FROM A STREAM OF RELATIONAL QUERIES TO DISTRIBUTED STREAM PROCESSING

Qiong Zou, Huayong Wang, Robert Soule, Martin J Hirzel, Henrique Andrade, Bugra Gedik, Kun-lung Wu

Agenda

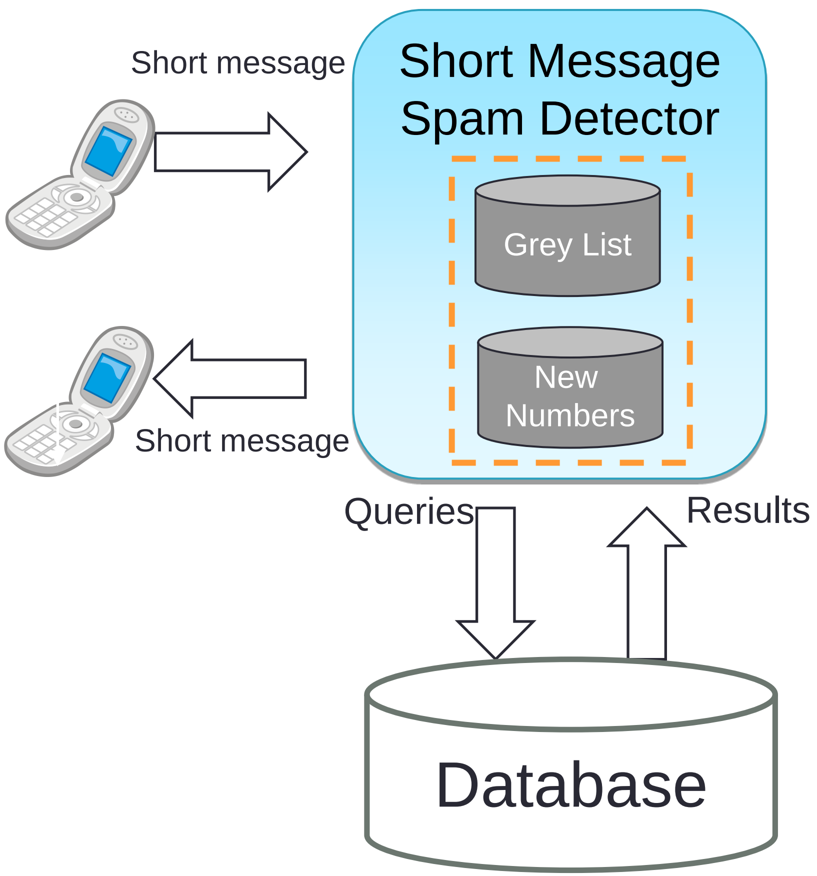
- ✓Motivation
- ✓Example: Spam Short Message Filtering
- ✓Solution: Stream-Based DB Cache
  - ✓Step 1: Identify critical queries
  - ✓Step 2: Replace ODBC driver
  - ✓Step 3: Generate stream program
  - ✓Step 4: Partition the data set
- ✓Performance Evaluation
- ✓Conclusion

Agenda

- ✓Motivation
- ✓Example: Spam Short Message Filtering
- ✓Solution: Stream-Based DB Cache
  - ✓Step 1: Identify critical queries
  - ✓Step 2: Replace ODBC driver
  - ✓Step 3: Generate stream program
  - ✓Step 4: Partition the data set
- ✓Performance Evaluation
- ✓Conclusion

Motivation from Applications

Spam Short Message Filtering (SSMF)

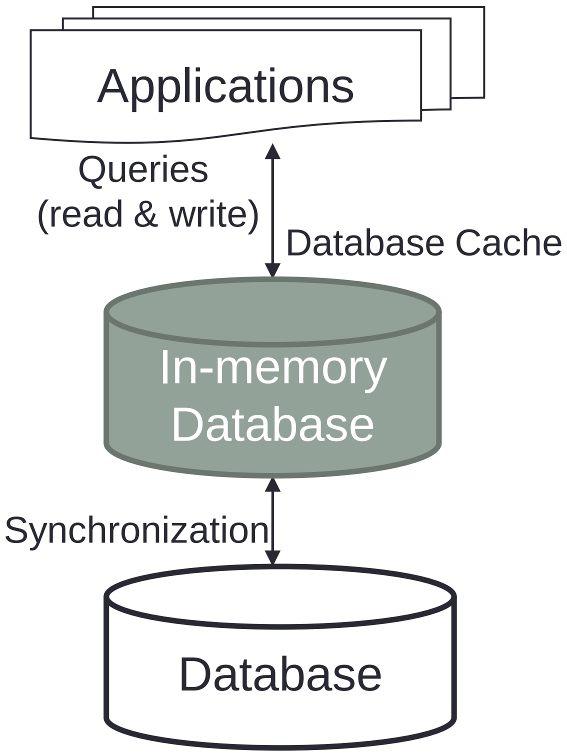


- ✓ **SSMF represents a category of applications, their characteristics are:**
  - ✓ Most of time is spent on database operations
  - ✓ Infrequently modified rules/models/data
  - ✓ Tolerate out-of-date data
- ✓ **Breakdown for query:**
  - ✓ Query execution
  - ✓ Disk IO
  - ✓ Locking
  - ✓ Logging and Transaction

  assure ACID
- ✓ **High throughput and low latency are required**
  - ✓ >6GB data, 50 queries/sec, latency < 0.1sec
  - ✓ Disk IO/Locking/Logging in database are burdens
  - ✓ **How to remove them?**

