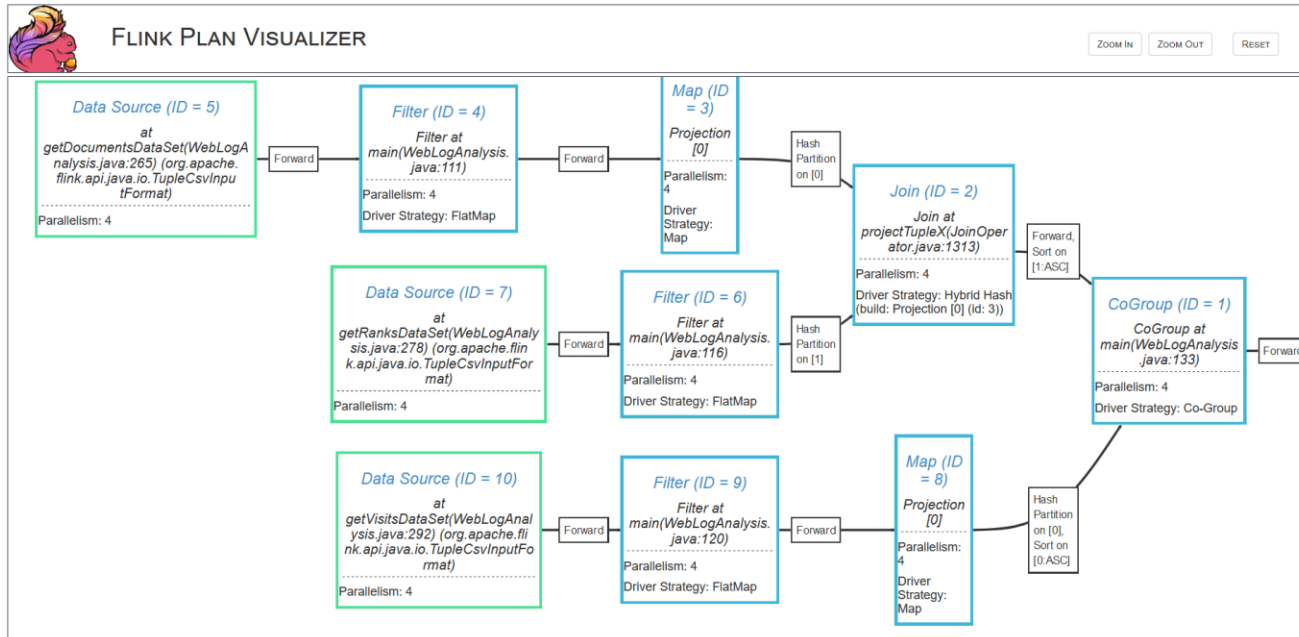
Yicong Huang, Zuozhi Wang, Chen Li
University of California, Irvine
Information System Group (ISG)



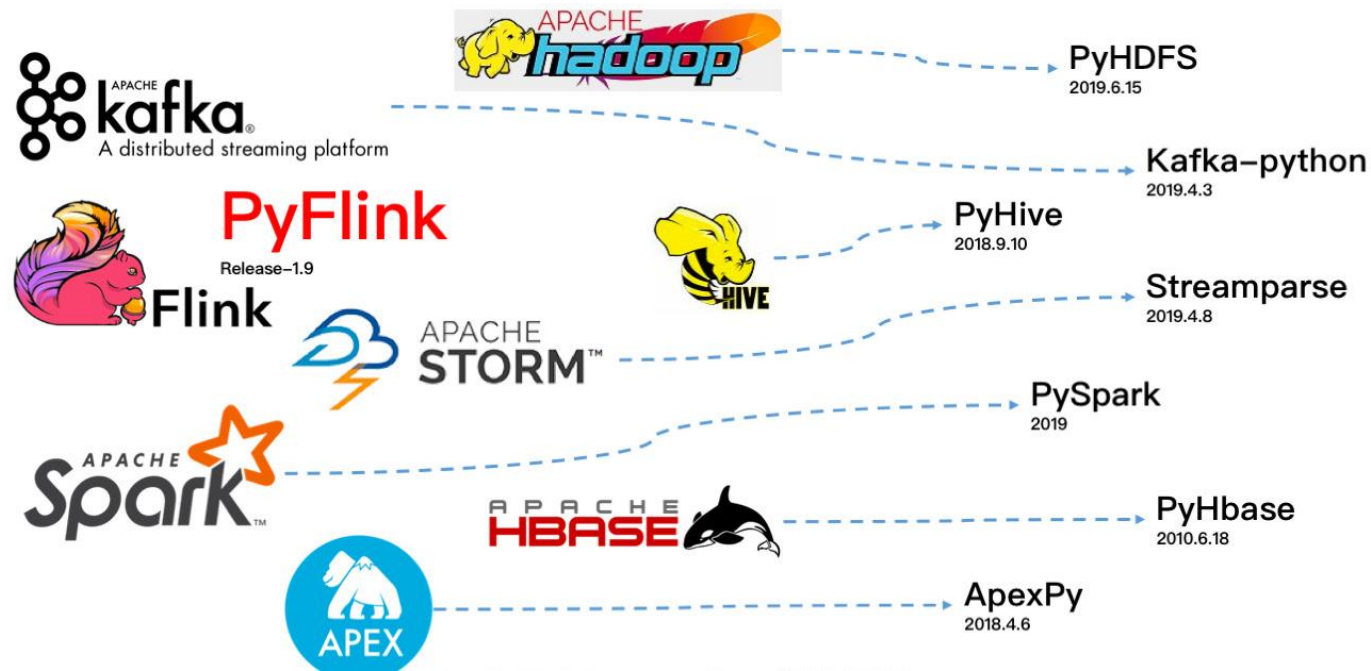
UCIRVINE



Dataflow tasks can be complex



User-defined Functions are common!



```
@udf(result_type='BIGINT')
def add(i, j):
    return i + j
```

```
class Top2(TableAggregateFunction):
```

```
    def emit_value(self, accumulator):
        yield Row(accumulator[0])
        yield Row(accumulator[1])
```

