Towards Efficient and Secure UDF Execution with BabelfishLib

Philipp M. Grulich

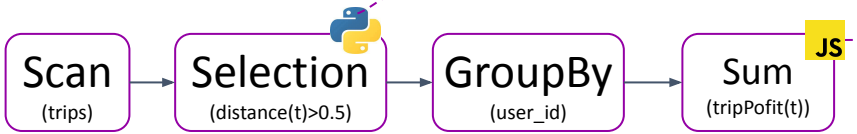
Technische Universität Berlin

Motivation

Polyglot Queries combine **relational operations** with **user-defined functions** in **different programming languages**.

Polyglot query in a bike sharing company:

How much profit do we earn per user?

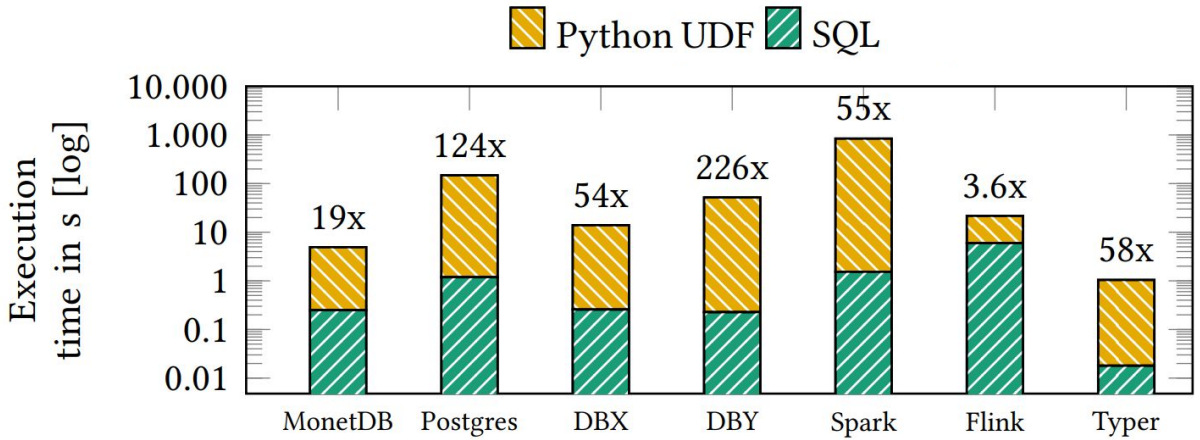



```sql
-- SQL Query --
SELECT sum(tripProfit(t))
FROM trips t
WHERE distance(t.start, t.end) > 0.5
GROUP BY t.user_id
```

```python
# Python distance function
from haversine import haversine, Unit
def distance(start, end):
 return haversine(start, end, Unit.KILOMETERS)
```

```javascript
// JavaScript profit function
function tripProfit(t){
 let price;
 if(t.date.before("2020-01-01")){
  price = t.duration * 0.5;
 } else {
  price = distance(t.start, t.end) * 0.3;
 }
 return t.hasVoucher ? price * -1: price;
}
```