Traditional Solution – DB Cache

- ✓ **Using in-memory database as a database cache**
  - ✓ still has costs like locking, transactions and logging
  - ✓ performance of distributed in-memory databases is not good yet

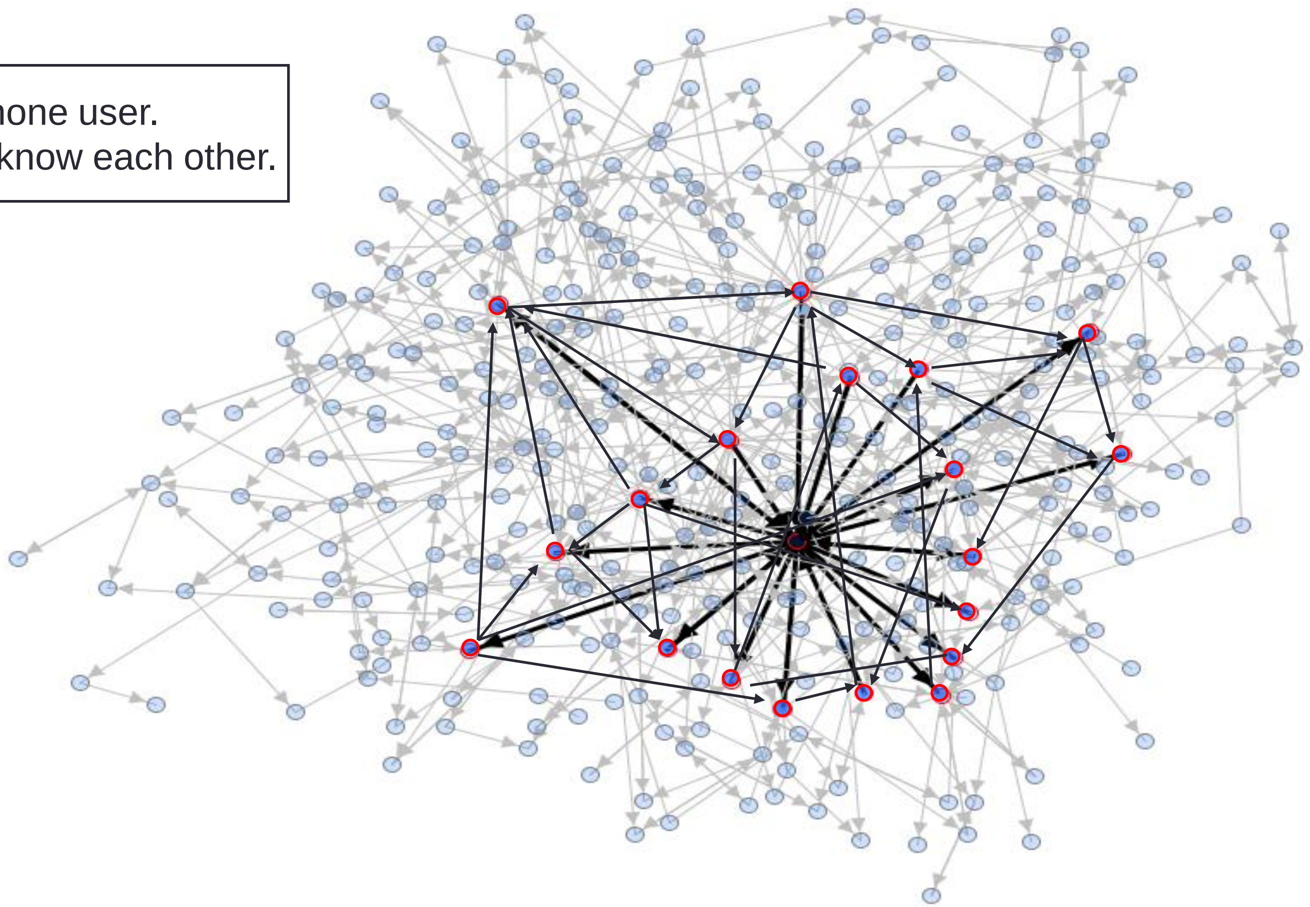


Agenda

- ✓Motivation
- ✓Example: Spam Short Message Filtering
- ✓Solution: Stream-Based DB Cache
  - ✓Step 1: Identify critical queries
  - ✓Step 2: Replace ODBC driver
  - ✓Step 3: Generate stream program
  - ✓Step 4: Partition the data set
- ✓Performance Evaluation
- ✓Conclusion