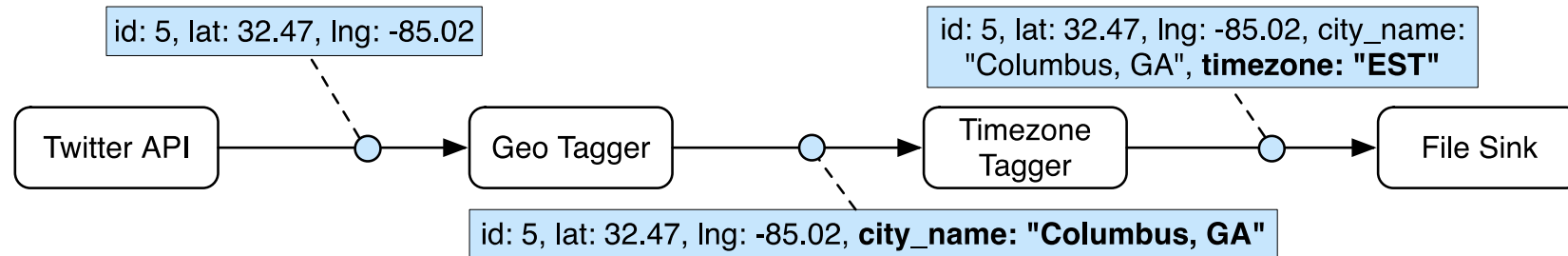
```
    def create_accumulator(self):
        return [None, None]
```

```
    def accumulate(self, accumulator, row):
        if row[0] is not None:
            if accumulator[0] is None or row[0] > accumulator[0]:
                accumulator[1] = accumulator[0]
                accumulator[0] = row[0]
            elif accumulator[1] is None or row[0] > accumulator[1]:
                accumulator[1] = row[0]
```

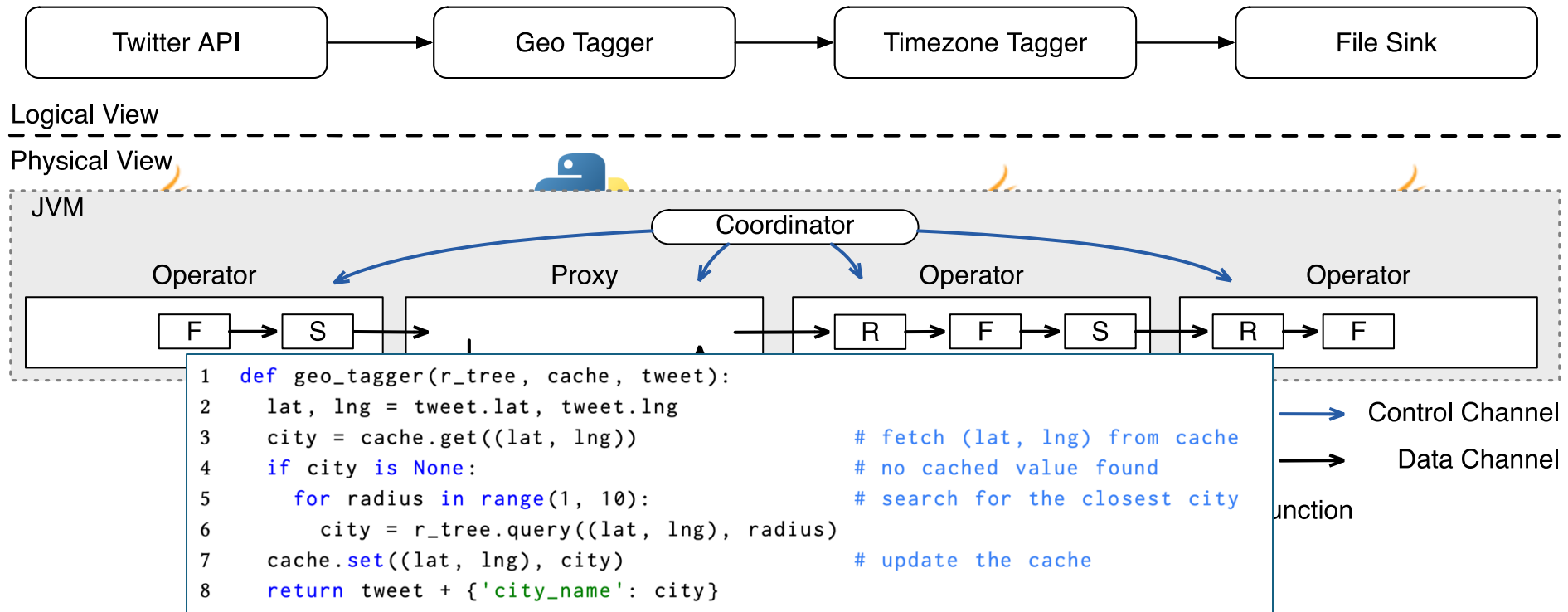
```
    def get_accumulator_type(self):
        return 'ARRAY<BIGINT>'
```

```
    def get_result_type(self):
        return 'ROW<a BIGINT>'
```

