VLDB 2022

Artifacts Available acm V1.1

Hu-Fu: Efficient and Secure Spatial Queries over Data Federation

Yongxin Tong[1], Xuchen Pan[1], Yuxiang Zeng[2], Yexuan Shi[1], Chunbo Xue[1], Zimu Zhou[3], Xiaofei Zhang[4], Lei Chen[2], Yi Xu[1], Ke Xu[1], Weifeng Lv[1]

[1]Beihang University

[2]The Hong Kong University of Science and Technology

[3]Singapore Management University

[4]University of Memphis

北京航空航天大學 BEIHANG UNIVERSITY   THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY   SMU SINGAPORE MANAGEMENT UNIVERSITY   THE UNIVERSITY OF MEMPHIS

Outline

- Background
- Problem Statement
- System Design
- Evaluations
- Conclusion

Widespread Applications of Spatial Queries

Taxi Calling

Logistics Planning

Map Service

Contact Tracing

Quality of service depends on access to big data

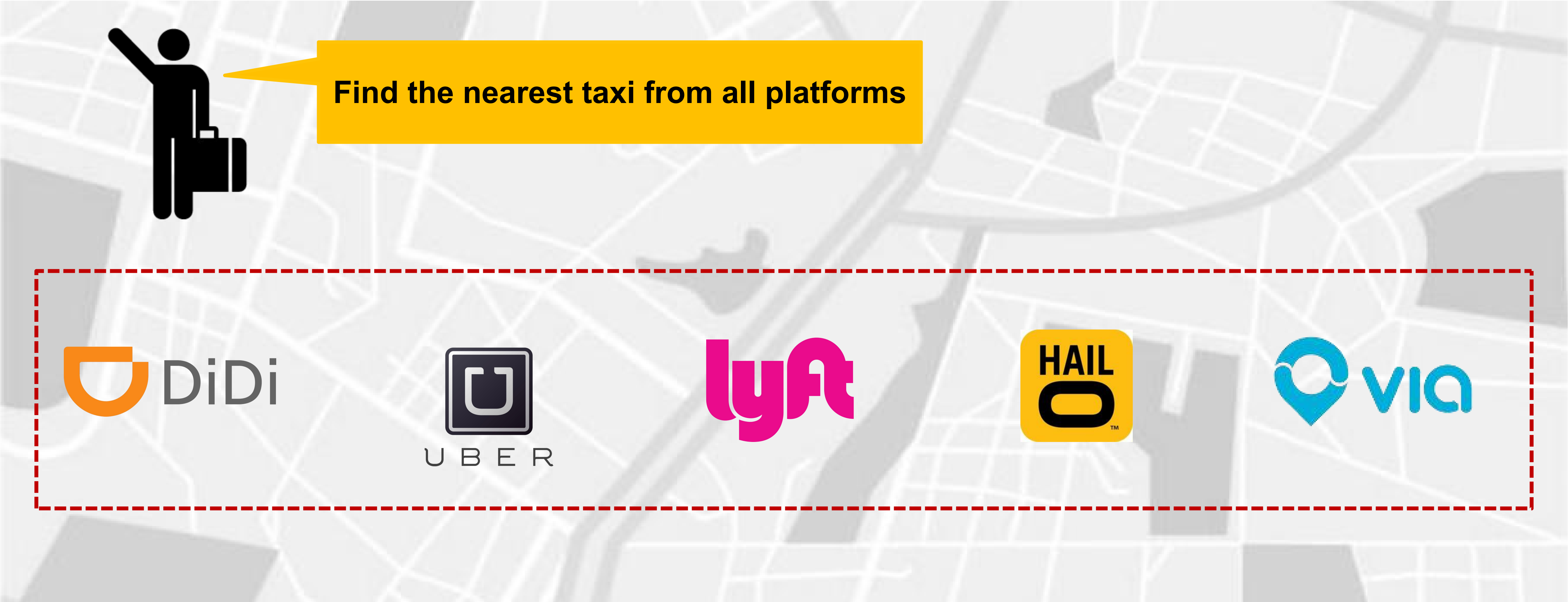
Scaling Spatial Queries to Data Federation

- Example of taxi calling
  - Traditionally: isolated taxi calling platforms

UBER
via
DiDi
lyft
HAIL O

Scaling Spatial Queries to Data Federation

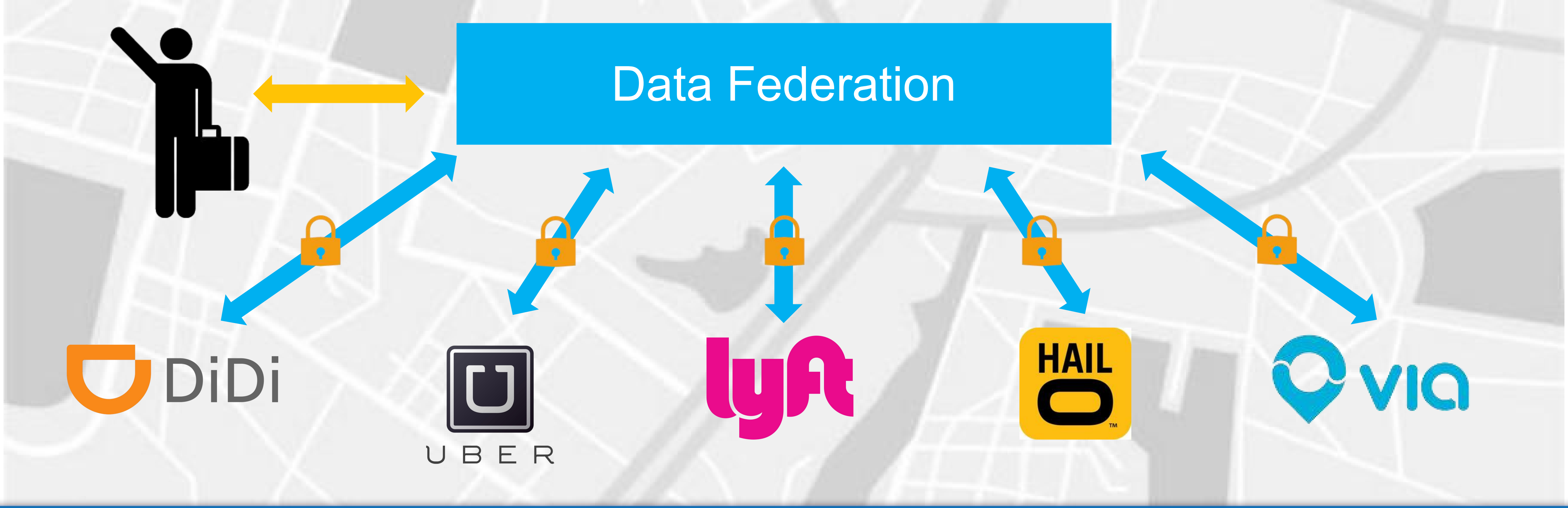
- Example of taxi calling
  - Traditionally: isolated taxi calling platforms
  - Emerging trend: unite multiple taxi calling platforms as a data federation



Find the nearest taxi from all platforms

Scaling Spatial Queries to Data Federation

- Generic settings
  - Multiple mutually distrusted silos
  - Demand security to protect each silo’s data



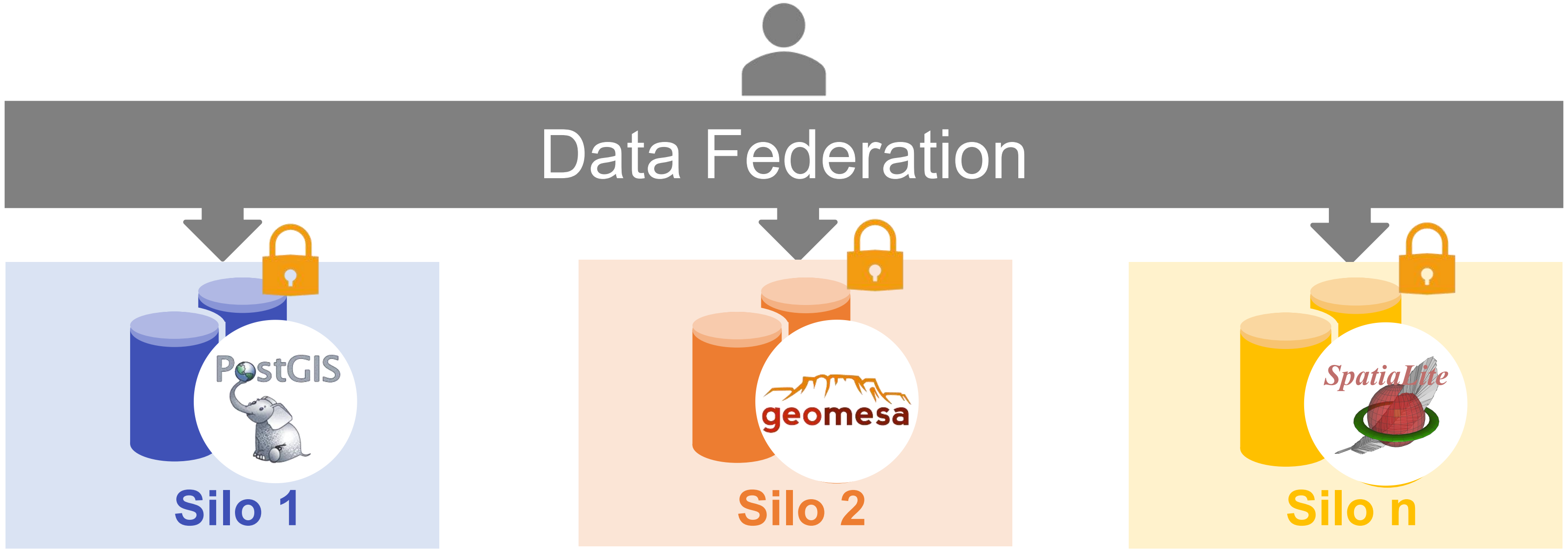
Secure spatial queries over large-scale data federation are non-trivial

Outline

- Background
- Problem Statement
- System Design
- Evaluations
- Conclusion

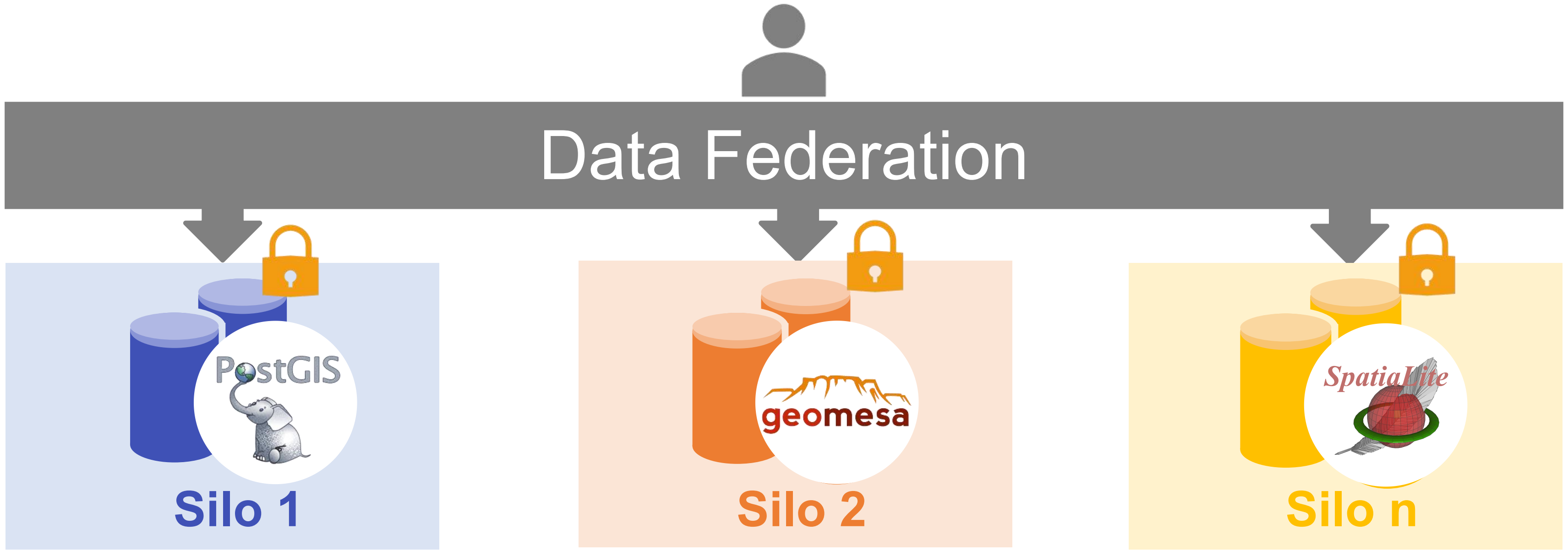
Problem Scope: Federated Spatial Queries

