Query Guard: Privacy-preserving Latency-aware Query Optimization for Edge Computing

Runhua Xu, Balaji Palanisamy and James Joshi
University of Pittsburgh
Pittsburgh, PA, US
runhua.xu@pitt.edu
Edge Computing

It allows data produced by IoT devices to be processed geographically closer to where it is created instead of sending it across long routes to data centers/clouds.

(The background image is from xtelesis.com)

The Laboratory for Education and Research on Security Assured Information Systems (LERSAIS)

Wednesday, August 1, 2018
Why/How Edge Computing

- Why does edge computing matter
  - IoT devices have poor connectivity
  - It’s not efficient for IoT devices to be constantly connected to a central cloud.
  - latency-sensitive processing requirement
- How edge computing works
  - Triage the data locally
Overview of Query Processing

A 3-step Process

It transforms a high-level query into an equivalent and more efficient lower-level query.
Query Processing Example

**Query in high-level language**

```
SELECT attr₁, attr₂, attr₃
FROM t₁, t₂, t₃
WHERE t₁.a₁ = t₂.a₁ AND t₁.a₂ = t₃.a₂
```

**SQL expression**

```
SELECT attr₁, attr₂, attr₃
FROM t₁, t₂, t₃
WHERE t₁.a₁ = t₂.a₁ AND t₁.a₂ = t₃.a₂
```

**Algebra expression**

```
Π_{attr₁,attr₂,attr₃}(t₂ ⋈_{a₁} t₁ ⋈_{a₂} t₃)
```

**Query execution plans**

1. `Π_{attr₁,attr₂,attr₃}(
   t₂ ⋈_{a₂} t₃
   ⋈_{a₁} t₁
)

2. `Π_{attr₁,attr₂,attr₃}(
   t₂ ⋈_{a₂} t₂
   ⋈_{a₁} t₁
)

Result of the Query
Distributed Query Processing Example

**Query in high-level language**

```
SELECT attr1, attr2, attr3
FROM t1, t2, t3
WHERE t1.a1 = t2.a1 AND t1.a2 = t3.a2
```

**SQL expression**

```
SELECT attr1, attr2, attr3
FROM t1, t2, t3
WHERE t1.a1 = t2.a1 AND t1.a2 = t3.a2
```

**Algebra expression**

```
\Pi_{\text{attr1}, \text{attr2}, \text{attr3}} (t_2 \bowtie_{s_1} t_1 \bowtie_{s_2} t_3)
```

**Query Optimizer**

**Query Evaluation Engine**

**Result of the Query**

**Query execution plans**

- **Global Optimization**
- **Local Optimization**

---

Wednesday, August 1, 2018

The Laboratory for Education and Research on Security Assured Information Systems (LERSAIS)
Query Processing in Edge Computing

Join query: \( t_1 \bowtie s_1 \bowtie s_2 \)

Optimal query plan

**stream s_1**
- \( id \) 23
- \( attr_3 \) M

**stream s_2**
- \( id \) 12
- \( attr_4 \) Yes

**relation t_1**
- \( id \) 101
- \( s_1 \) 23
- \( s_2 \) 12
- \( attr_1 \) 35
- \( attr_3 \) AB

**Edge Node E_2**

**Edge Node E_4**

**Cloud**

**Edge Node E_5**

**mobile application**

**request query**

**SELECT** \( attr_1, attr_2, attr_3, attr_4 \)

**FROM** \( t_1, s_1[RANGE 30], s_2[RANGE 30] \)

**WHERE** \( t_1.s_1 = s_1.id \) AND \( t_1.s_2 = s_2.id \)
Challenges and Concerns: Edge vs Cloud

• Management policy
  • *Cloud servers are managed through strict and regularized policies*
  • *Edge nodes may not have the same degree of regulatory and monitoring oversight.*
    • Ship selected/projected data to edge nodes that may be untrusted/semi-trusted
    • Lead to disclosure of private information within the edge nodes.

• Latency
  • *Cloud: query in/cross data center(s) → proprietary network bandwidth*
    • Emphasis of QO is primarily on minimizing the query computation time
  • *Edge: nodes are scattered geographically with varying degrees of network connectivity*
    • A special emphasis of QO is network latency or statbility
Latency Analysis

Suppose that edge nodes are located in city A and the closest cloud data center is located in city B.