Example Application – Spam Short Message Filtering

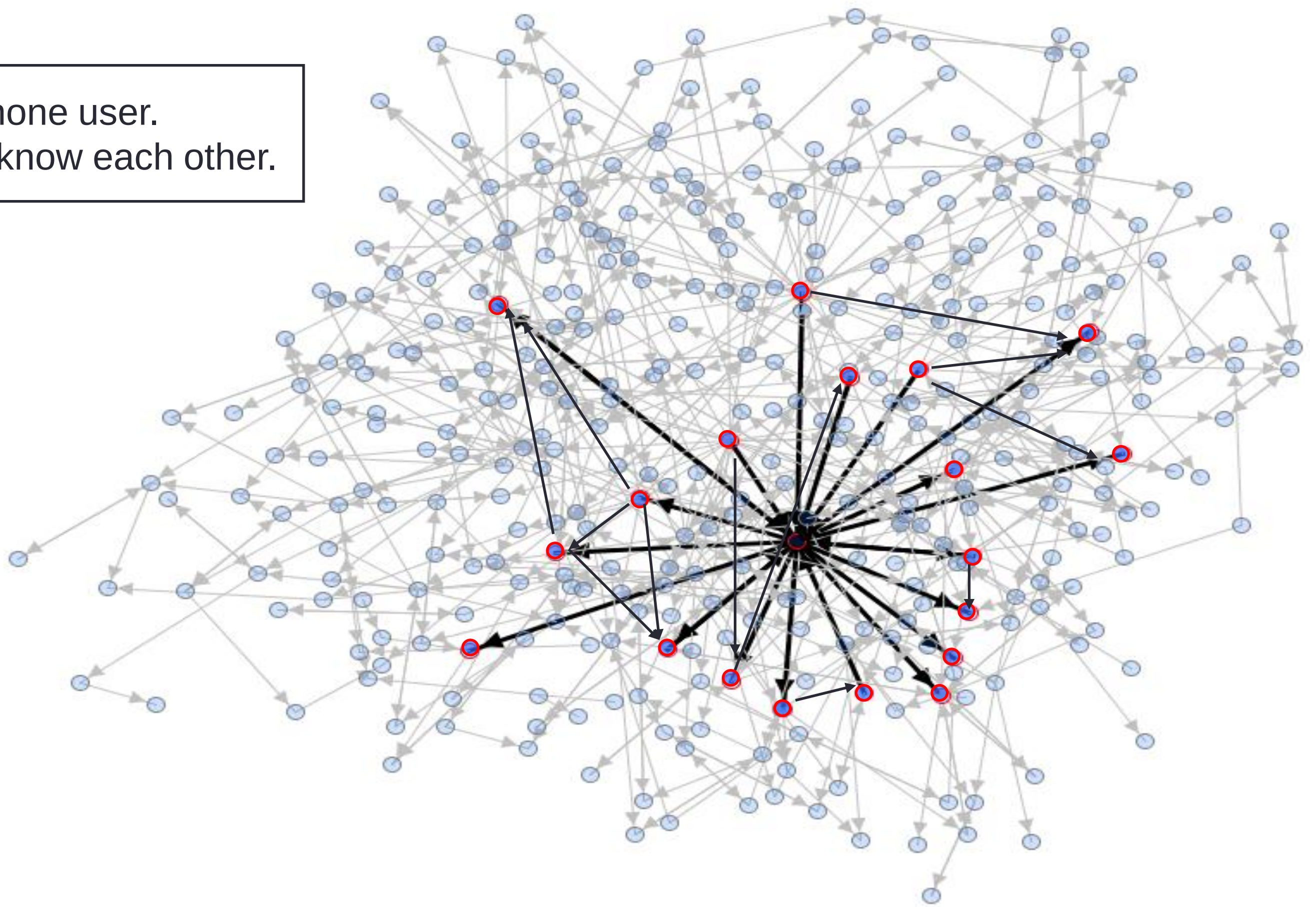
Vertex: Mobile phone user.
Edge: two users know each other.



In a normal case, the short messages from a particular user are sent to a group of people who know each other.

Example Application – Spam Short Message Filtering

Vertex: Mobile phone user.
Edge: two users know each other.