- Assumption & Requirement 1:
  - Autonomous databases at individual silos (≥ 2 silos)
    - Heterogenous databases managed independently by each silo
    - Minimal modifications to databases of each silo



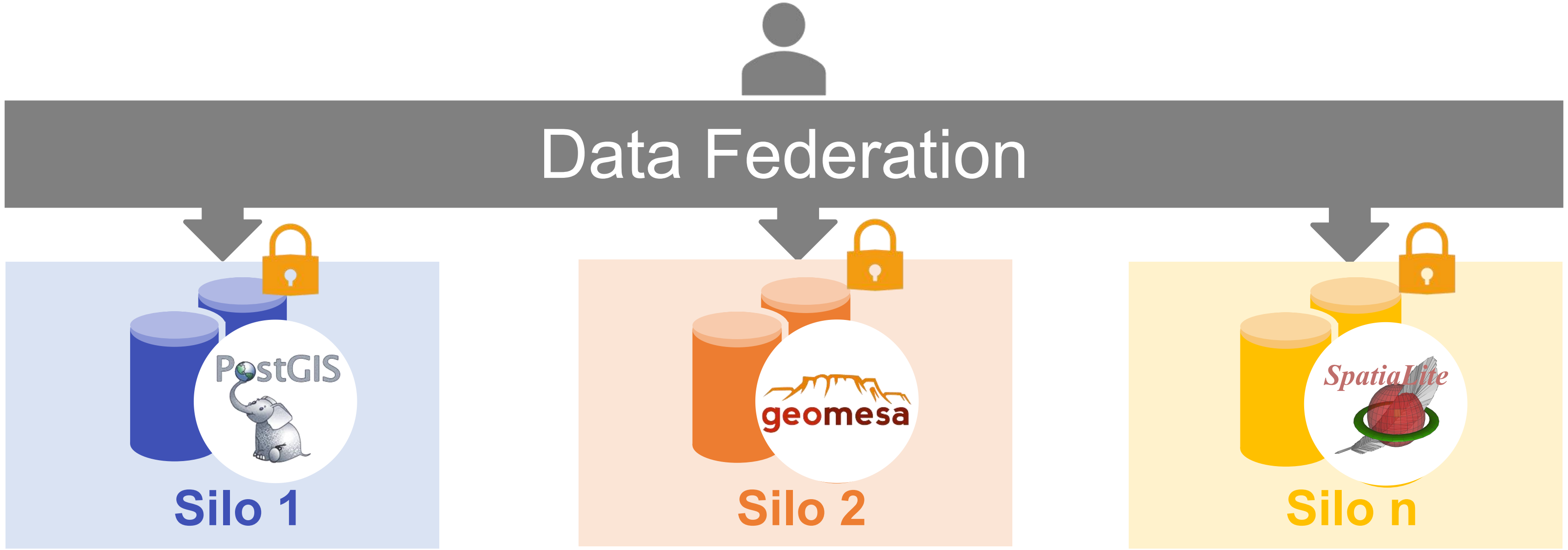
Problem Scope: Federated Spatial Queries

- Assumption & Requirement 2:
  - Secure queries against semi-honest adversary model
    - Silos may attempt to infer other silos’ data but execute queries honestly



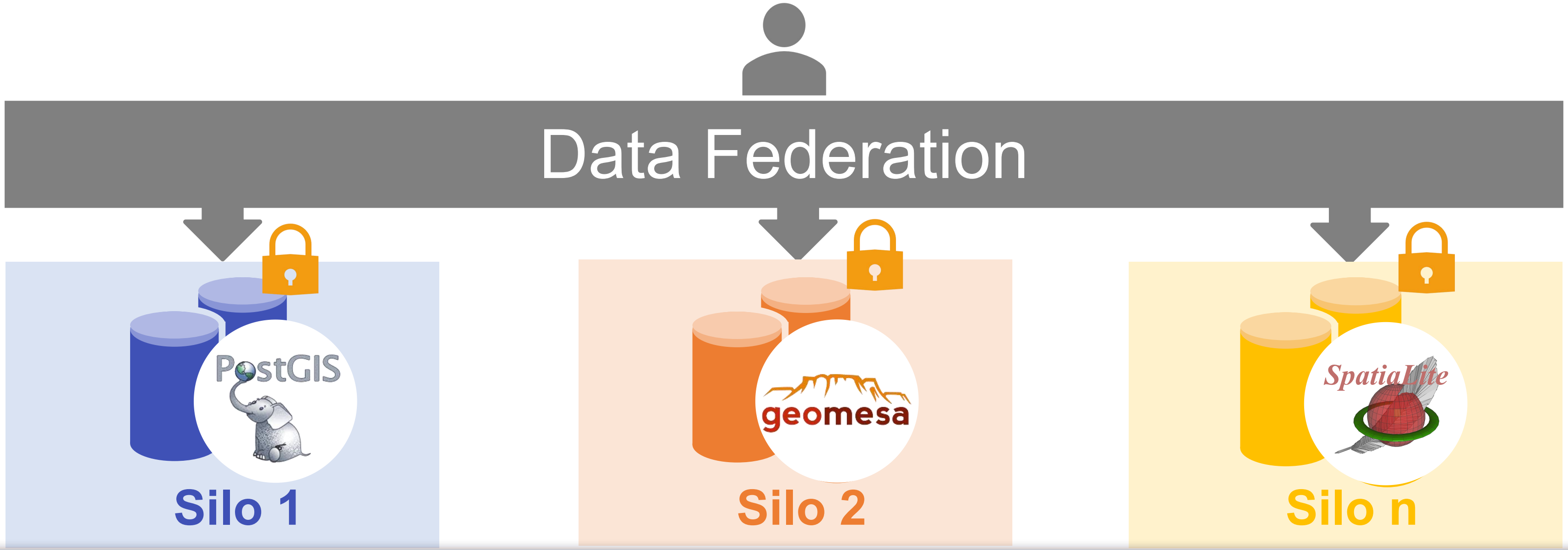
Problem Scope: Federated Spatial Queries

- Assumption & Requirement 3:
  - Efficient execution of mainstream spatial queries
    - Range query, range counting, kNN query, distance join, kNN join, ...



Problem Scope: Federated Spatial Queries

- In a nutshell
  - Autonomous databases at individual silos (≥ 2 silos)
  - Secure queries against semi-honest adversary model
  - Efficient execution of mainstream spatial queries



Existing solutions fail to fulfill all these requirements

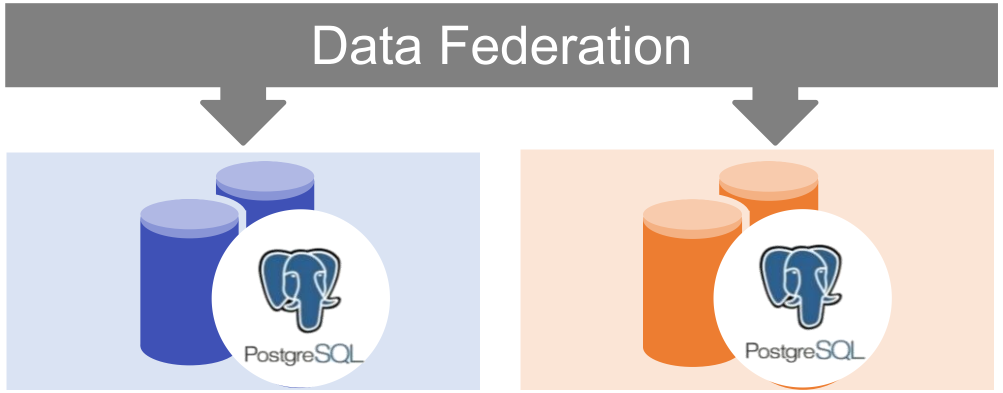
Limitations of Existing Solutions

- Requirements
  - Autonomous databases at individual silos (≥ 2 silos) ✗
  - Secure queries against semi-honest adversary model ✓
  - Efficient execution of mainstream spatial queries ✗
- STOA data federation systems
  - SMCQL [1] & Conclave [2]
  - Limited usability
  - Inefficient for spatial queries

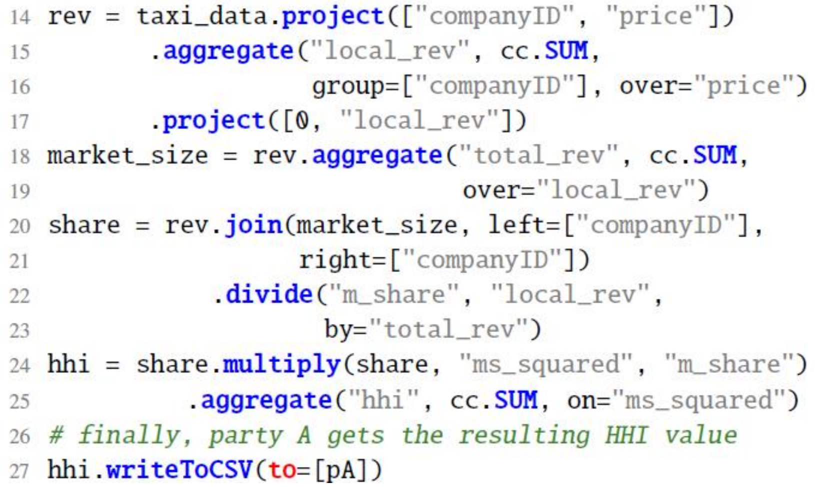
[1] Johes Bater, Gregory Elliott, Craig Eggen, et al. SMCQL: Secure Query Processing for Private Data Networks. PVLDB 2017.
[2] Nikolaj Volgushev, Malte Schwarzkopf, Ben Getchell, et al. Conclave: secure multi-party computation on big data. EuroSys 2019.

Existing Solutions: Limited Useability

- Cannot adapt to heterogenous databases
  - SMCQL [1] only supports PostgreSQL
- Unfriendly user interface
  - Conclave [2] does not support queries in SQL



SMCQL



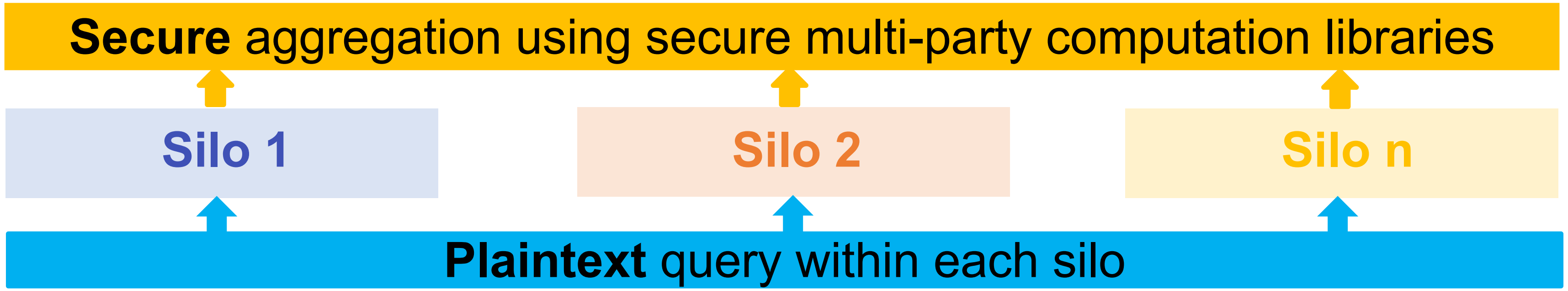
```
rev = taxi_data.project(["companyID", "price"])
       .aggregate("local_rev", cc.SUM,
                  group=["companyID"], over="price")
       .project([0, "local_rev"])
market_size = rev.aggregate("total_rev", cc.SUM,
                            over="local_rev")
share = rev.join(market_size, left=["companyID"],
                 right=["companyID"])
           .divide("m_share", "local_rev",
                   by="total_rev")
hhi = share.multiply(share, "ms_squared", "m_share")
          .aggregate("hhi", cc.SUM, on="ms_squared")
# finally, party A gets the resulting HHI value
hhi.writeToCSV(to=[pA])
```

Conclave

[1] Johes Bater, Gregory Elliott, Craig Eggen, et al. SMCQL: Secure Query Processing for Private Data Networks. PVLDB 2017.
[2] Nikolaj Volgushev, Malte Schwarzkopf, Ben Getchell, et al. Conclave: secure multi-party computation on big data. EuroSys 2019.