**Edge-based approach**

\[ t_{edge} = \max_{i \in \{e_2, e_3\}, j \in \{(e_1, e_2), (e_1, e_3)\}} (v_i t / v_{net} + t_j) + T \]

22.223 ms

**Cloud-based approach**

\[ t_{cloud} = \max_{i \in \{e_1, e_2, e_3\}} (v_i t / v_{net} + t_{a,b}) + T + \sum v_i t / v_{net} + t_{a,b} \]

84.818 ms

---

**Table I**

<table>
<thead>
<tr>
<th>Location</th>
<th>Distance</th>
<th>Latency (ms)</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_1 - E_2</td>
<td>d_{e1,e2}</td>
<td>0.022 \cdot d_{e1,e2} + 4.862</td>
<td>t_{e1,e2} = 5.082</td>
</tr>
<tr>
<td>E_2 - E_3</td>
<td>d_{e2,e3}</td>
<td>0.022 \cdot d_{e2,e3} + 4.862</td>
<td>t_{e2,e3} = 5.202</td>
</tr>
<tr>
<td>E_3 - E_1</td>
<td>d_{e3,e1}</td>
<td>0.022 \cdot d_{e3,e1} + 4.862</td>
<td>t_{e3,e1} = 5.192</td>
</tr>
<tr>
<td>A - B</td>
<td>d_{a,b}</td>
<td>0.022 \cdot d_{a,b} + 4.862</td>
<td>t_{a,b} = 26.862</td>
</tr>
</tbody>
</table>

\^ Note that the latency of network traffic is estimated based on the distance using a linear model: \( y = 0.022x + 4.862 \) with coefficient of determination \( R^2 = 0.907 \) proposed in [14].

\^ The distance between the data center and the city is assumed to be 1000 miles, while the distance between edge nodes is 10, 20, and 15 miles, respectively.

**Table II**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t )</td>
<td>30 min</td>
<td>Time interval of the query</td>
</tr>
<tr>
<td>( v_{e1} )</td>
<td>1 KB/min</td>
<td>Speed of stream data generating at edge ( E_1 )</td>
</tr>
<tr>
<td>( v_{e2} )</td>
<td>2 KB/min</td>
<td>Speed of stream data generating at edge ( E_2 )</td>
</tr>
<tr>
<td>( v_{e3} )</td>
<td>3 KB/min</td>
<td>Speed of stream data generating at edge ( E_3 )</td>
</tr>
<tr>
<td>( v_{net} )</td>
<td>100 Mbit/s</td>
<td>Ethernet speed</td>
</tr>
<tr>
<td>( T )</td>
<td>10 ms</td>
<td>Query time in a single machine</td>
</tr>
</tbody>
</table>
Query Processing in Edge Computing

**Join query:** \( t_1 \Join s_1 \Join s_2 \)

**Optimal query plan**

**Stream** \( s_1 \)
- **id:** 23
- **attr:** M

**Edge Node (privacy-sensitive)** \( E_2 \)

**Stream** \( s_2 \)
- **id:** 12
- **attr:** Yes

**Relation** \( t_1 \)
- **id:** 101
- **s1:** 23
- **s2:** 12
- **attr1:** 35
- **attr2:** AB

**SELECT** \( attr_1, attr_2, attr_3, attr_4 \)

**FROM** \( t_1, s_1 \text{[RANGE 30]}, s_2 \text{[RANGE 30]} \)

**WHERE** \( t_1.s_1 = s_1.id \text{ AND } t_1.s_2 = s_2.id \)
Privacy Disclosure Risk

Suppose that $E_4$ is controlled by the adversary who tries to collect users’ private information. As a result, the adversary at the public edge node $E_4$ can acquire the intermediate sensitive data even if it does not have access to edge nodes where the sensitive data is stored.

Samples of query execution plan candidates
Adversary Model

- **Public Adversary**
  - **has complete control of public edge nodes**
  - **can access any data stored in public edge nodes**

- **Private Adversary**
  - **can access the private edge nodes belonging to a specific privacy level**

-the adversary can access any intermediate data shipped to its controlled edge nodes during the query plan execution phase

→ the intermediate data inference attack.
Privacy Guarantee

• No privacy-sensitive information is disclosed in the query processing phase in the edge computing.
  • *if an adversary controls a public edge node*
    • it will not infer any privacy-sensitive information from monitoring the query operations
  • *even if the adversary controls a private edge node with privacy level* $p$
    • it cannot infer any sensitive information with privacy level higher than $p$
Query Guard Framework

• A traditional dynamic programming enumeration skeleton
  • the optimal plan is generated by joining optimal sub-plans in a bottom-up manner

• Specifically
  • Iterative dynamic programming approach
  • Heuristic-based methods