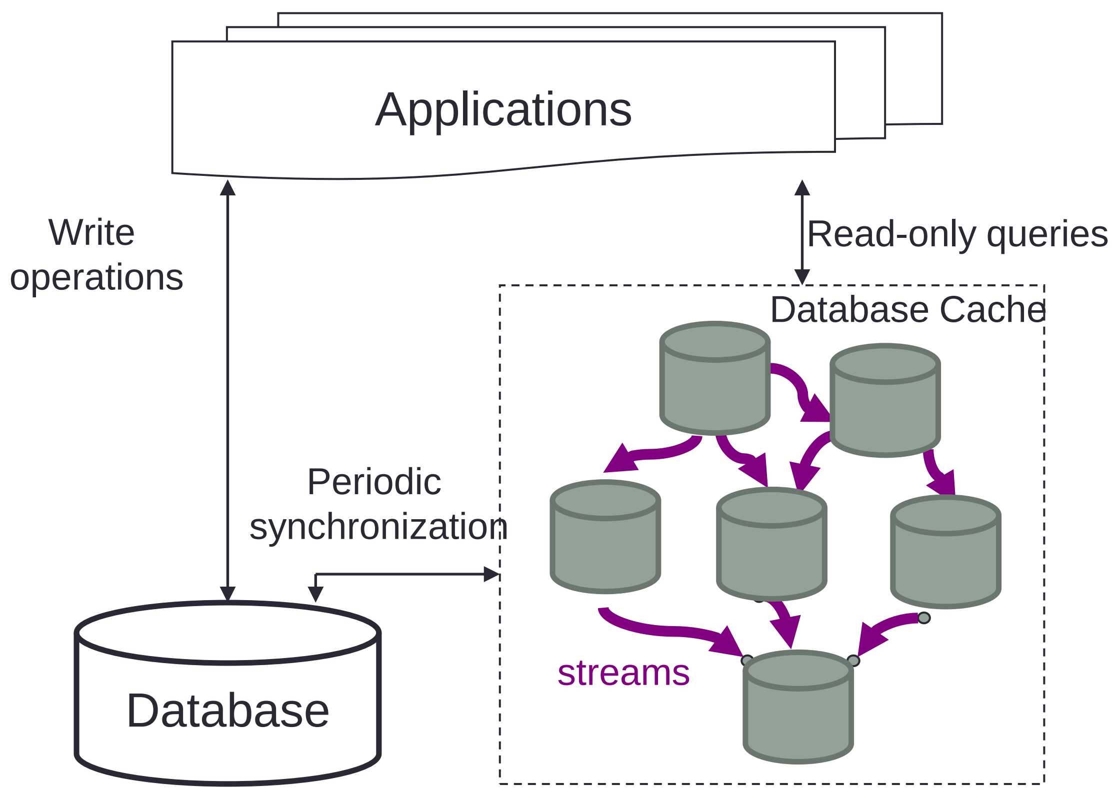
In a case of spam short message, the messages from a particular user are sent to a group of people who rarely know each other.

Agenda

- ✓Motivation
- ✓Example: Spam Short Message Filtering
- ✓Solution: Stream-Based DB Cache
  - ✓Step 1: Identify critical queries
  - ✓Step 2: Replace ODBC driver
  - ✓Step 3: Generate stream program
  - ✓Step 4: Partition the data set
- ✓Performance Evaluation
- ✓Conclusion

Our Solution: Stream-Based DB Cache

- ✓ **Using stream computing system as a database cache**
  - ✓ High performance (provide only query processing).
  - ✓ Transparency (few modification to application source code)
  - ✓ Scalability (naturally parallel execution)



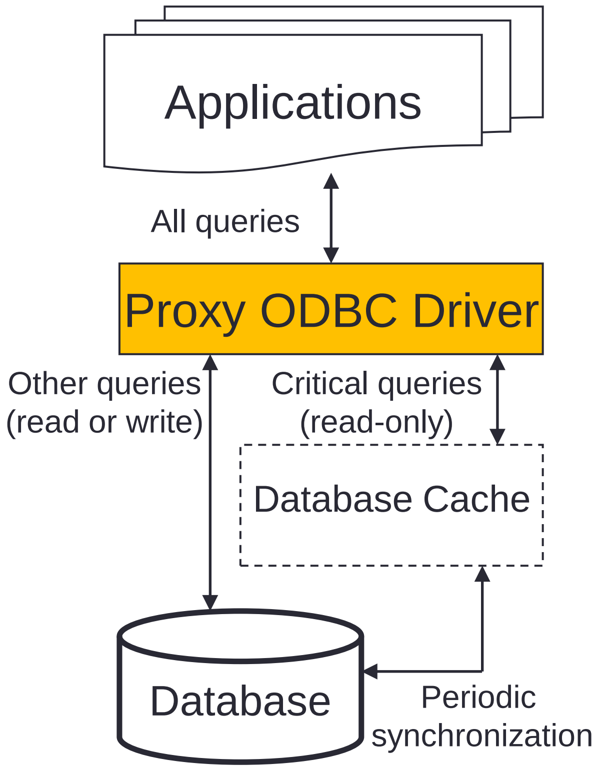
Step 1: Identify critical queries

Step1: Profile the application and identify critical queries – queries that account for the bulk of the application execution time.

1) Collect timing information regarding each ODBC call.
2) Aggregate time information for each query.
3) Rank the queries based on the time.