Existing Solutions: Inefficient for Spatial Queries

- Measurement study
  - Principles of SMCQL [1] & Conclave [2]



Secure aggregation using secure multi-party computation libraries

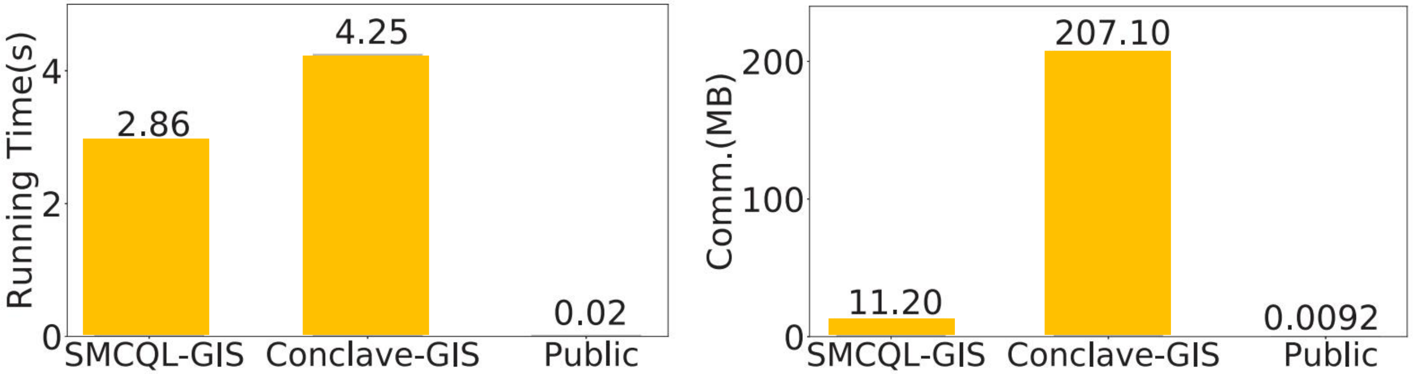
Silo 1

Silo 2

Silo n

Plaintext query within each silo

  - Extend to support spatial queries (SMCQL-GIS & Conclave-GIS)



(a) Running time

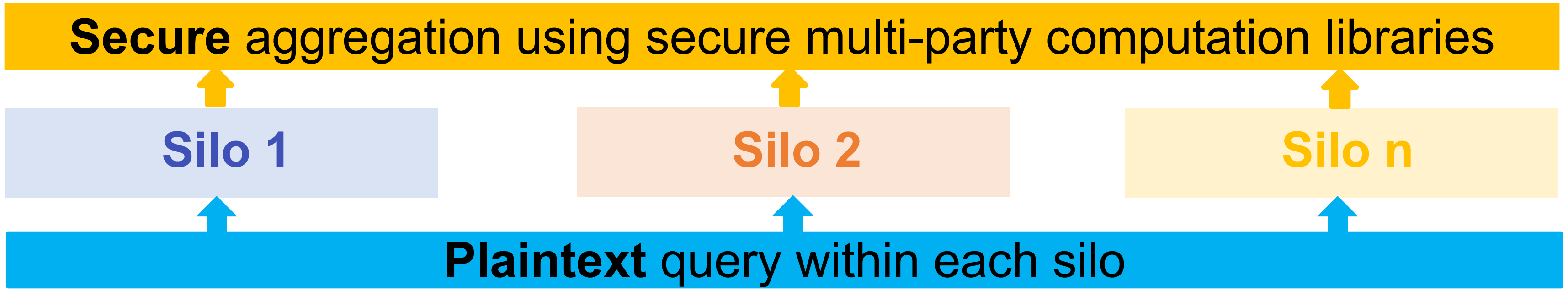
(b) Communication cost

[1] Johes Bater, Gregory Elliott, Craig Eggen, et al. SMCQL: Secure Query Processing for Private Data Networks. PVLDB 2017.

[2] Nikolaj Volgushev, Malte Schwarzkopf, Ben Getchell, et al. Conclave: secure multi-party computation on big data. EuroSys 2019.

Existing Solutions: Inefficient for Spatial Queries

- Measurement study
  - Principles of SMCQL [1] & Conclave [2]



Secure aggregation using secure multi-party computation libraries

Plaintext query within each silo

  - Efficiency bottleneck
    - Excessive secure operations
    - Reliance on general-purpose libraries

System	Plaintext	Secure
SMCQL-GIS with ObliVM [3]	0.14%	99.86%
Conclave-GIS with MP-SPDZ [4]	0.10%	99.90%

Percentage of time spent for plaintext or secure operations for a federated kNN query

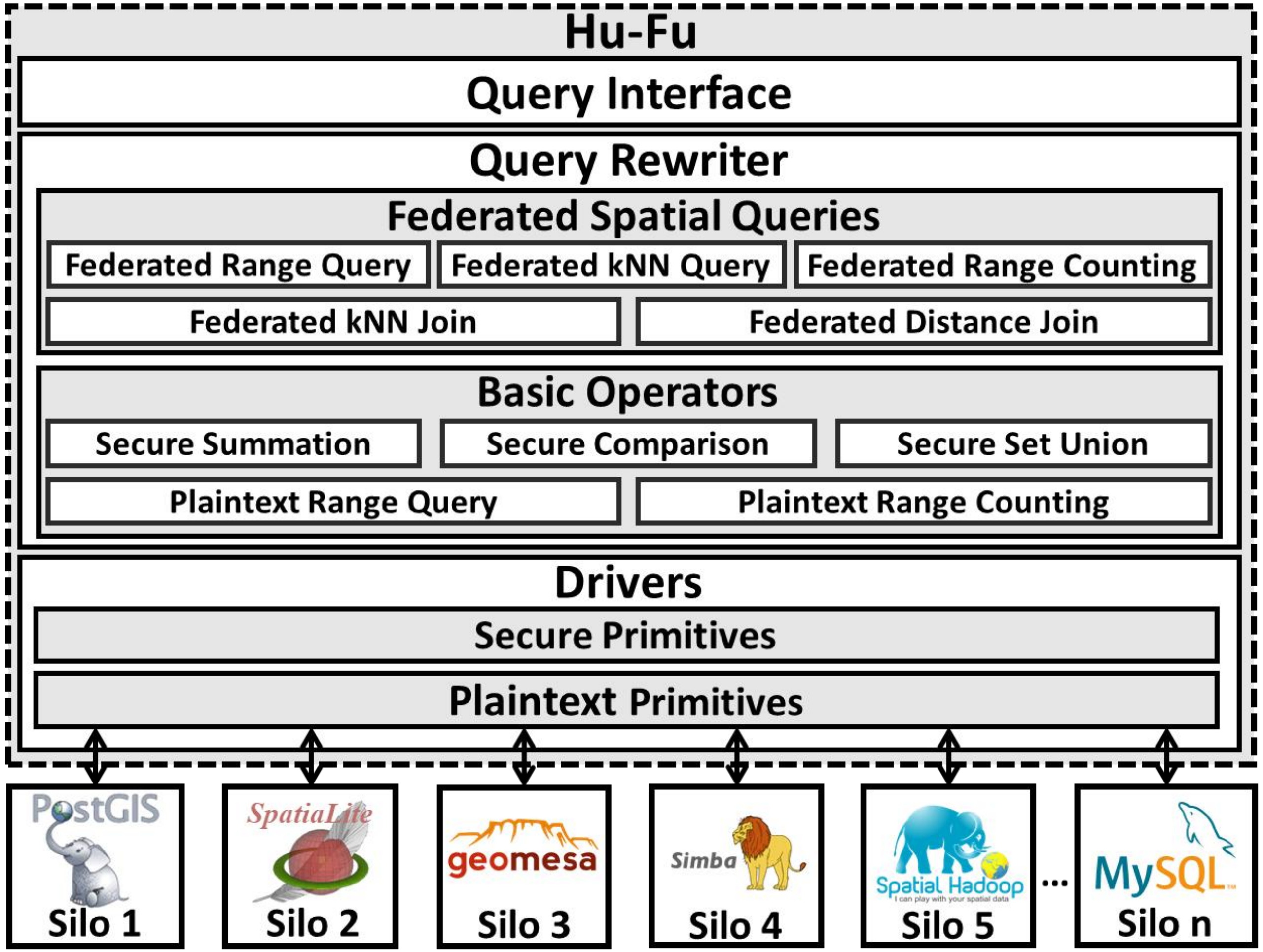
[3] Chang Liu, Xiao Shaun Wang, Kartik Nayak, et al. ObliVM: A Programming Framework for Secure Computation. S&P 2015.
[4] Marcel Keller. MP-SPDZ: A Versatile Framework for Multi-Party Computation. CCS 2020.

Outline

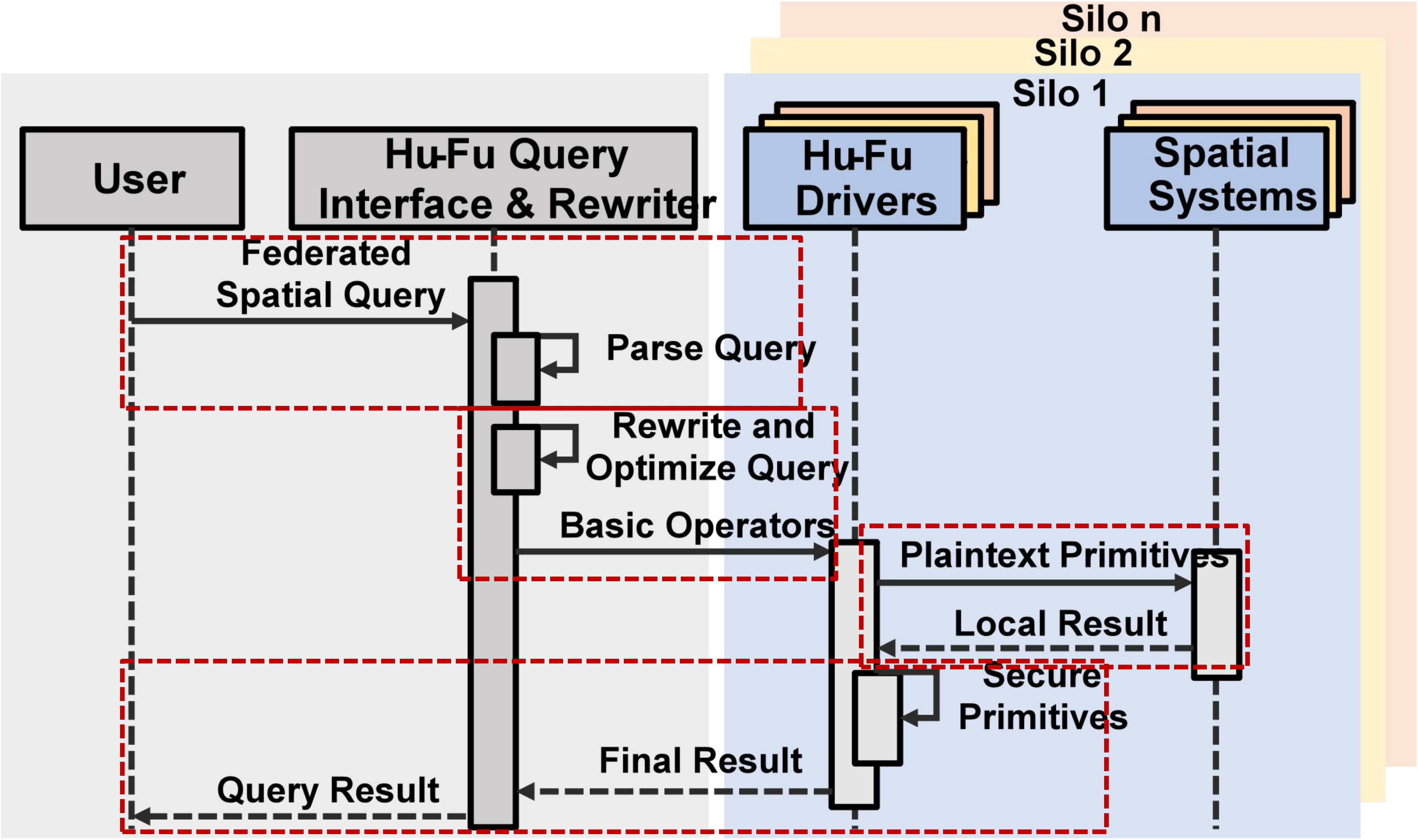
- Background
- Problem Statement
- System Design
- Evaluations
- Conclusion

Our Solution: Hu-Fu

- System for federated spatial queries
- Components
  - Query Rewriter
  - Drivers
  - Query Interface
- Features
  - Efficient & Secure
  - Extensible & User-friendly