Algorithm 1: Pseudocode for QueryGuard framework

```
Input: A set of relations or streams $R = \{R_i\}$ with size $n$ generated from a query $Q$
Output: The optimized query plan

for $i = 1$ to $n$
do
  plans($\{R_i\}$) := access-plans($\{R_i\}$)
  LATENCY-AWARE-PRUNE(plans($\{R_i\}$))
  toDo := $R$
while $|\text{toList}| > 1$
do
  $b := \text{balanced-parameter}(|\text{toList}|, k)$
  for $i = 2$ to $b$
do
    $S \in R$ and $|S| = i$
do
      plans($S$) := $\emptyset$
      forall $O \subseteq S$ and $O \neq \emptyset$
do
        plans($S$) := plans($S$) $\cup$ PRIVACY-JOIN(plans($O$), plans($S \setminus O$))
        LATENCY-AWARE-PRUNE(plans($S$))
      find $P, V$ with $P \in \text{plans}(V)$, $V \subseteq \text{toList}$, $|V| = k$ such that eval($P$) = min{eval($P'$) | $P' \in \text{plans}(W)$, $W \subseteq \text{toList}$, $|W| = k$}
      generate new symbol: $T$, plans($T$) = $\{P\}$
      toDo = toDo $\setminus V \cup \{T\}$
    forall $O \subseteq V$
do
      delete(plans($O$))
  finalize-plans(plans($R$))
  LATENCY-AWARE-PRUNE(plans($R$))
return plans($R$)
```
Privacy Join

• Privacy Settings
  • Privacy Preference
    • the data is assigned a privacy preference parameter by data owner to control the data shipment scope
    • no ship out-of-scope in join operation
  • Privacy Level
    • each edge node is assigned a privacy level
    • the privacy level of data can be directly inferred from the privacy levels of edge nodes
    • no ship down in join operation
An illustration of the critical phases in Query Guard

Possible Joins

Privacy Levels: \( PL_4 > PL_3 > PL_2 > PL_1 \)

Algorithm 1: Pseudocode for QueryGuard framework

```
Input: A set of relations or streams \( R = \{ R_i \} \) with size \( n \) generated from a query \( Q \)
Output: The optimized query plan
for \( i = 1 \) to \( n \) do
    plans(\( \{ R_i \} \)) := access-plans(\( \{ R_i \} \))
    LATENCY-AWARE-PRUNE(plans(\( \{ R_i \} \)))
toDo := \( R \)
while \( |\text{toDo}| > 1 \) do
    \( b := \text{balanced-parameter(|toDo|, k)} \)
    for \( i = 2 \) to \( b \) do
        for all \( S \subset R \) and \( |S| = i \) do
            plans(\( S \)) := \( \emptyset \)
            forall \( O \subset S \) and \( O \neq \emptyset \) do
                plans(\( S \)) := plans(\( S \)) \cup PRIVACY-JOIN(plans(\( O \)), plans(\( S \setminus O \)))
                LATENCY-AWARE-PRUNE(plans(\( S \))
find \( P, V \) with \( P \in \text{plans}(V), V \subset \text{toDo}, |V| = k \) such that\( \text{eval}(P) = \text{min}\{\text{eval}(P') | P' \in \text{plans}(W), W \subset \text{toDo}, |W| = k \} \)
generate new symbol: \( T, \text{plans}(T) = \{ P \} \)
toDo := toDo \setminus V \cup \{ T \}
for all \( O \subset V \) do
    delete(plans(\( O \)))
finalise-plans(plans(\( R \)))
LATENCY-AWARE-PRUNE(plans(\( R \)))
return plans(\( R \))
```
An illustration of the critical phases in Query Guard

Privacy-preserving Joins

Privacy Levels: PL₄ > PL₃ > PL₂ > PL₁
An illustration of the critical phases in Query Guard

Latency-aware Prune

Privacy Levels: $PL_4 > PL_3 > PL_2 > PL_1$

**Algorithm 3: Latency-aware function**

1. function $LATENCY$-\textit{AWARE}$-$\textit{PRUNE}$(plans(S))
2. result := $\{\}$;
3. foreach site $e$ do $t[e]$ := null;
4. foreach plan $p$ in $plans(S)$ do
5.   $c :=$ extract the catalog information;
6.   if $f_c(p) < f_c(t[e])$ such $t[e] \neq \text{null}$ then $t[e] := p$;
7. foreach site $e$ do result.add($t[e]$) such $t[e] \neq \text{null}$;
8. return result