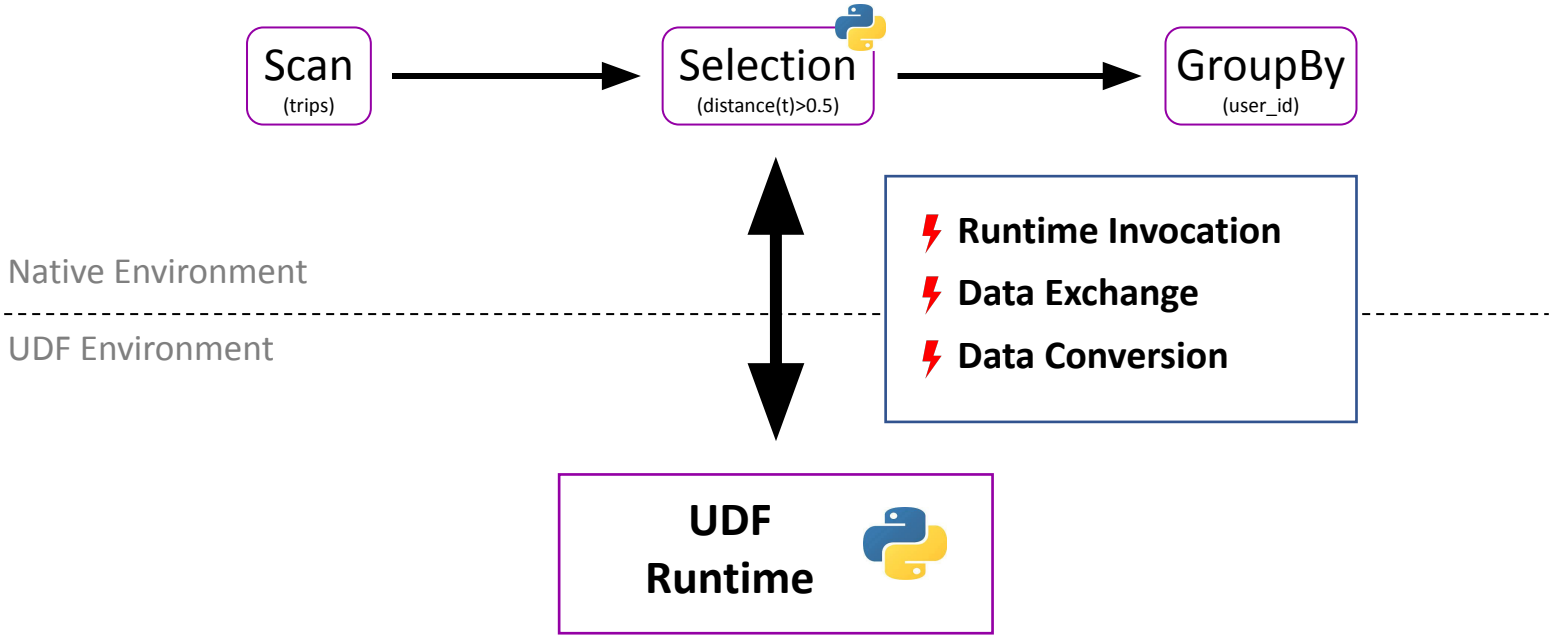
Limitations of state-of-the-art Systems



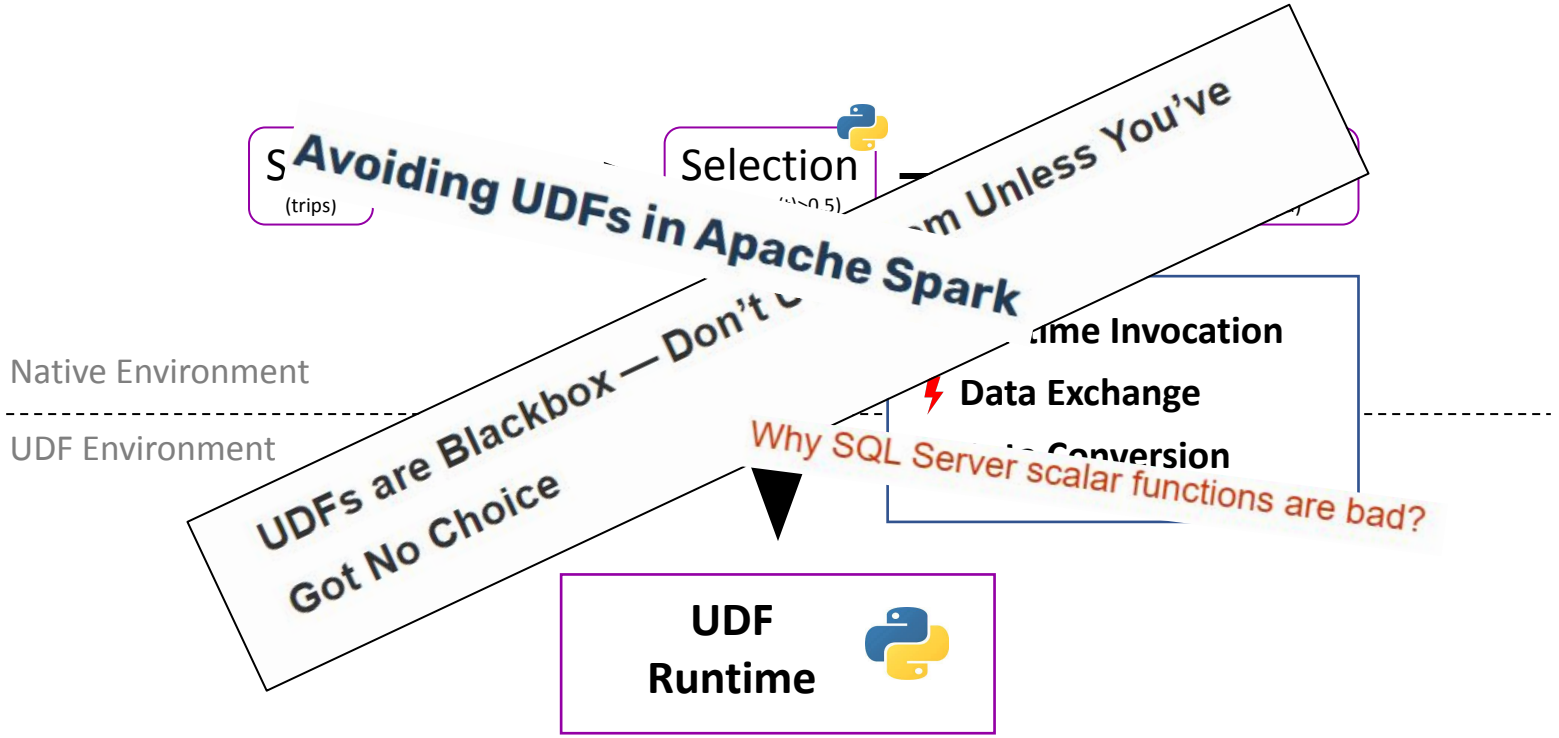
Overhead of TPC-H Query 6 with Python UDF on different systems.