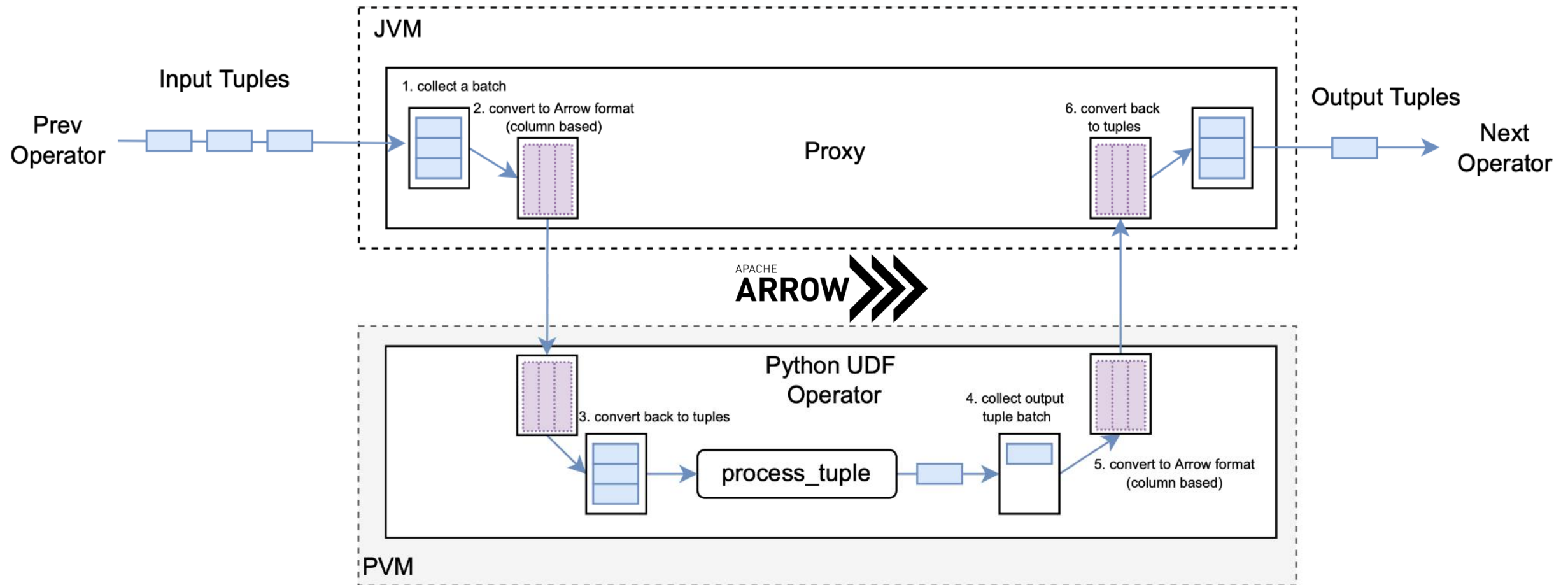
Consider a running example workflow



Execution of the example workflow



Data transformation between language VMs





Runtime Bugs in UDF code

Geo Tagger

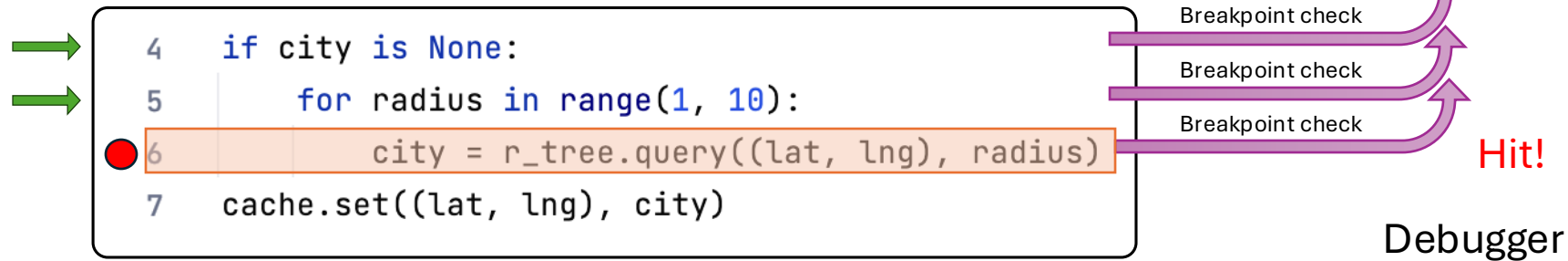
```
1 def geo_tagger(r_tree, cache, tweet):
2     lat, lng = tweet.lat, tweet.lng
3     city = cache.get((lat, lng))           # fetch (lat, lng) from cache
4     if city is None:                     # no cached value found
5         for radius in range(1, 10):      # search for the closest city
6             city = r_tree.query((lat, lng), radius)
7     cache.set((lat, lng), city)          # update the cache
8     return tweet + {'city_name': city}
```

- i) **Data errors:** If the lat field is absent from the input tweet, the `r_tree.query()` function may raise exceptions or return empty results.
- ii) **Code errors:** The for-loop gradually increases the search radius to find the closest city but does not terminate after a city is found. Thus the loop may continue and identify another city when a larger radius is provided, leading to an incorrect output.