Hu-Fu Workflow

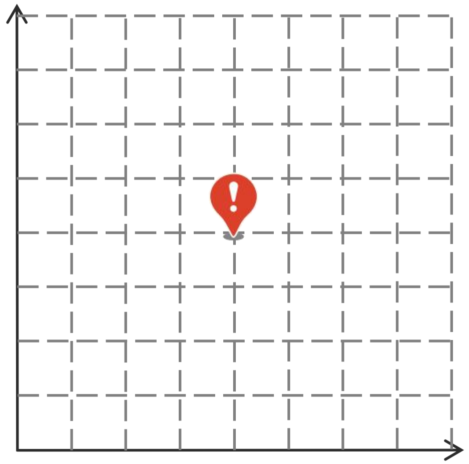


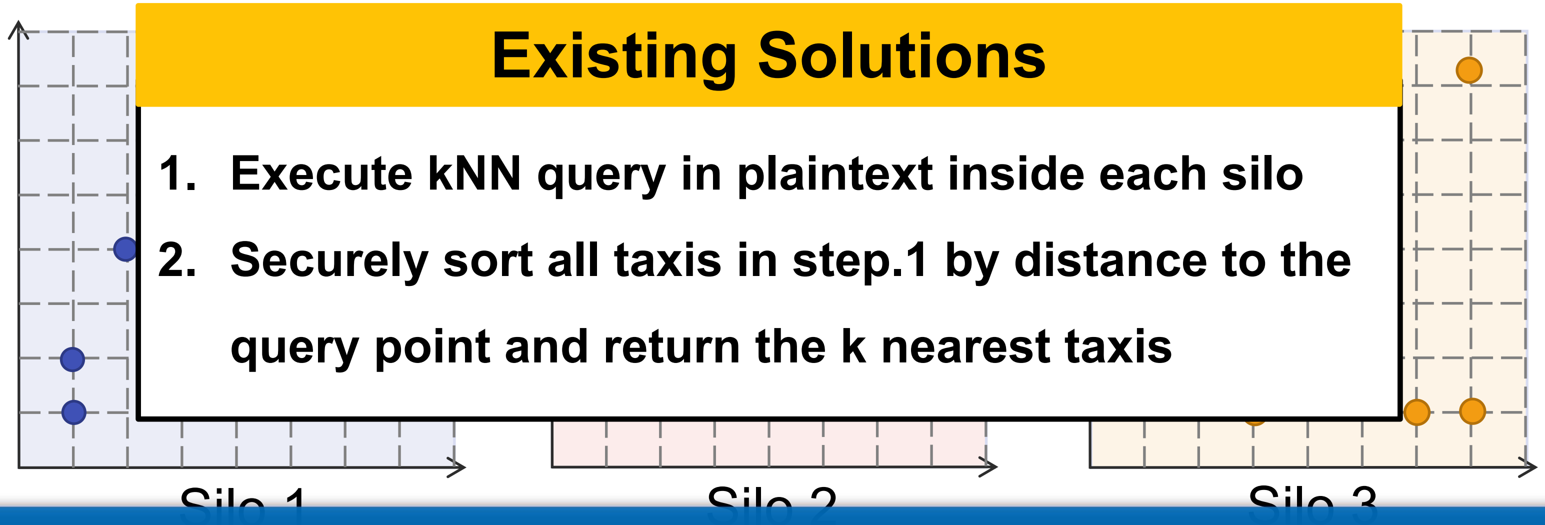
Hu-Fu Query Rewriter

- Functionalities
  - Decompose federated spatial queries into multiple basic operators
- Techniques
  - Define plaintext & secure operators necessary for mainstream federated spatial queries
    - Plaintext operators: range query, range counting
    - Secure operators: summation, comparison, set union
  - Novel query decomposition plans with many plaintext operators (within silos) and few secure operators (across silos) w/o compromising security

Hu-Fu Query Rewriter: Example of kNN Query

Find the k nearest taxis





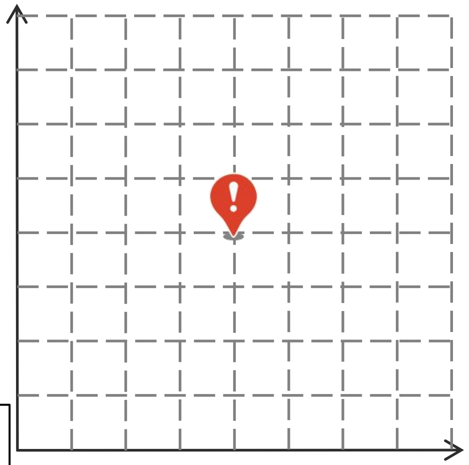
Existing Solutions

1. **Execute kNN query in plaintext inside each silo**
2. **Securely sort all taxis in step.1 by distance to the query point and return the k nearest taxis**

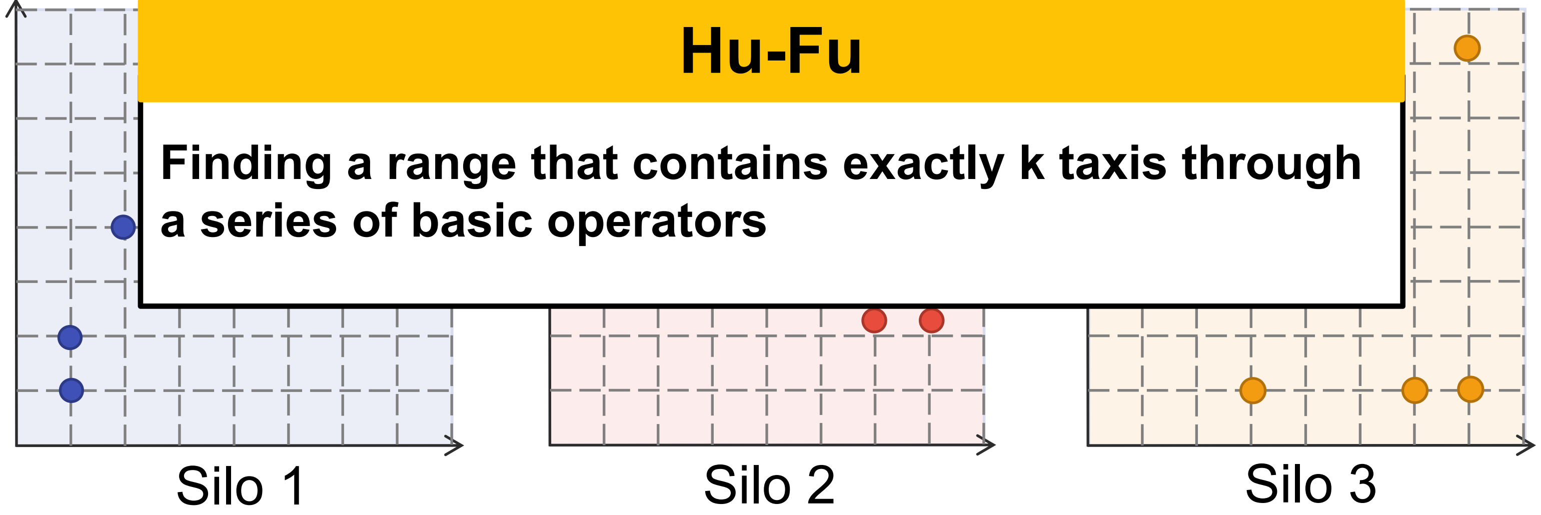
Inefficient : using $O(nklog(nk))$ secure distance comparisons

Hu-Fu Query Rewriter: Example of kNN Query

Find the 3 nearest taxis



SELECT taxi_id **FROM** taxi
WHERE KNN(POINT(x, y**),** taxi_location, 3**)**

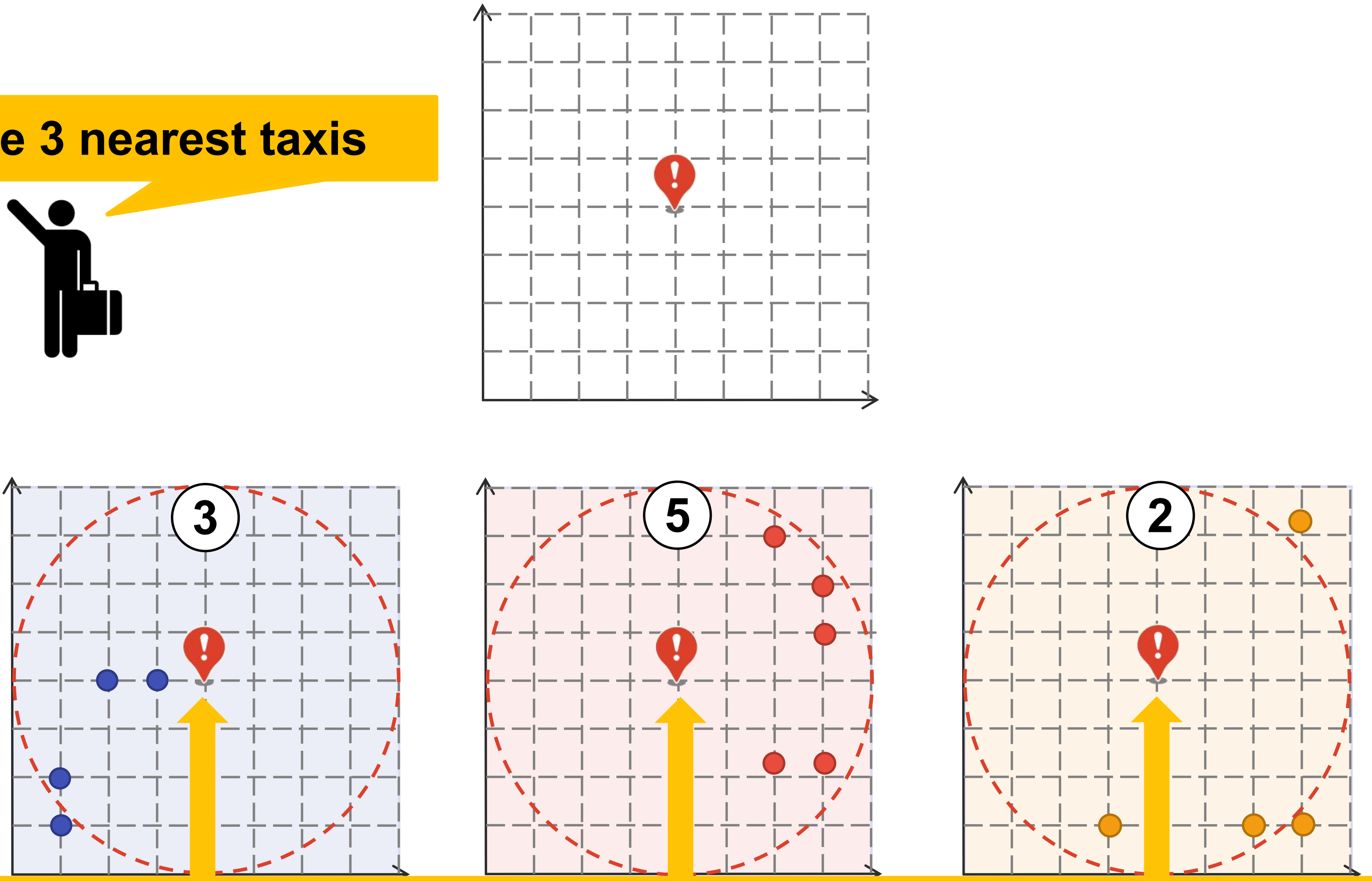


Hu-Fu

Finding a range that contains exactly k taxis through a series of basic operators

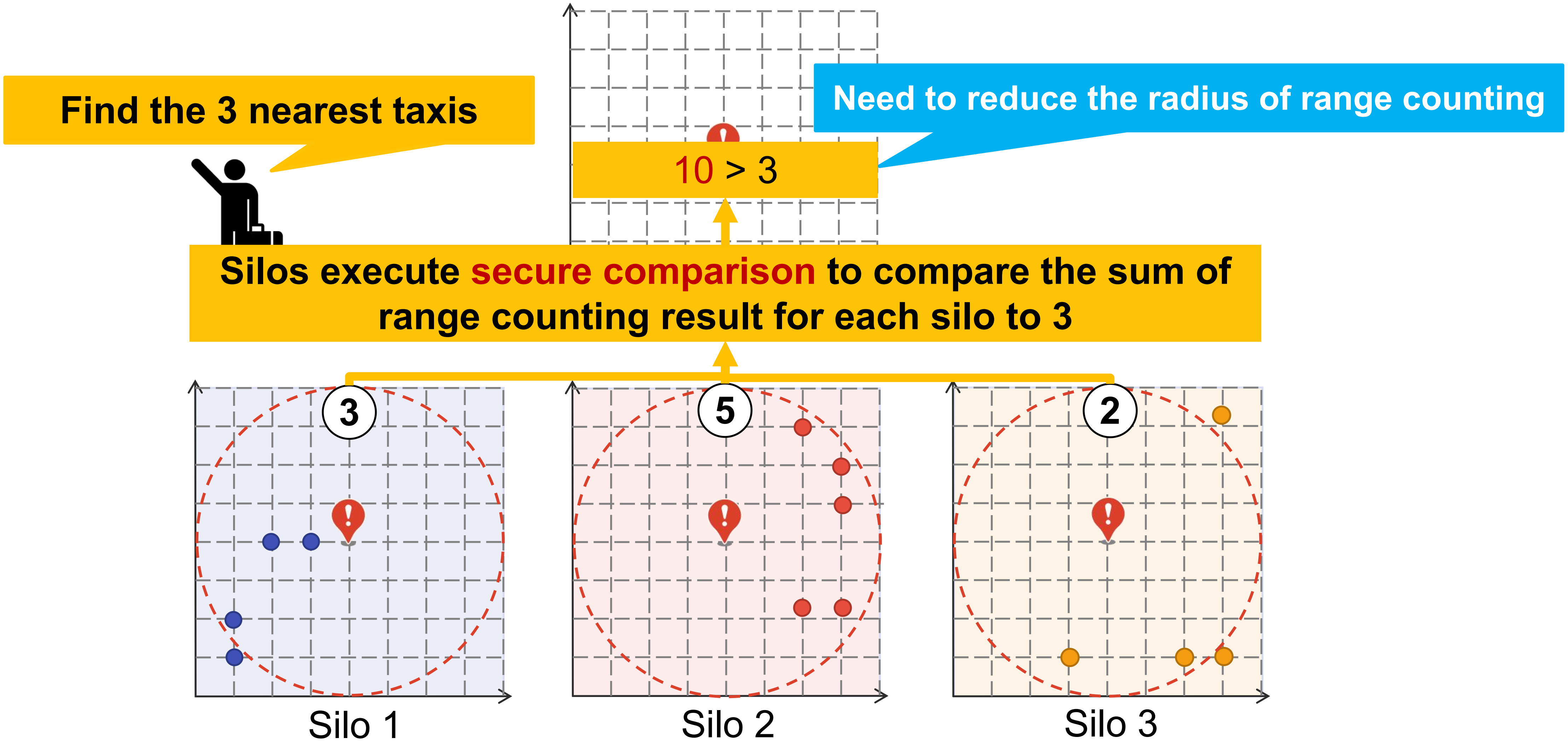
Hu-Fu Query Rewriter: Example of kNN Query