\[ f_{cost}(\mathcal{L}) = C_{cent} + \sum_{(e_i \rightarrow e_j) \in \mathcal{L}} (n_{e_i \rightarrow e_j} \cdot t_{e_i \rightarrow e_j}^{estimate}) \]

\[ t_{e_i \rightarrow e_j}^{estimate} = \alpha \cdot t_{avg} / n_{send} \]

\[ \arctan(d_{geo}(e_i, e_j)) \cdot 2 / \pi \]
Experimental Evaluation

• General Setup
  • Simulate a set of edge nodes with artificially injected network latency
    • 15 edge nodes with specific geography information
    • Latency (ms) of the network traffic is estimated based on the distance (miles) using a linear model
      • \( y = 0.022x + 4.862 \)

All the experiments were executed using randomly generated queries over randomly generated relations/streams that are distributed on the 15 edge nodes.

---

<table>
<thead>
<tr>
<th>Edge Node Address</th>
<th>Privacy Level</th>
<th>Geography</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.0.1.1-8</td>
<td>{0,0,1,2,3,4,5}</td>
<td>Area nearby Pittsburgh, PA</td>
</tr>
<tr>
<td>10.0.1.9</td>
<td>0</td>
<td>Erie, PA</td>
</tr>
<tr>
<td>10.0.1.10</td>
<td>1</td>
<td>Philadelphia, PA</td>
</tr>
<tr>
<td>10.0.1.11</td>
<td>2</td>
<td>Allentown, PA</td>
</tr>
<tr>
<td>10.0.1.12</td>
<td>3</td>
<td>Harrisburg, PA</td>
</tr>
<tr>
<td>10.0.1.13</td>
<td>0</td>
<td>Cleveland, OH</td>
</tr>
<tr>
<td>10.0.1.14</td>
<td>2</td>
<td>Morgantown, WV</td>
</tr>
<tr>
<td>10.0.1.15</td>
<td>3</td>
<td>Washington D.C.</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>Relation/Stream</th>
<th>Edge Node</th>
<th>Transmission Threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>10.0.1.{3,4,5,7,10,14,15}</td>
<td>10.0.1.{1-12}</td>
</tr>
<tr>
<td>B2</td>
<td>10.0.1.{6,8,11,12}</td>
<td>10.0.1.{1-12}</td>
</tr>
<tr>
<td>C3</td>
<td>10.0.1.{2,6,11}</td>
<td>10.0.1.{1-12}</td>
</tr>
<tr>
<td>D4</td>
<td>10.0.1.{2,4,5,6,11,12,13}</td>
<td>10.0.1.{1-12}</td>
</tr>
<tr>
<td>E5</td>
<td>10.0.1.{4,12,13}</td>
<td>10.0.1.{1-12}</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>Relation Type</th>
<th>Cardinality of Relation</th>
<th>Simulation Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>10-100</td>
<td>5%</td>
</tr>
<tr>
<td>II</td>
<td>100-1000</td>
<td>15%</td>
</tr>
<tr>
<td>III</td>
<td>1,000-10,000</td>
<td>30%</td>
</tr>
<tr>
<td>IV</td>
<td>10,000-100,000</td>
<td>30%</td>
</tr>
<tr>
<td>V</td>
<td>100,000-100,000,000</td>
<td>15%</td>
</tr>
<tr>
<td>VI</td>
<td>1,000,000-10,000,000</td>
<td>5%</td>
</tr>
</tbody>
</table>

† The cardinality of a stream indicates the size of synopsis in DSMS.
Experimental Evaluation

A case study of privacy-preserving processing

A1 ⊗ B2 ⊗ C3 ⊗ D4 ⊗ E5
Experimental Evaluation

• Comparison to IDP1
  • Execution Time
  • our proposed technique has non-negligible performance advantage in execution time
Experimental Evaluation

• Comparison to IDP1
  • Memory Usage
  • our proposed technique has non-negligible performance advantage in memory usage aspects.
Experimental Evaluation

• Effect of latency awareness setting
  • to evaluate whether the latency-aware cost model influences the performance of our proposed framework

• The latency-aware setting has a negligible effect on the memory usage of the algorithm, while the execution time cost has slight growth when the relation number increase.
Conclusion

• A privacy-preserving latency-aware query optimization framework
  • *Privacy disclosure risk analysis*
  • *Latency concerns analysis*

• *Tackled privacy-aware and latency optimized query processing in edge computing environments*
• *Evaluate the proposed techniques in terms of execution time and memory usage*
  • our results show that the proposed methods perform better than conventional techniques while achieving the intended privacy goals.
Q & A

Thanks