Runtime Language Debuggers



Debugger Front-end

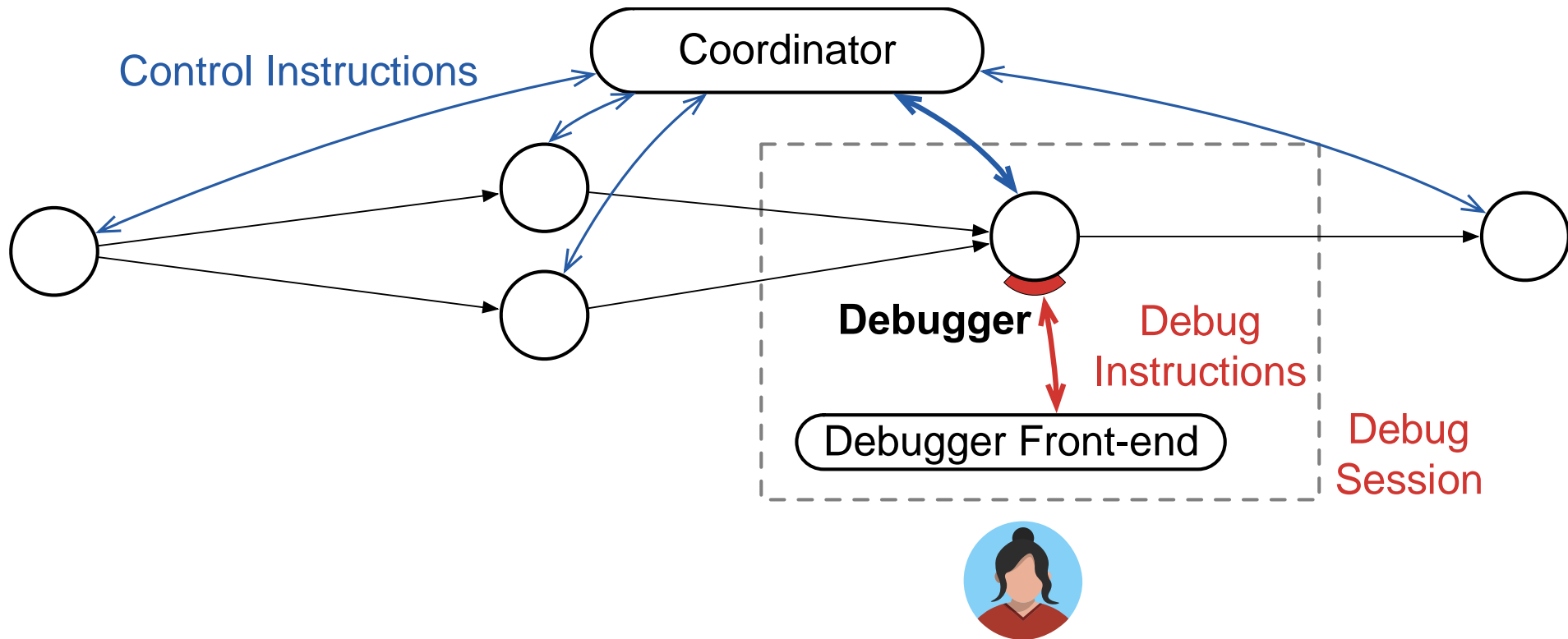


Python debuggers:

- pdb (The standard CPython debugger)
- PyDev.Debugger

How to integrate a language debugger into a data engine?

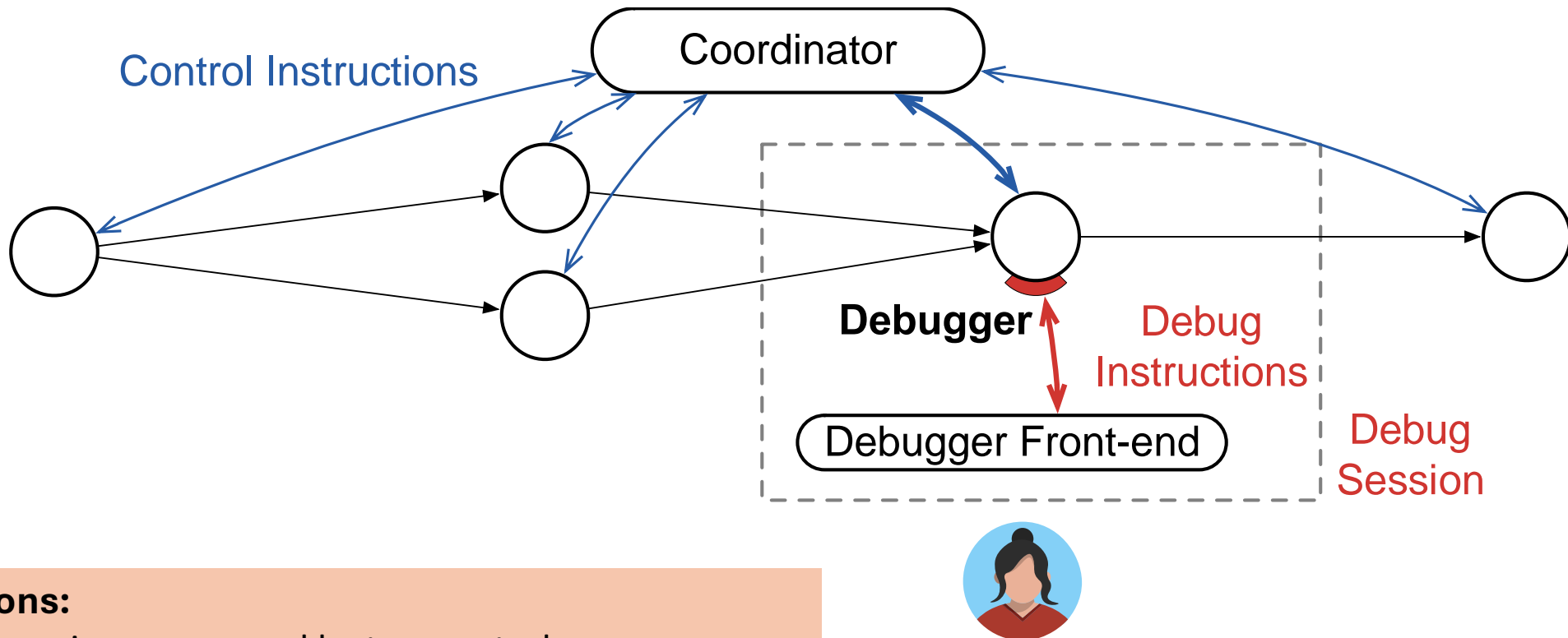
Approach 1: The Debugger Runs Separately from the Engine



PyFlink's recommended UDF debugging:

```
import pydevd_pycharm
pydevd_pycharm.settrace('localhost', port=6789)
```

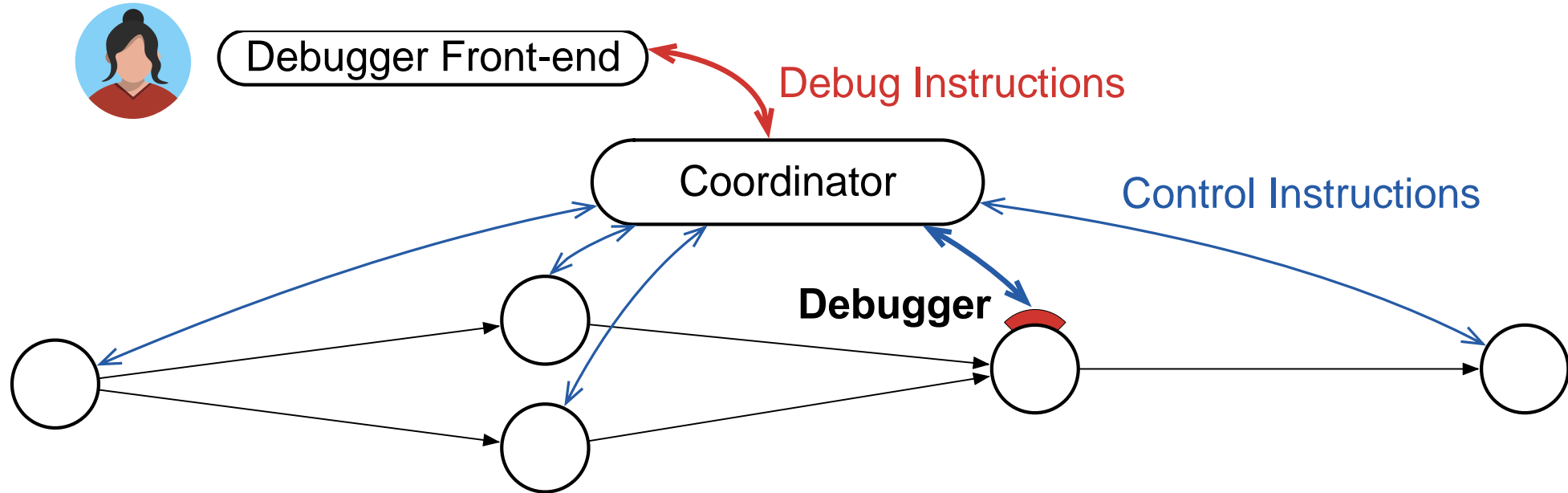
Approach 1: The Debugger Runs Separately from the Engine