Find the 3 nearest taxis



Each silo executes plaintext range counting on local database

Hu-Fu Query Rewriter: Example of kNN Query



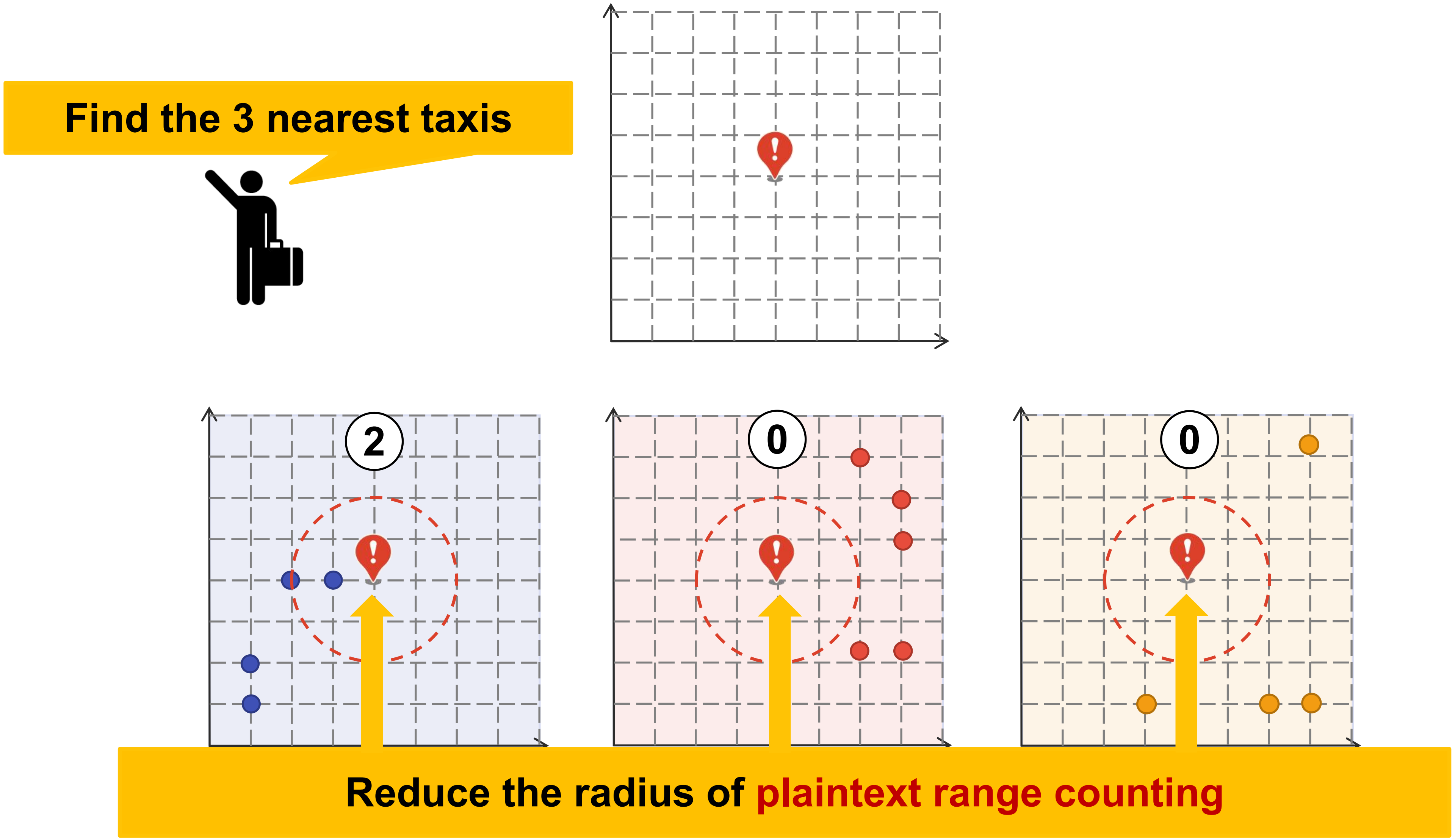
Find the 3 nearest taxis

Need to reduce the radius of range counting

10 > 3

Silos execute secure comparison to compare the sum of range counting result for each silo to 3

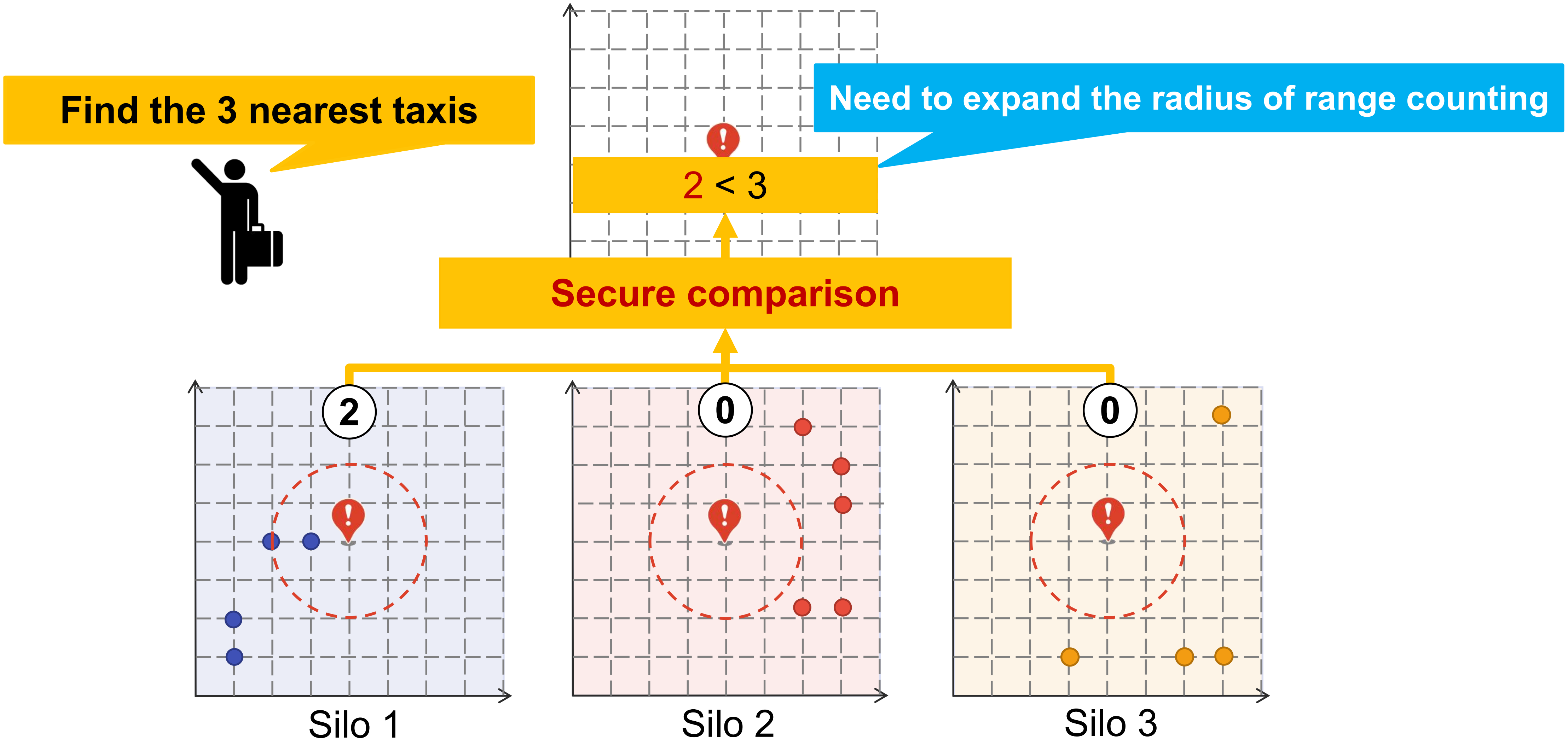
Hu-Fu Query Rewriter: Example of kNN Query



Find the 3 nearest taxis

Reduce the radius of plaintext range counting

Hu-Fu Query Rewriter: Example of kNN Query



Find the 3 nearest taxis

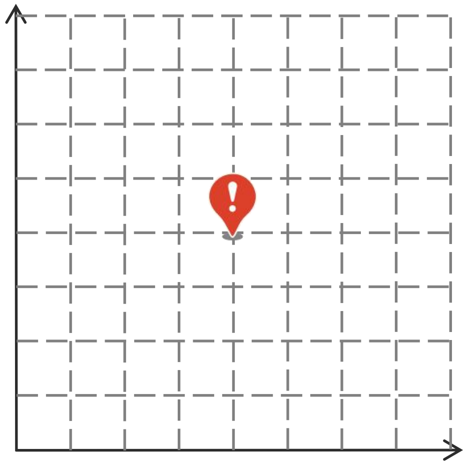
Need to expand the radius of range counting