Polyglot Queries cause a **significant overhead** on commercial and open source data processing systems.

Limitations of state-of-the-art Systems



Limitations of state-of-the-art Systems



Is this not already solved?

Logical Optimizations: Hueske et al. [VLDB 12], Venkatesh et al. [SIGMOD 16], Froid [VLDB 17], Aggify [SIGMOD 22], Duta et al. [CIDR 19, SIGMOD 20], Grizzly [CIDR 19], Haralampos et al. [SIGMOD 23]

Execution Strategies: Raasveldt et al. [SSDBM 16], Rosenfeld et al. [SoCC 17], Schüle et al. [SSDBM 19], Sichert et al. [VLDB 22]

Execution Engines: Trill [VLDB 14], Tupleware [VLDB 15], Gerenuk [SOSP 19], Tuplex [VLDB 21], Magpie [CIDR 21], YeSQL [VLDB 22], Babelfish [VLDB 22]

Several approaches tackle the problem.
Require significant amount of engineering.

Let's take a step back!

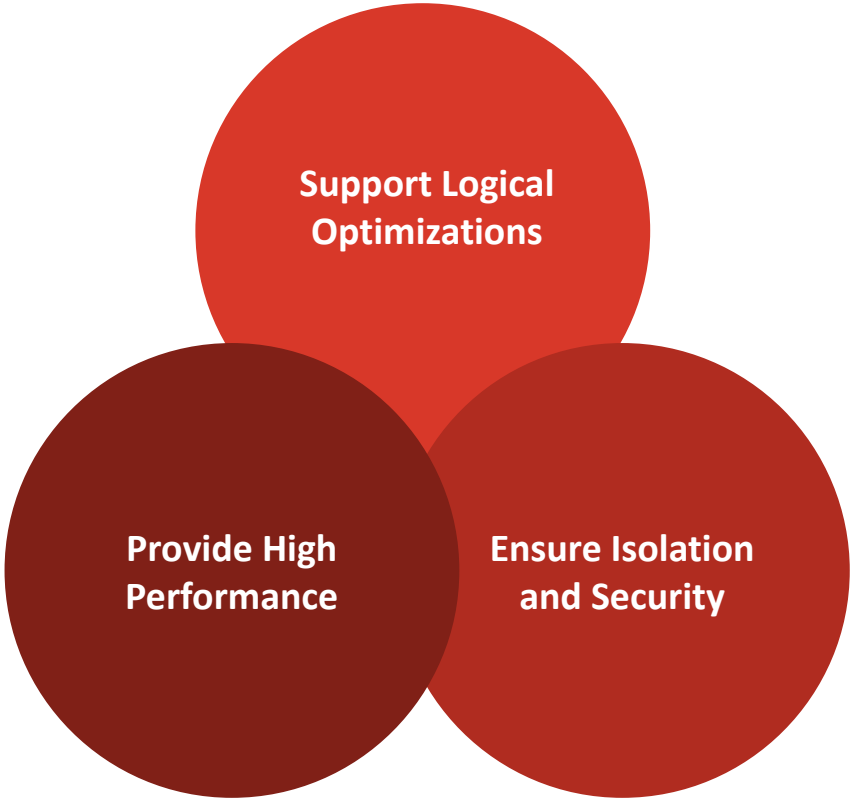
Could we provide efficient UDF execution as a library?

Goals for a UDF Accelerator

Translate UDFs to relation expressions.
Infer interesting-properties.
Enable cost-based optimizations.

Avoid data copies.
Integrate optimized language runtimes.
Specialize execution across operator boundaries

Isolation of different execution contexts.
Containerize faulty and malicious UDFs.



Towards BabelfishLib

BabelfishLib:

- **Efficient execution** of polyglot queries.
- Supporting **logical optimizations.**
- Ensuring **isolation** and **secure** execution.

Open Questions:

- Which logical optimizations are most beneficial?
- How to ensure portability across different systems?
- How to materialize intermediate data?
- Which level of isolation is required?

https://nebula.stream