Limitations:

1. Unresponsiveness caused by two control sources.
2. Uncontrolled workflow suspension.
3. Lack of synchronization between multiple debuggers.

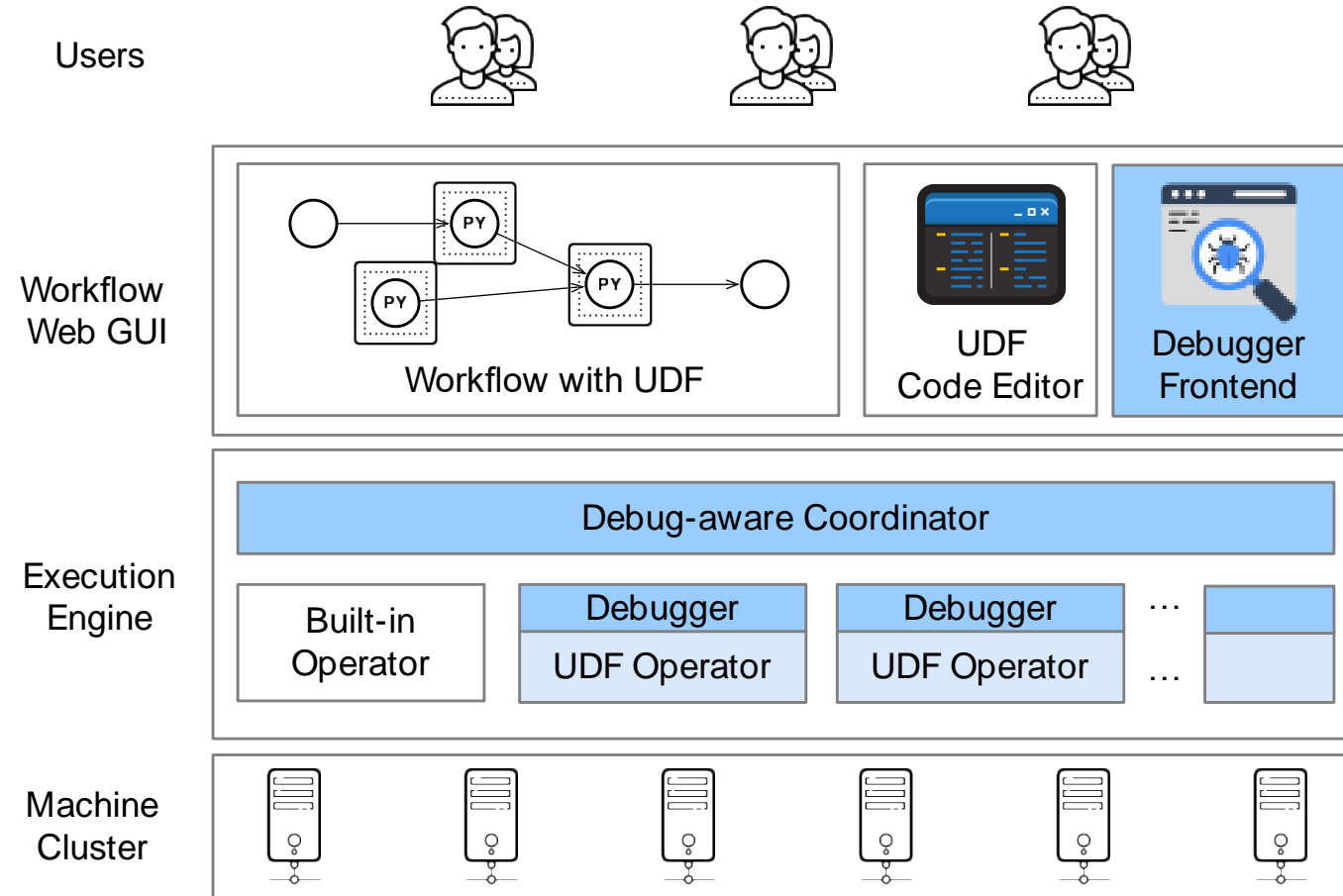
Approach 2: The Engine Controls the Debugger



Advantages:

1. A *single* source of control.
2. Coordinated workflow suspension.
3. Supporting synchronization between multiple debuggers.