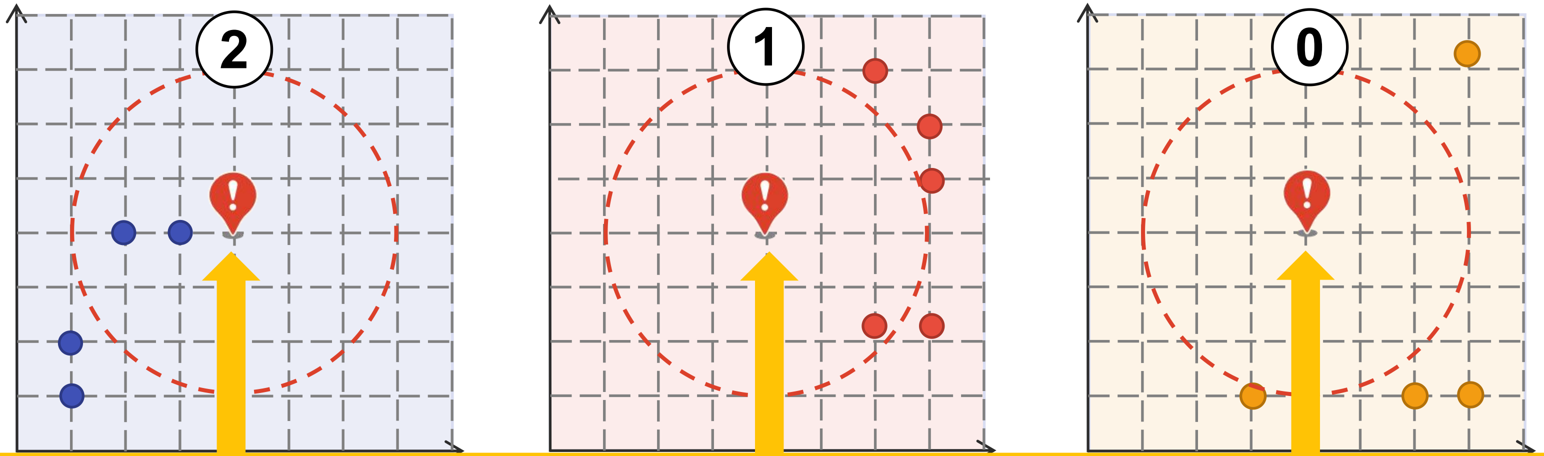
2 < 3

Secure comparison

Hu-Fu Query Rewriter: Example of kNN Query

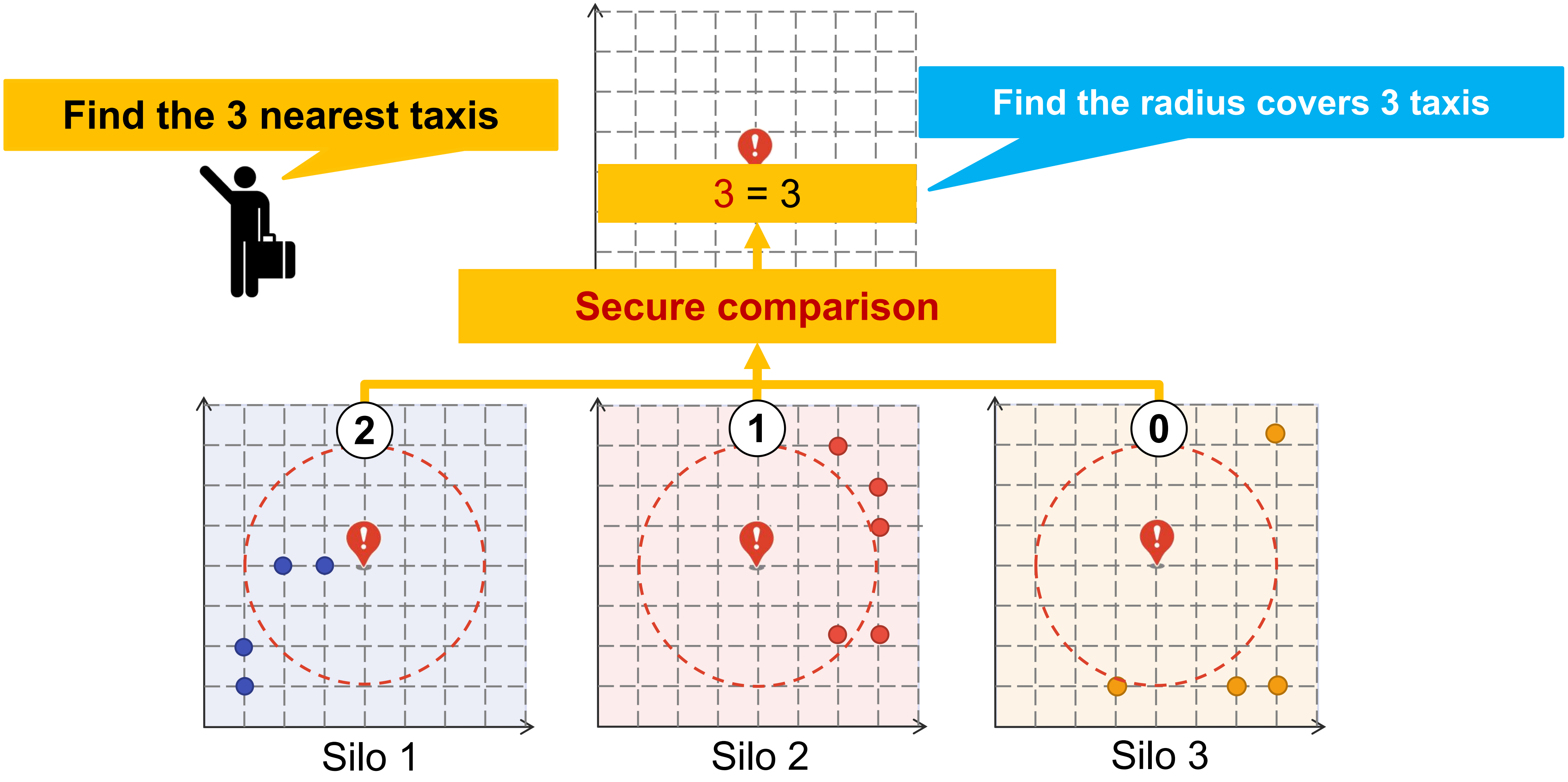
Find the 3 nearest taxis



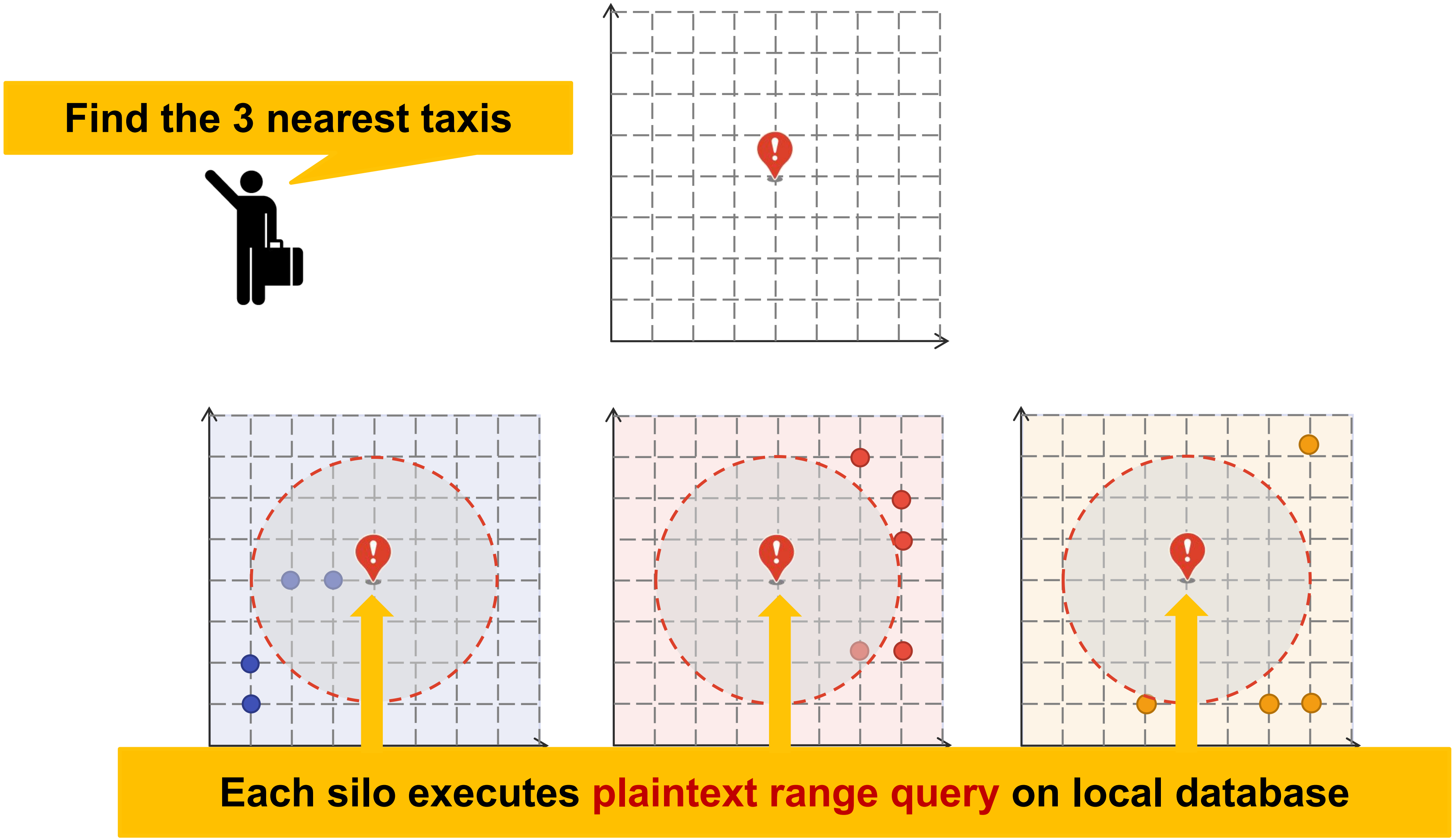


Expand the radius of plaintext range counting

Hu-Fu Query Rewriter: Example of kNN Query



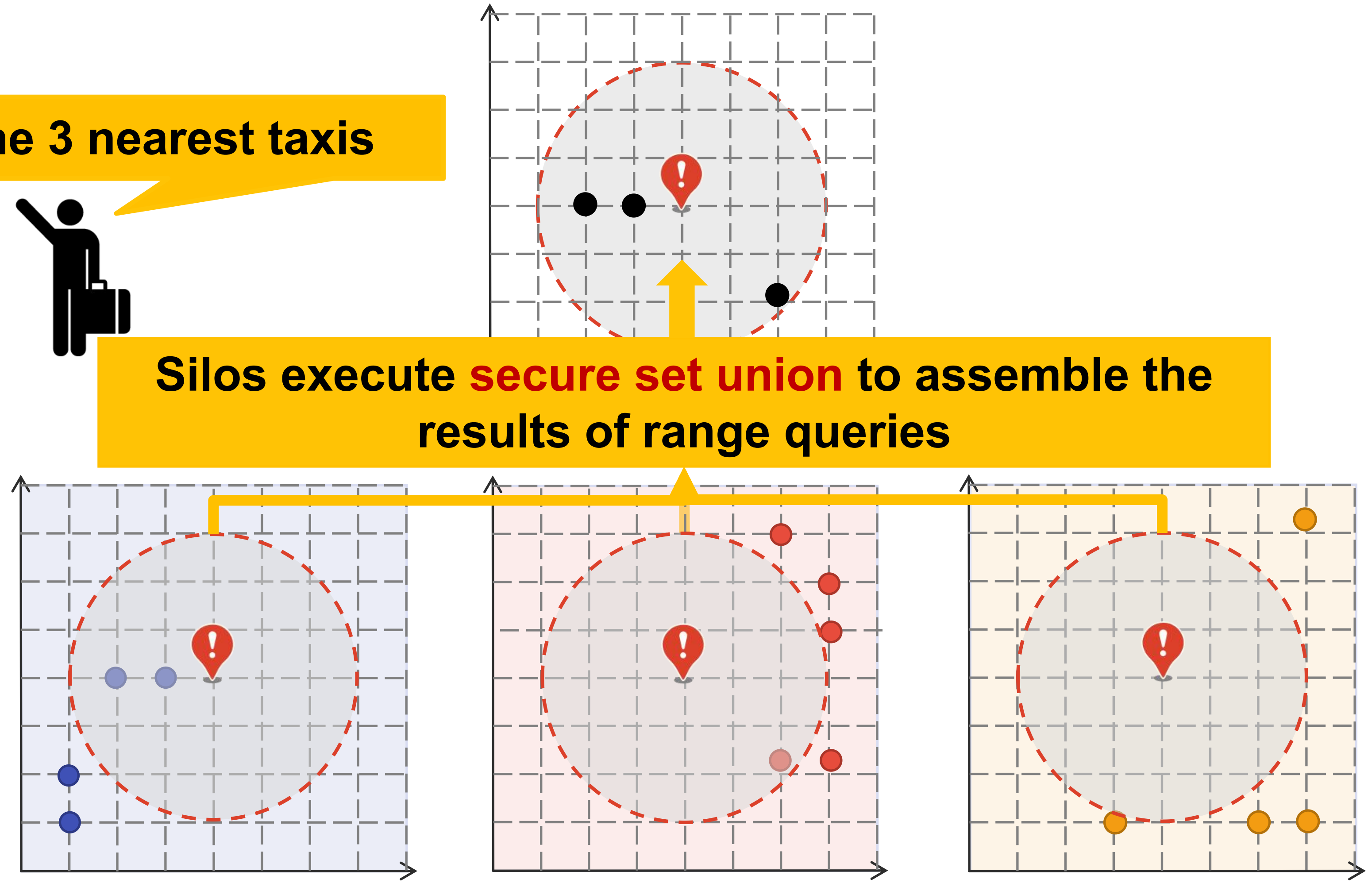
Hu-Fu Query Rewriter: Example of kNN Query