Step2: Replace ODBC driver

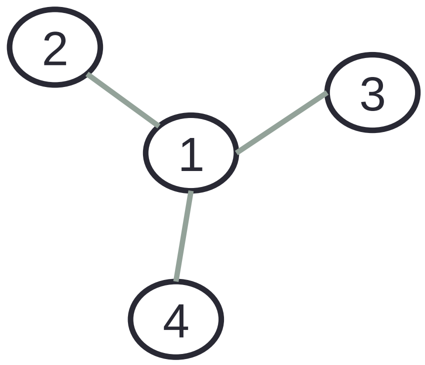
- ✓ **In a Java/C++/C program, we only need change JDBC/ODBC driver name for the whole program.**



The Graph and Query

The graph stored in relational database

src	dst
1	2
1	3
1	4



To know how many of User1's neighbors know each other

SELECT COUNT(*) FROM graph
WHERE src IN { SELECT dst FROM graph WHERE src = 1 }
AND dst IN { SELECT dst FROM graph WHERE src = 1 }

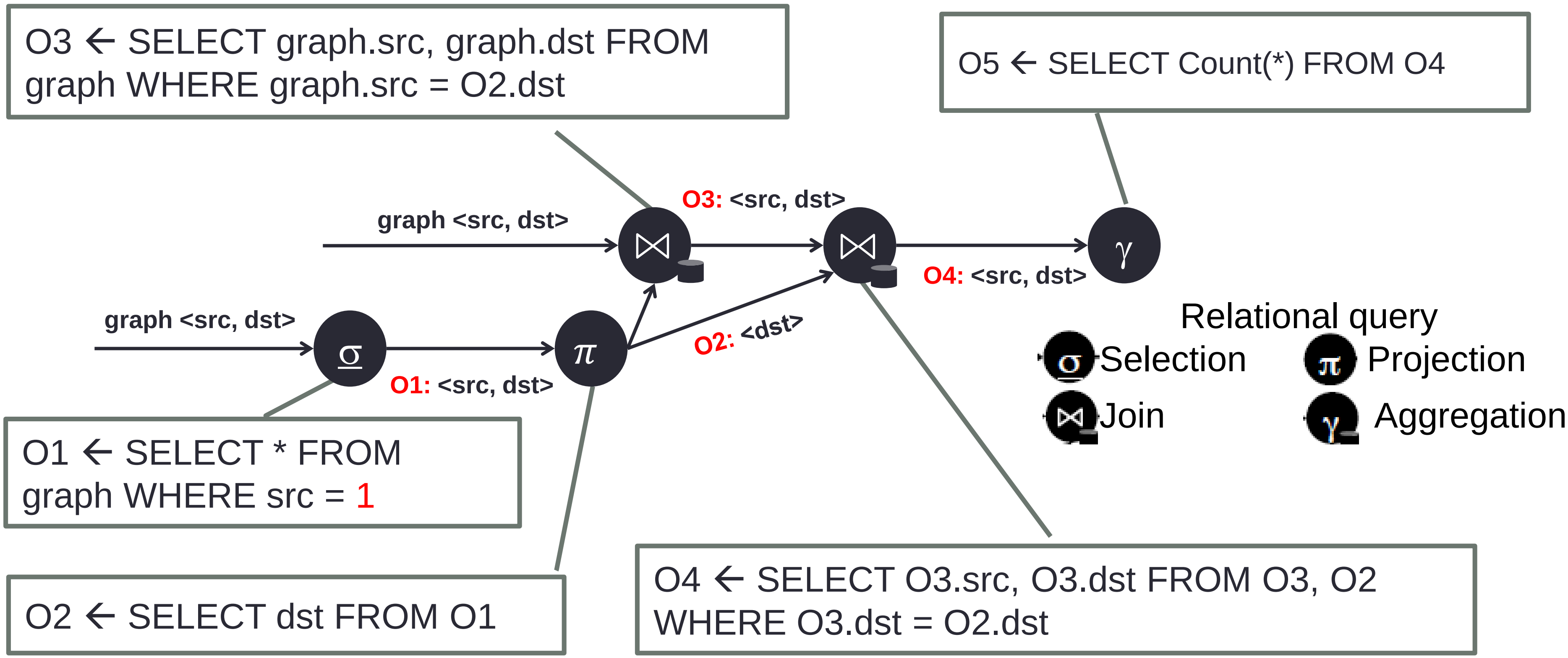
1. To select out User1's neighbors