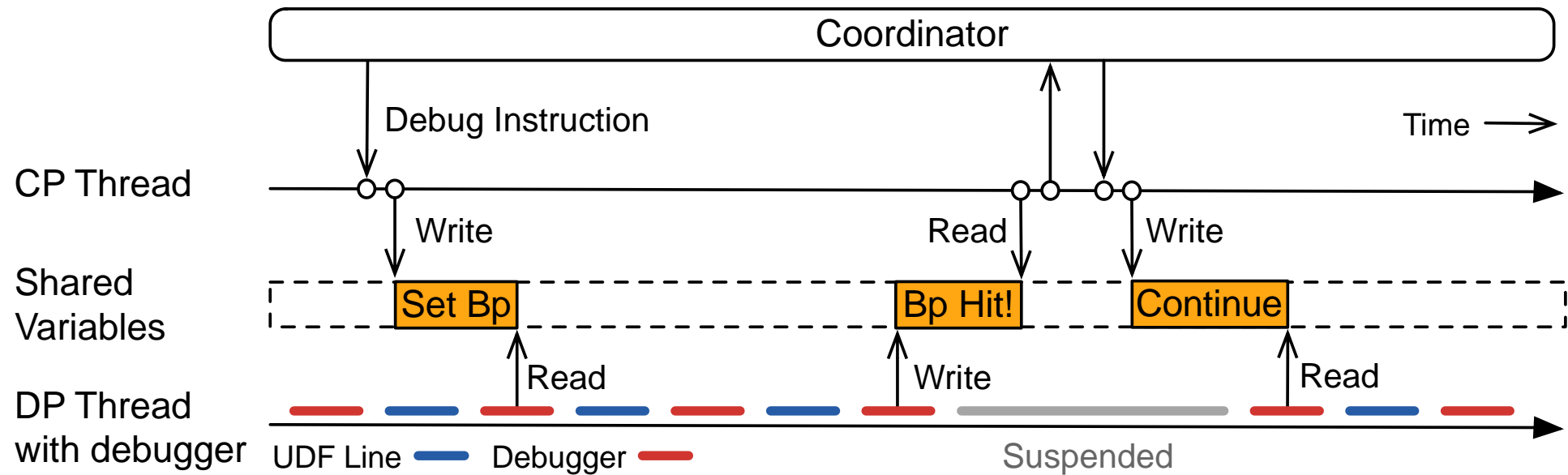
Debug-aware Coordinator



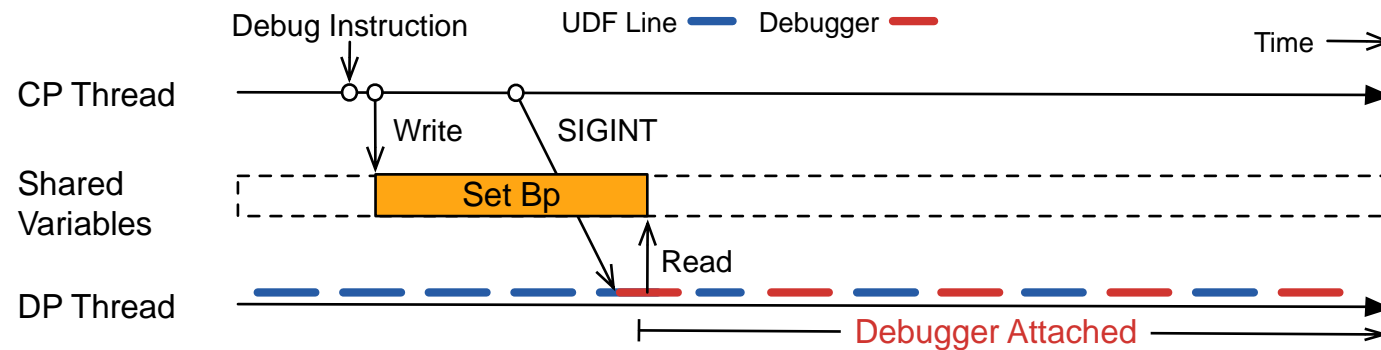
How to send a debug instruction to a UDF?

A Novel UDF Execution Model: Two-threaded execution model

To receive debug instructions dynamically.

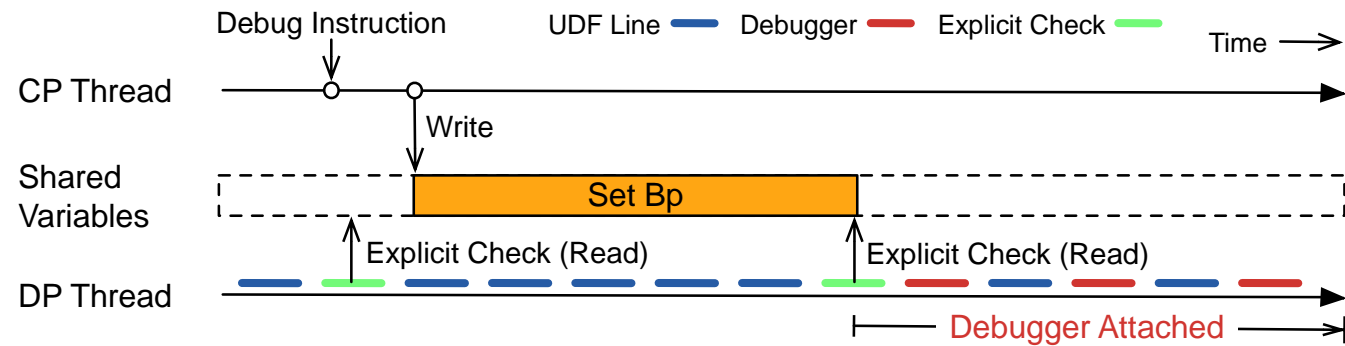


Pass debug instructions between threads



Signal-based (Forcibly)

Pass debug instructions between threads



Explicit check (Voluntary)