Find the 3 nearest taxis

Each silo executes plaintext range query on local database

Hu-Fu Query Rewriter: Example of kNN Query



Find the 3 nearest taxis

Silos execute secure set union to assemble the results of range queries

Efficient : using $O(log)$ secure comparisons and 1 secure set union

Hu-Fu Query Rewriter: Decomposition Plans

- Decomposition plans for mainstream spatial queries

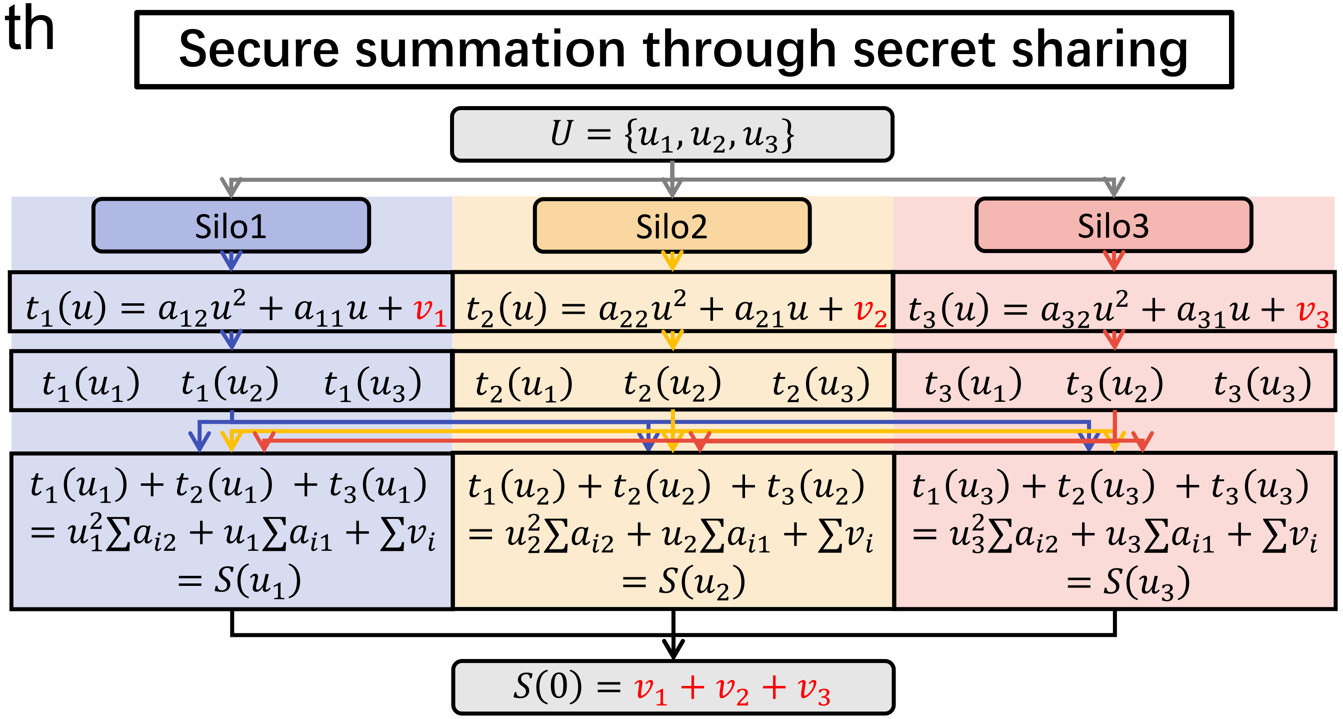
Federated Spatial Query	Number of Secure Operator			Number of Plaintext Operator	
	Comparison	Summation	Set Union	Range Query	Range Counting
Federated Range Query	0	0	1	n	0
Federated Range Counting	0	1	0	0	n
Federated Distance Join	0	0	1	$N\|R\|$	0
Federated kNN Query	$O(log\frac{v_0}{\epsilon_0})$	0	1	n	$O(n\ log\frac{v_0}{\epsilon_0})$
Federated kNN Join	$O(\|R\|log\frac{v_0}{\epsilon_0})$	0	1	$N\|R\|$	$O(\|R\|log\frac{v_0}{\epsilon_0})$

- Secure under semi-honest adversary (proof in full paper [5])
- Possible extensions to other queries (range type, aggregation)

[5] https://hufudb.com/static/paper/2022/PVLDB2022_Hu-Fu.pdf

Hu-Fu Drivers

- Functionalities
  - Execute basic operators sent by the query rewriter
- Techniques
  - Implement secure primitives with dedicated secure multi-party computation protocols



Secure summation through secret sharing

$U = \{u_1, u_2, u_3\}$

Silo1

Silo2

Silo3

$t_1(u) = a_{12}u^2 + a_{11}u + v_1$

$t_2(u) = a_{22}u^2 + a_{21}u + v_2$

$t_3(u) = a_{32}u^2 + a_{31}u + v_3$

$t_1(u_1) \quad t_1(u_2) \quad t_1(u_3)$

$t_2(u_1) \quad t_2(u_2) \quad t_2(u_3)$

$t_3(u_1) \quad t_3(u_2) \quad t_3(u_3)$

$t_1(u_1) + t_2(u_1) + t_3(u_1) = u_1^2\sum a_{i2} + u_1\sum a_{i1} + \sum v_i = S(u_1)$

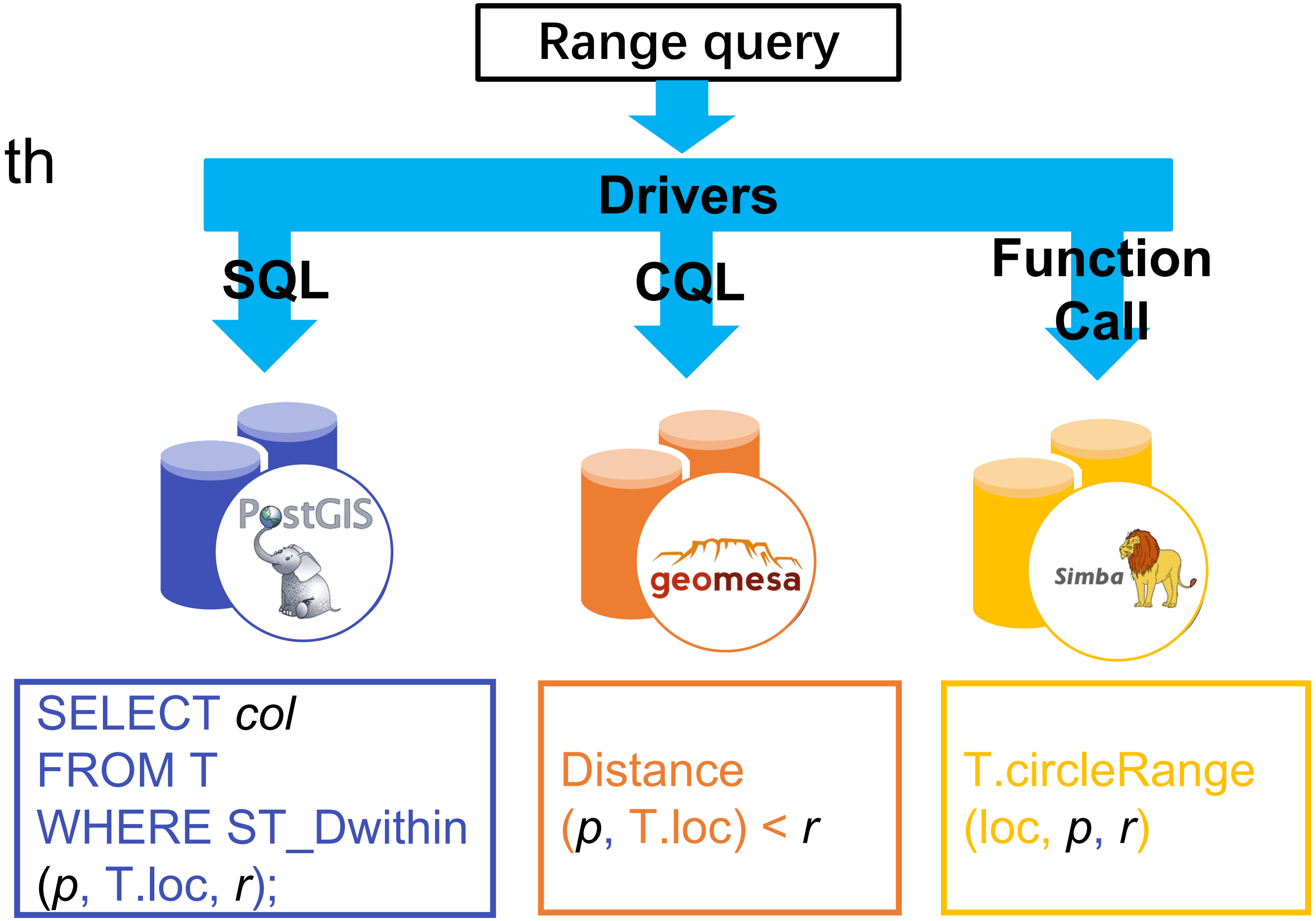
$t_1(u_2) + t_2(u_2) + t_3(u_2) = u_2^2\sum a_{i2} + u_2\sum a_{i1} + \sum v_i = S(u_2)$

$t_1(u_3) + t_2(u_3) + t_3(u_3) = u_3^2\sum a_{i2} + u_3\sum a_{i1} + \sum v_i = S(u_3)$

$S(0) = v_1 + v_2 + v_3$

Hu-Fu Drivers

- Functionalities
  - Execute basic operators sent by the query rewriter
- Techniques
  - Implement secure primitives with dedicated secure multi-party computation protocols
  - Implement plaintext primitives leveraging spatial database



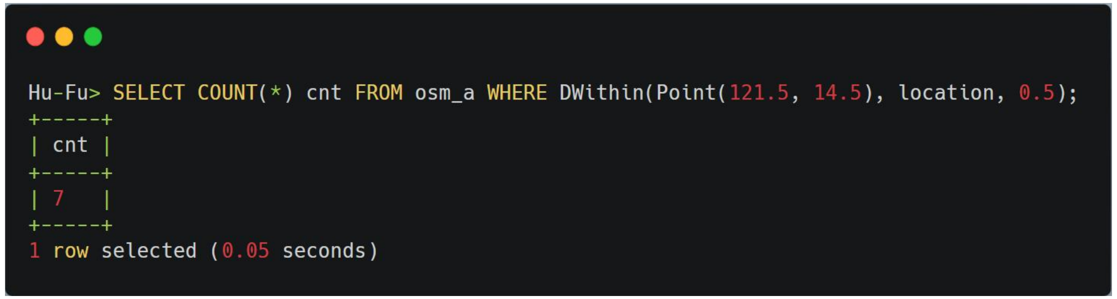
SELECT *col*
FROM T
WHERE ST_Dwithin
(*p*, T.loc, *r*);

Distance
(*p*, T.loc) < *r*

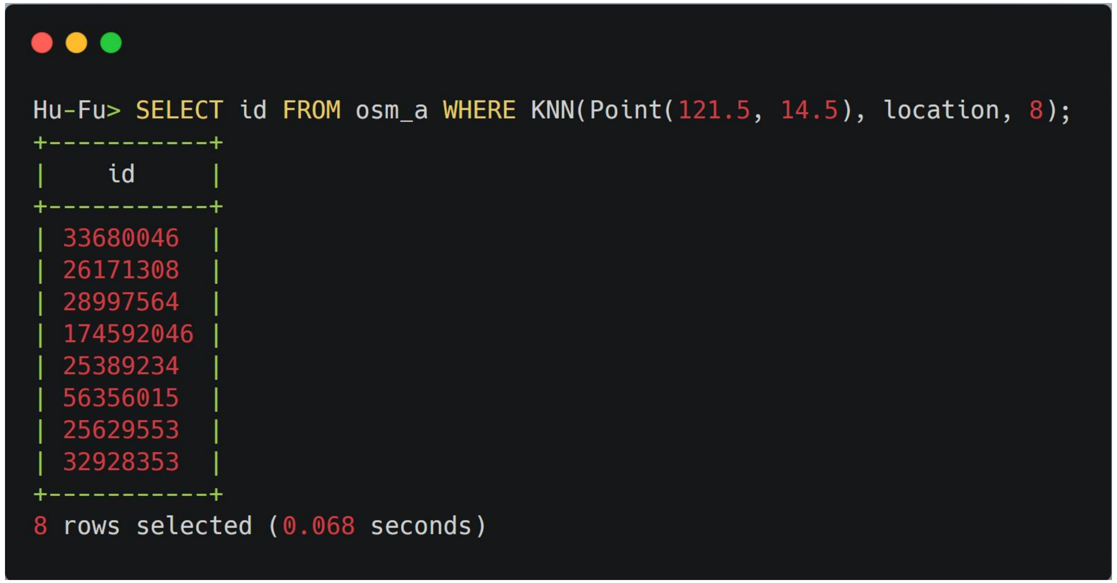
T.circleRange
(loc, *p*, *r*)

Hu-Fu Query Interface

- Functionalities
  - Provide federated view to users
  - Support federated spatial queries written in SQL
- Techniques
  - Extend the schema manager and parser of Apache Calcite



```
Hu-Fu> SELECT COUNT(*) cnt FROM osm_a WHERE DWithin(Point(121.5, 14.5), location, 0.5);
+-----+
| cnt |
+-----+
| 7   |
+-----+
1 row selected (0.05 seconds)
```



```
Hu-Fu> SELECT id FROM osm_a WHERE KNN(Point(121.5, 14.5), location, 8);
+-----------+
|    id     |
+-----------+
| 33680046  |
| 26171308  |
| 28997564  |
| 174592046 |
| 25389234  |
| 56356015  |
| 25629553  |
| 32928353  |
+-----------+
8 rows selected (0.068 seconds)
```

Edmon Begoli, Jesús Camacho-Rodríguez, Julian Hyde, et al. Apache Calcite: A Foundational Framework for Optimized Query Processing Over Heterogeneous Data Sources. SIGMOD 2018.