Neighbors ← SELECT dst FROM graph WHERE src = 1

2. To select out the sub-graph by User1's neighbors

SELECT * FROM graph WHERE src IN Neighbors AND dst IN Neighbors

Step 3: Generate stream program: SQL->CQL

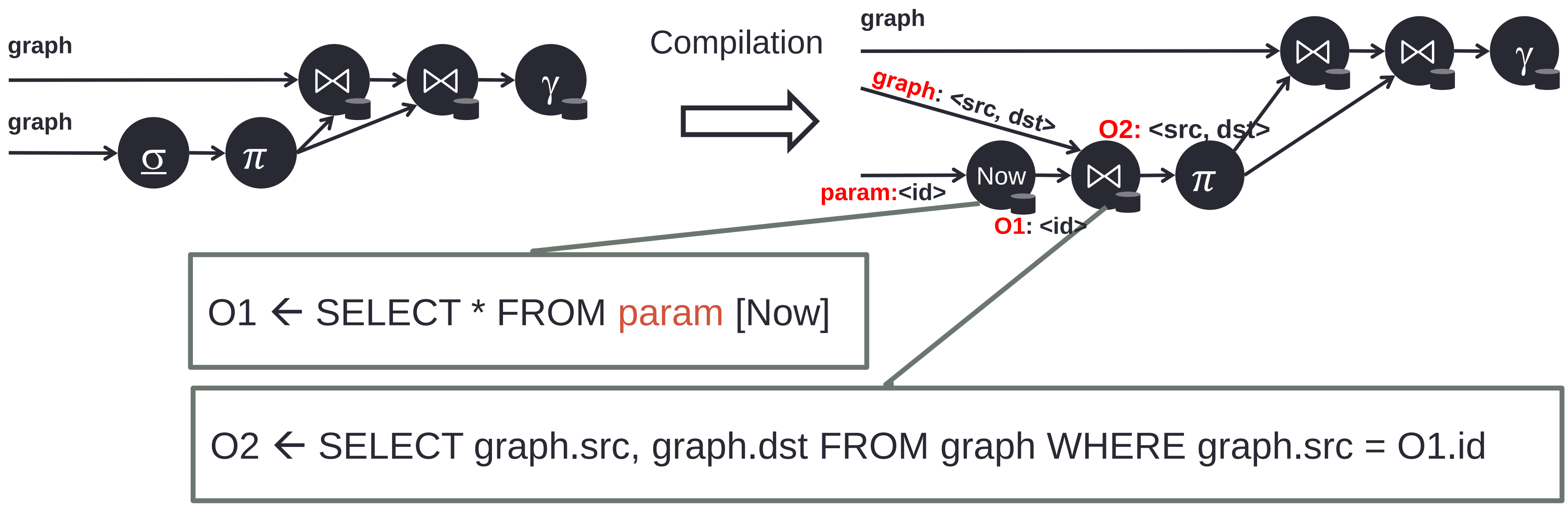


SELECT COUNT(*) FROM graph
WHERE src IN { SELECT dst FROM graph WHERE src = 1 }
AND dst IN { SELECT dst FROM graph WHERE src = 1 }

Step 3: Generate stream program: CQL->Stream

CQL query

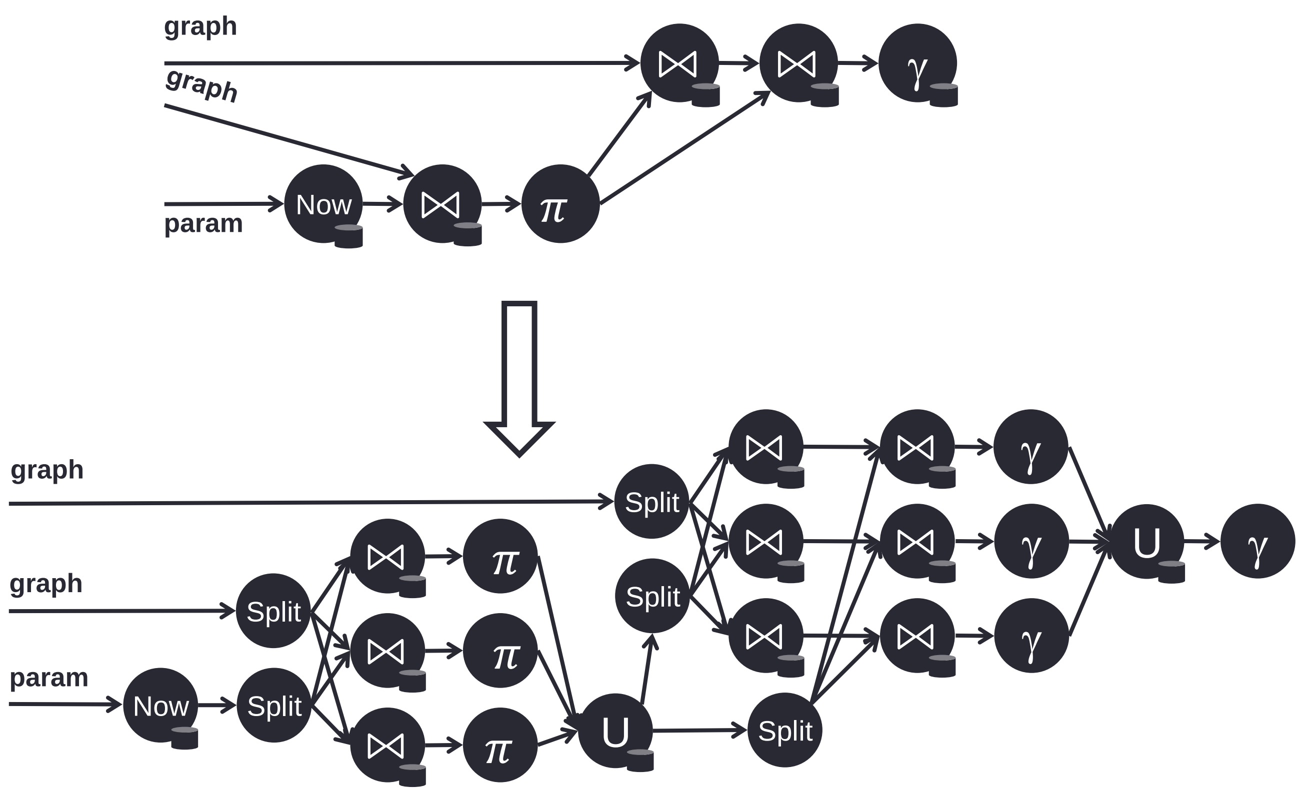
Continuous query (stream)