Supports all debug instructions

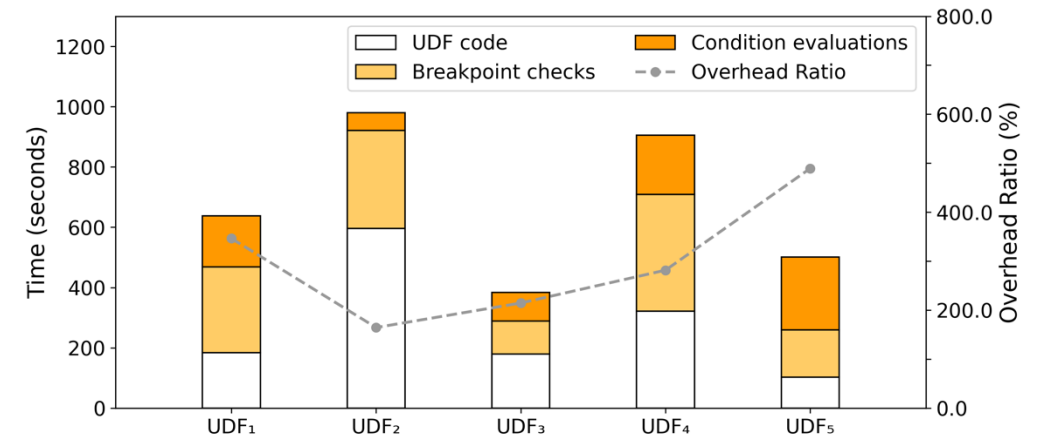
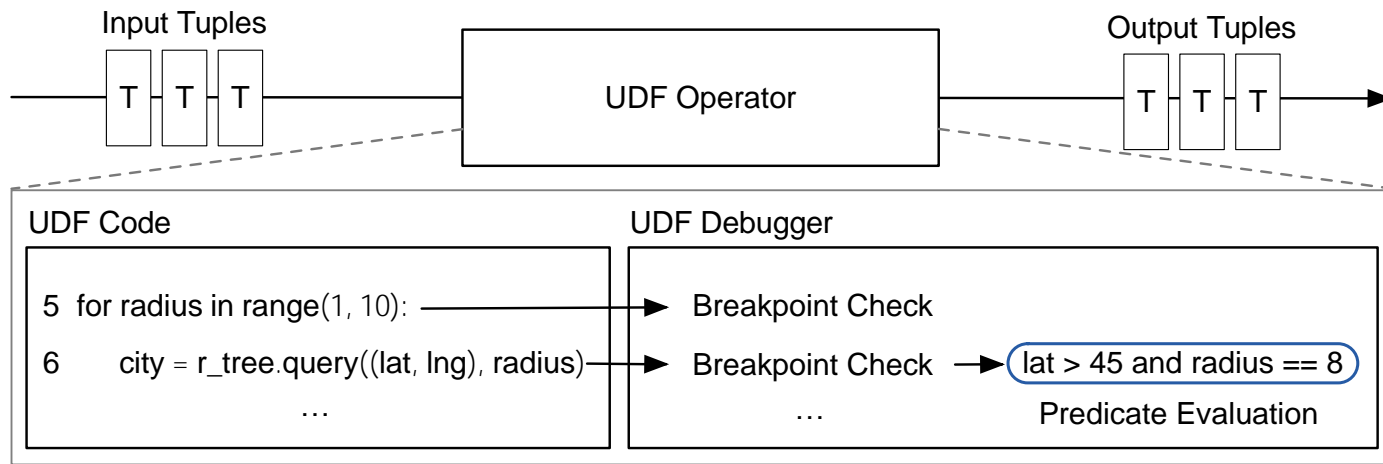
1. ``break`` (or ``b``): Set a breakpoint at a specified line or function.
2. ``continue`` (or ``c``): Resume execution until the next breakpoint.
3. ``next`` (or ``n``): Continue execution until the next line in the current function is reached.
4. ``step`` (or ``s``): Execute the current line and stop at the first possible occasion.
5. ``return`` (or ``r``): Continue execution until the current function returns.
6. ``list`` (or ``l``): Display the source code around the current line.
7. ``print`` (or ``p``): Evaluate and print the value of an expression.

...

As Python is an interpreted language, you can also dynamically **change a UDF code** and **update an operator state** using a language debugger.

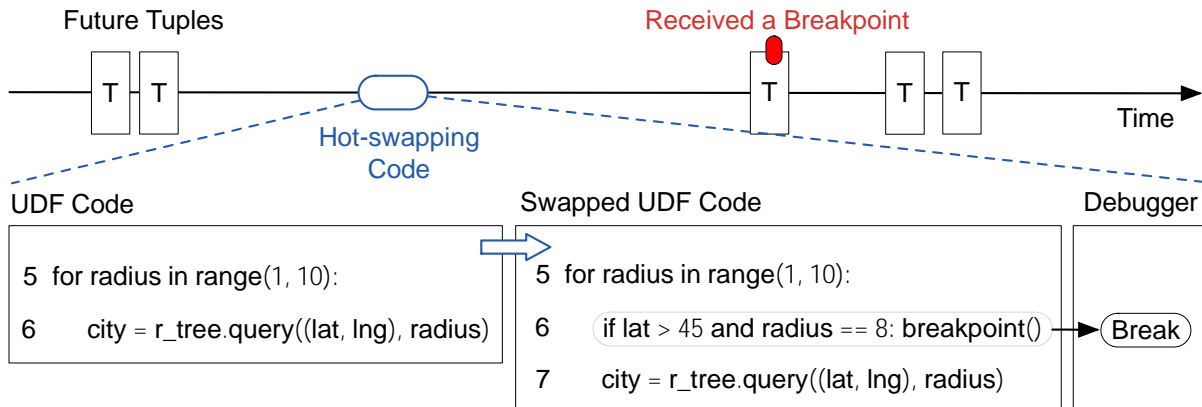
How to reduce debugging overhead?

Reducing the significant runtime overhead introduced by debuggers



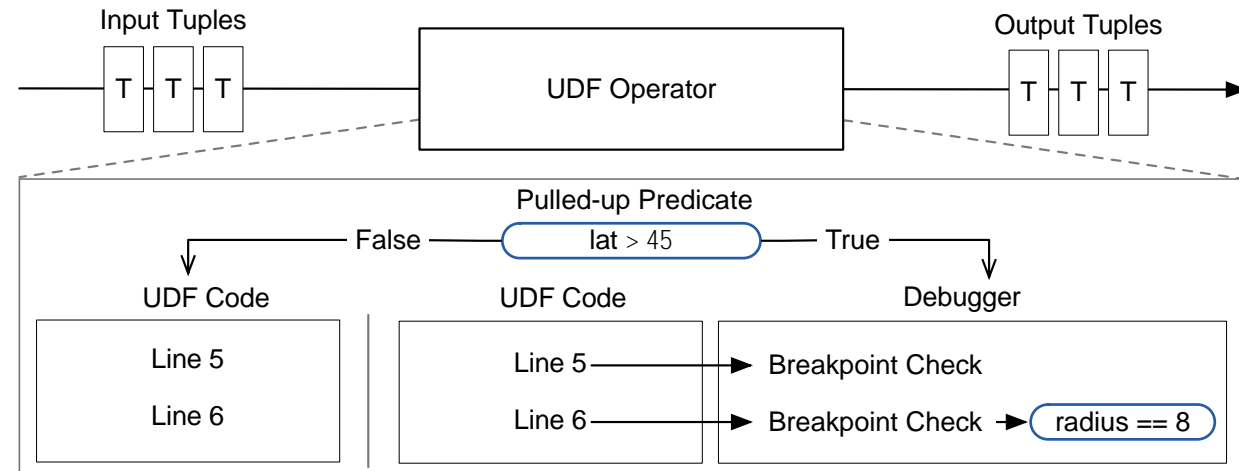
Depends on the UDF, we observe more than 2X-5X slowdown.