Outline

- Background
- Problem Statement
- Hu-Fu Overview
- Evaluations
- Conclusion

Experimental Setups

- Dataset
  - Multi-company Spatial Data in Beijing
  - OpenStreetMap (OSM)
- Parameter settings
  - # of silos: 2 ~ 10
  - # of spatial objects: 10^4 ~ 10^9
  - Spatial database system types:
    - PostGIS, MySQL, SpatiaLite, GeoMesa, Simba [1], SpatialHadoop [2]
- Metrics
  - Running time & Communication cost

[1] Dong Xie, Feifei Li, Bin Yao, et al. Simba: Efficient In-Memory Spatial Analytics. SIGMOD 2016.
[2] Ahmed Eldawy, Mohamed F. Mokbel. SpatialHadoop: A MapReduce framework for spatial data. ICDE 2015.

Experimental Setups

- Baseline
  - SMCQL-GIS:
    - Extend SMCQL [1] to support spatial queries
    - Aggregate silos' result with ObliVM [3] (only supports 2 silos)
  - Conclave-GIS:
    - Extend Conclave [2] to support spatial queries
    - Aggregate silos' result with MP-SPDZ [4] (supports ≥ 2 silos)
  - Public:
    - Aggregate silos' result in plaintext
  - ☐ All baselines use PostGIS for each silo

[1] Johes Bater, Gregory Elliott, Craig Eggen, et al. SMCQL: Secure Query Processing for Private Data Networks. PVLDB 2017.

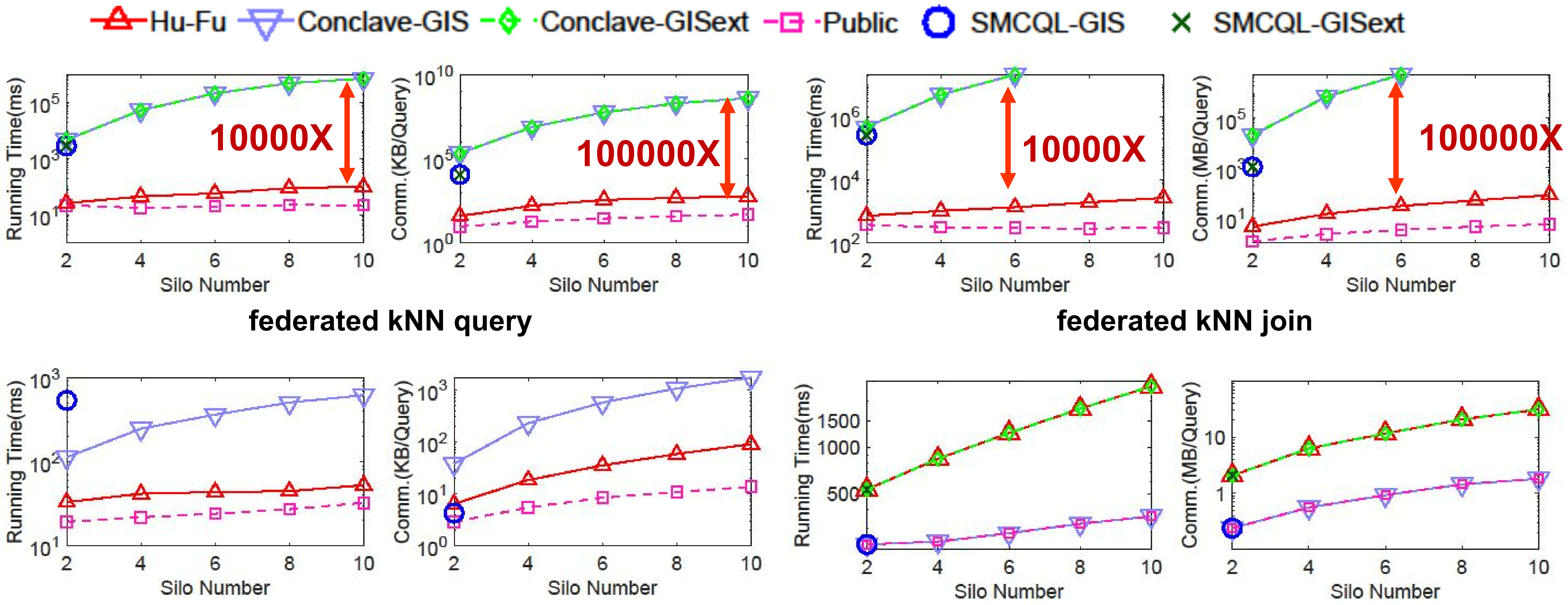
[2] Nikolaj Volgushev, Malte Schwarzkopf, Ben Getchell, et al. Conclave: secure multi-party computation on big data. EuroSys 2019.

[3] Chang Liu, Xiao Shaun Wang, Kartik Nayak, et al. ObliVM: A Programming Framework for Secure Computation. S&P 2015.

[4] Marcel Keller. MP-SPDZ: A Versatile Framework for Multi-Party Computation. CCS 2020.

Main Results

- Running time and communication cost of federated spatial queries
  - Up to 4 orders of magnitude faster
  - Up to 5 orders of magnitude lower communication cost



federated kNN query

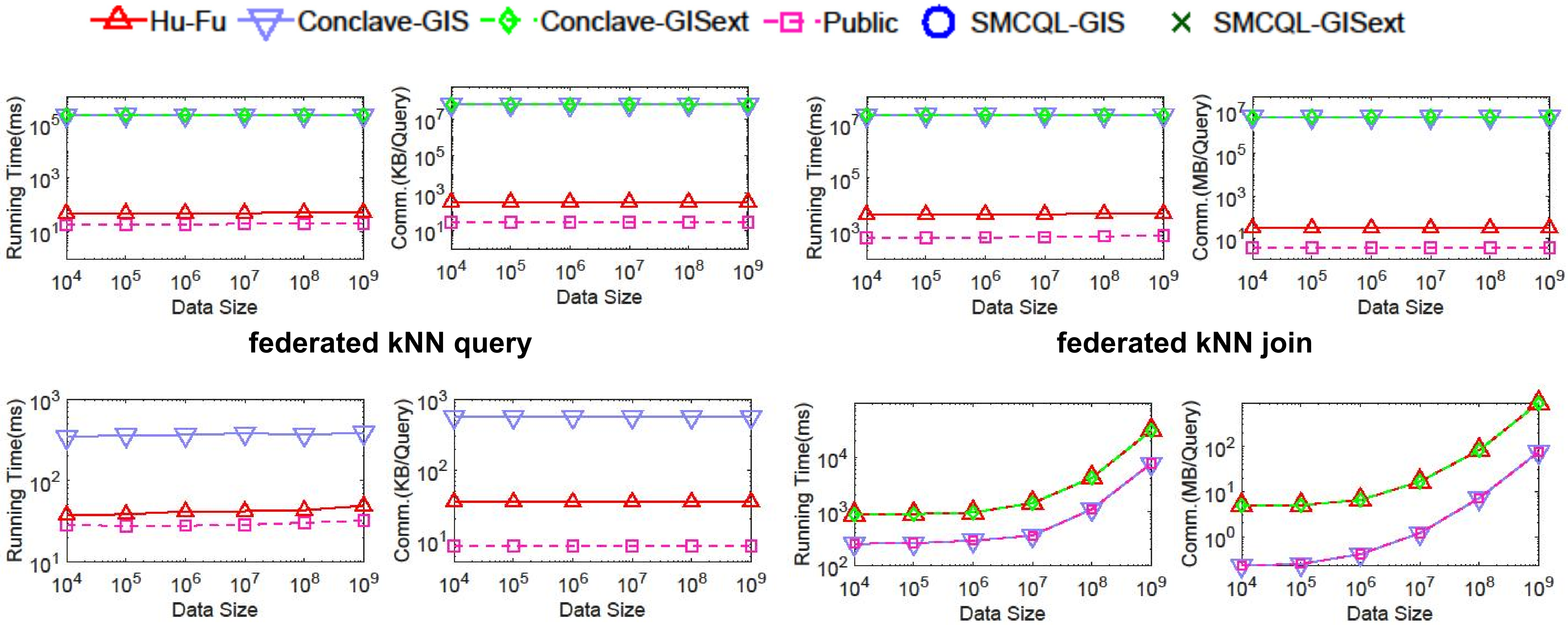
federated kNN join

federated range counting

federated distance join

Main Results

- Scalability
  - Hu-Fu scales well with data size for federated spatial queries



federated kNN query

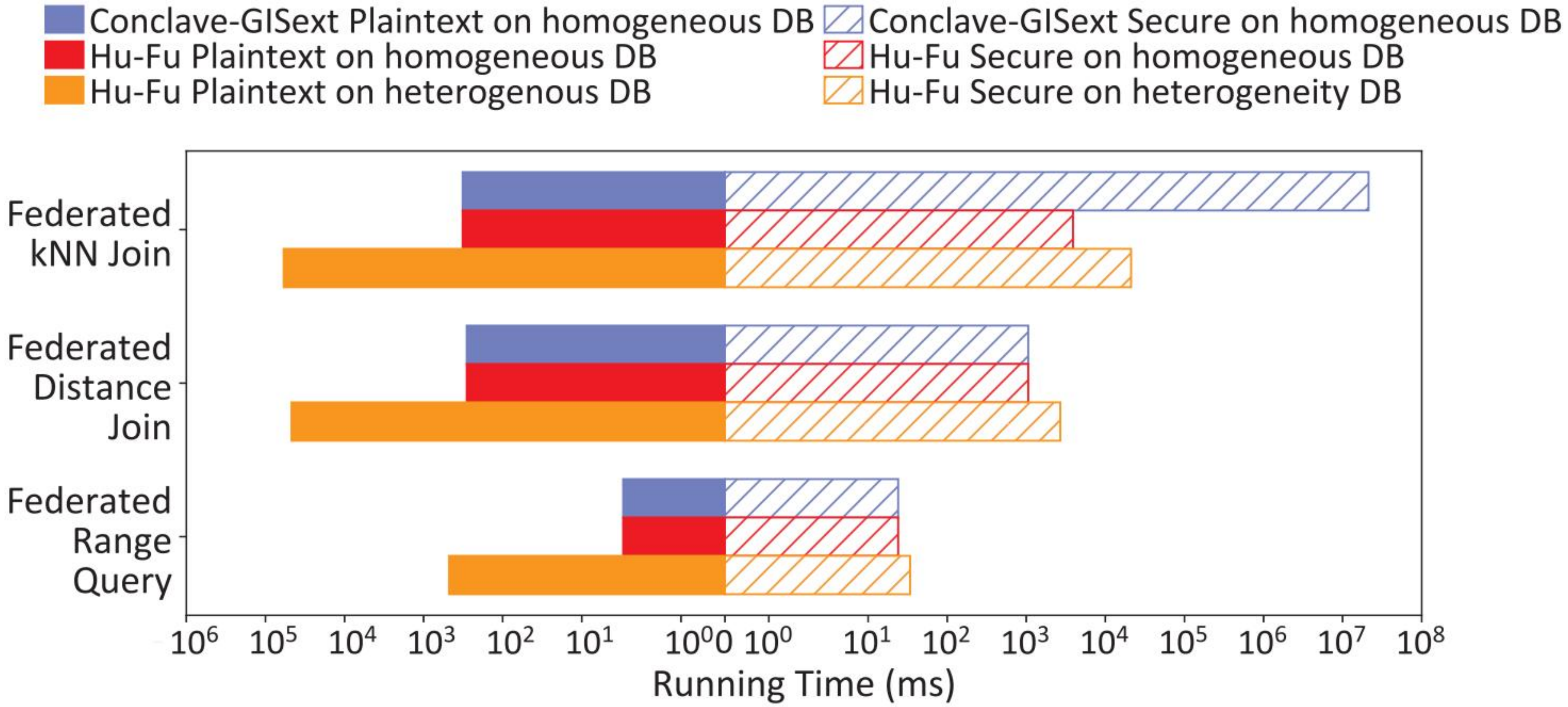
federated kNN join

federated range counting

federated distance join

Main Results

- Adapt to heterogeneous spatial databases
  - Hu-Fu functions with silos running heterogeneous databases
  - Performance can be limited by the slowest database in the federation



All silos use PostGIS ⇔ Each silo uses a different database system

Outline

- Background
- Problem Statement
- Hu-Fu Overview
- Evaluations
- Conclusion

Conclusion

- Hu-Fu is the first spatial data federation system
  - Design novel decomposition plans for secure and efficient federated spatial queries
  - Support SQL queries across multiple heterogeneous spatial databases
- Extensive experiments validate the efficiency of Hu-Fu

Hu Fu

https://github.com/BUAA-BDA/OpenHuFu

Thank You