SELECT COUNT(*) FROM graph
WHERE src IN { SELECT dst FROM graph WHERE src = param}
AND dst IN { SELECT dst FROM graph WHERE src = param}

Step 4: Partition the data set

✓**Partition the dataset and load data into stream program.**



After partitioning portions of the graph into multiple replicas, the stream query becomes complex. The compiler handles all the complexity.

Agenda

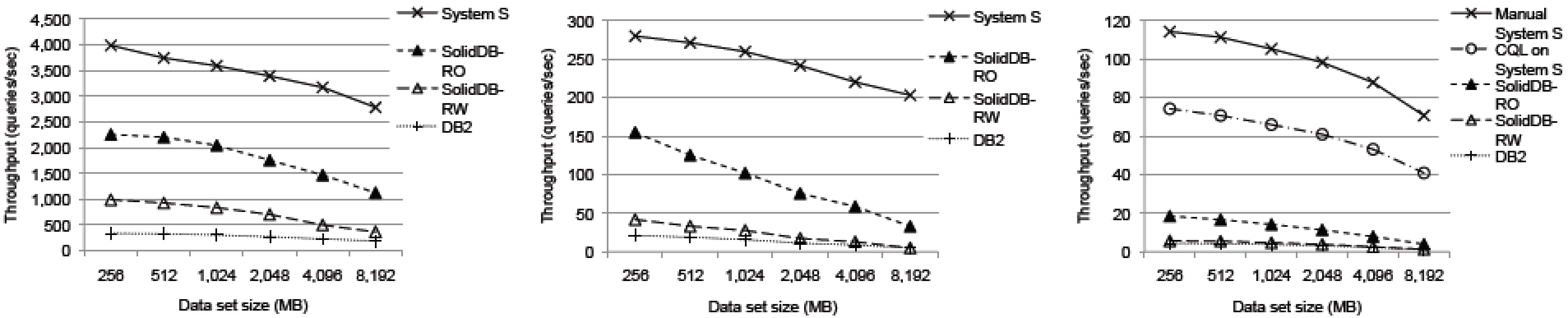
- ✓Motivation
- ✓Example: Spam Short Message Filtering
- ✓Solution: Stream-Based DB Cache
  - ✓Step 1: Identify critical queries
  - ✓Step 2: Replace ODBC driver
  - ✓Step 3: Generate stream program
  - ✓Step 4: Partition the data set
- ✓Performance Evaluation
- ✓Conclusion

The Other Two Applications

- **Trajectory mapping**: To resolve the shortest path problem with additional constraints. Incompleteness of GPS data is a problem for trajectory mapping. Shortest path is a reasonable criterion to speculate the missing points. Usually road conditions can be extra constraints.
- **Market Intelligence Portal**: A search engine system for market information. The collected data are stored in a relational database server. The repository is then mined and the resulting information can be mapped onto predefined taxonomies.
- **Common Characteristics**:
  - Originally using database (large data volume)
  - Having streaming paradigm in nature
  - Tolerate slightly-stale data (No ACID is required)

Performance Evaluation – Throughput (1)

Throughput by varying the dataset size



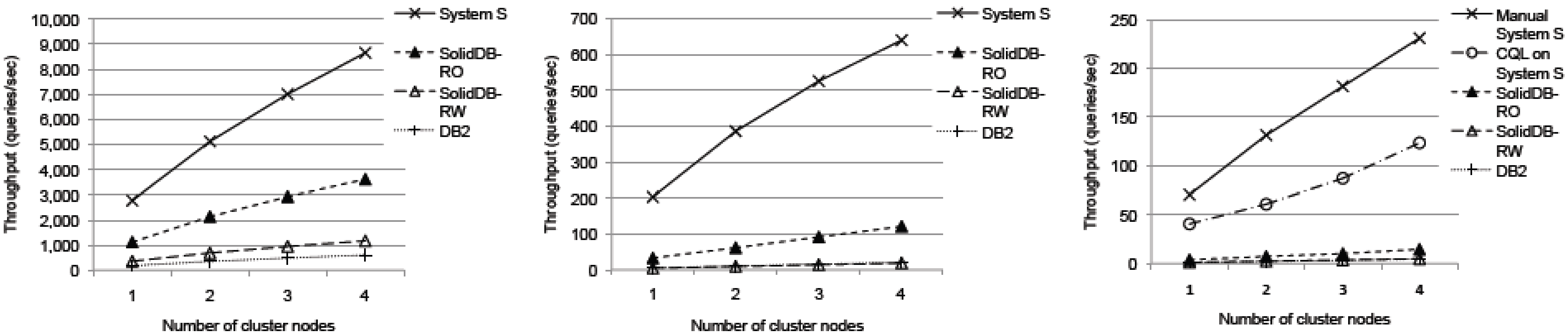
(a) Trajectory mapping. (b) Market intelligence portal. (c) Spam short message filtering.