Optimizations to automatically detach debuggers



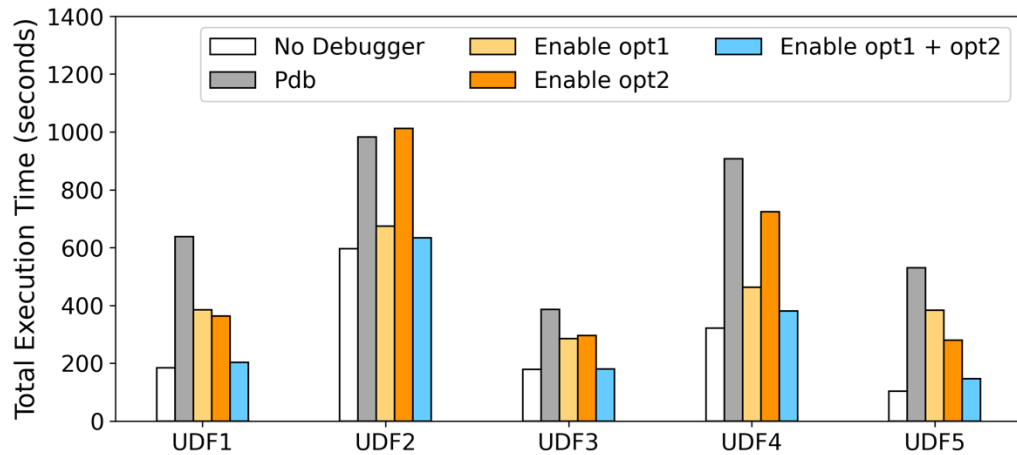
Optimization 1: Reducing Breakpoint Checks by Hot-swapping UDF Code

Optimization 2: Improving Evaluations by Pulling up Predicates

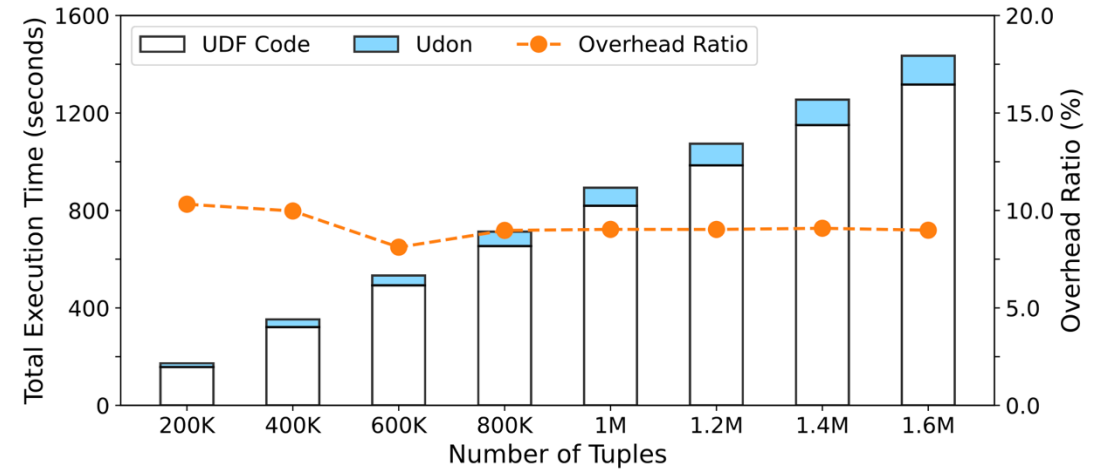


Experiments

- Twitter dataset.
- TPC-H dataset.
- COCO image dataset [19].



Less than 10% overall runtime overhead



Scale up with more data

Demonstration of Udon: Line-by-line Debugging of User-Defined Functions in Data Workflows

Come and checkout Udon in action on Texera, a GUI-based Workflow system for data science!

Group A

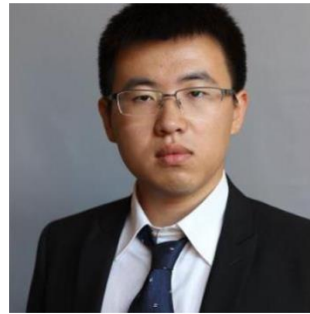
- Tuesday June 11 1:00 pm – 2:30 pm
Location: Europa
- Thursday June 13 5:00 pm – 6:30 pm
Location: Europa

The screenshot displays the Udon line-by-line debug interface within the Texera workflow system. The workflow consists of several steps: Target domains, Remove URL, Tokenizer, SVM Training, SVM inference, Training tweets, Convert to binary label, Tweets, Projection(d, time, text), and Windowed Average Sentiment. A 'Conditional Breakpoint' is set on line 19 of the Python UDF code, with the condition 'sentiment_score == 0.0'. The 'Debugger Frontend' shows the code and a 'Result Panel' with console output. The 'Debugger Instructions' panel shows 'All Workers' and 'Retry Tuple' options. The interface also displays 'Collaborative Debugging' with user names like 'Zuozhi Wang' and 'Yicong Huang'.

Udon: Efficient Debugging of User-Defined Functions in Big Data Systems with Line-by-Line Control



Yicong Huang



Zuozhi Wang



Chen Li



Acknowledgements: We thank Yiming Lin, Xi Lu, Shengquan Ni, the rest of the Texera team at UC Irvine, and the anonymous reviewers for their invaluable feedback. This work was funded by the National Science Foundation (NSF) under award III-2107150.

Yicong Huang, Zuozhi Wang, Chen Li | Udon