Throughput vs. data set size for critical queries, on one cluster node.

System S achieves 3 to 10 fold speedup over base DB2, 1.5 to 2 fold speedup to SolidDB-RO

Performance Evaluation – Throughput (2)

Throughput by varying the number of cluster nodes



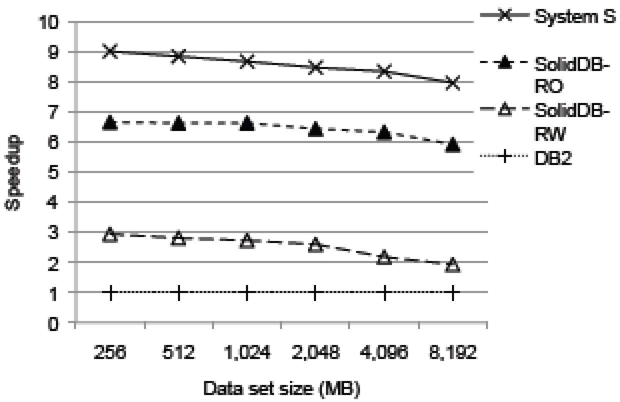
(a) Trajectory mapping. (b) Market intelligence portal. (c) Spam short message filtering.

Throughput vs. number of nodes, with 8GB data set for each application.

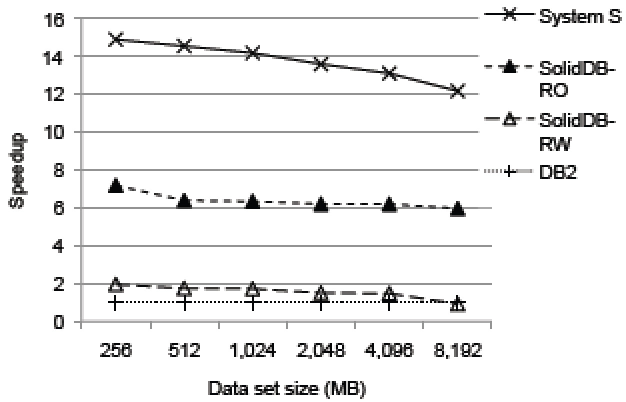
Scaling almost linearly with increase of computational resources

Performance Evaluation -- Latency

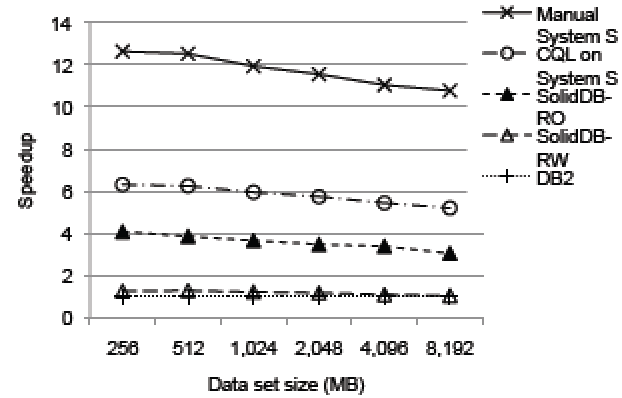
Latency by varying the dataset size



(a) Trajectory mapping.



(b) Market intelligence portal.



(c) Spam short message filtering.

Normalized latency vs. data set size for critical queries, on one cluster node.

System S achieves 8 to 15-fold speedup to base DB2

Agenda

- ✓Motivation
- ✓Example: Spam Short Message Filtering
- ✓Solution: Stream-Based DB Cache
  - ✓Step 1: Identify critical queries
  - ✓Step 2: Replace ODBC driver
  - ✓Step 3: Generate stream program
  - ✓Step 4: Partition the data set
- ✓Performance Evaluation
- ✓Conclusion

Conclusion

- ✓ We proposed a mechanism for converting a DB-based data analysis application into a streaming application
- ✓ A category of applications would benefit and gain performance improvements from this mechanism
- ✓ The properties of them are:
  - ✓ Streaming Paradigm
  - ✓ Tolerate slightly-stale data
  - ✓ Large volume of data
  - ✓ Partitionable datasets

Thank You!

Some Questions

- ✓**Can database cache be faster than database?**
  - ✓Unlike hardware cache, the performance gain of database cache usually comes from its simplicity and saving more data in memory.
- ✓**Is stream-based database cache different from other database cache?**
  - ✓Only cache read-only data, support periodic modification in batch. No burden of maintain synchronization between cache and database.
  - ✓Pre-specify what queries are to be executed in the cache.
- ✓**Will data be inconsistent in cache and database?**
  - ✓Yes. The application is required to tolerate this side-effect.
- ✓**Is it only usable for a very limited applications?**
  - ✓We have showcased three real-world applications in the paper. We believe a reasonable range of applications can benefit from this solution.