SCHISM: A WORKLOAD-DRIVEN APPROACH TO DATABASE REPLICATION AND PARTITIONING

ZEYNEP KORKMAZ
CS742 - PARALLEL AND DISTRIBUTED DATABASE SYSTEMS
UNIVERSITY OF WATERLOO

OUTLINE

1. Background
2. What is Schism?
3. Cost of Distributed Transactions
4. Partitioning and Replication
5. Optimization
6. Experiments
7. Conclusion
8. Discussion - Critiques

BACKGROUND

Problem:

- Scaling database workloads

Solution:

- Partioning minimize the number of nodes involved in answering a query
  - Round-robin
  - Range
  - Hash
- Social networking workloads
  - Hard to partition

BACKGROUND

Problem:

- Distributed transactions are expensive

Solution:

- Minimize the number of distributed transactions, while producing balanced partitions
  - **Schism**

SCHISM

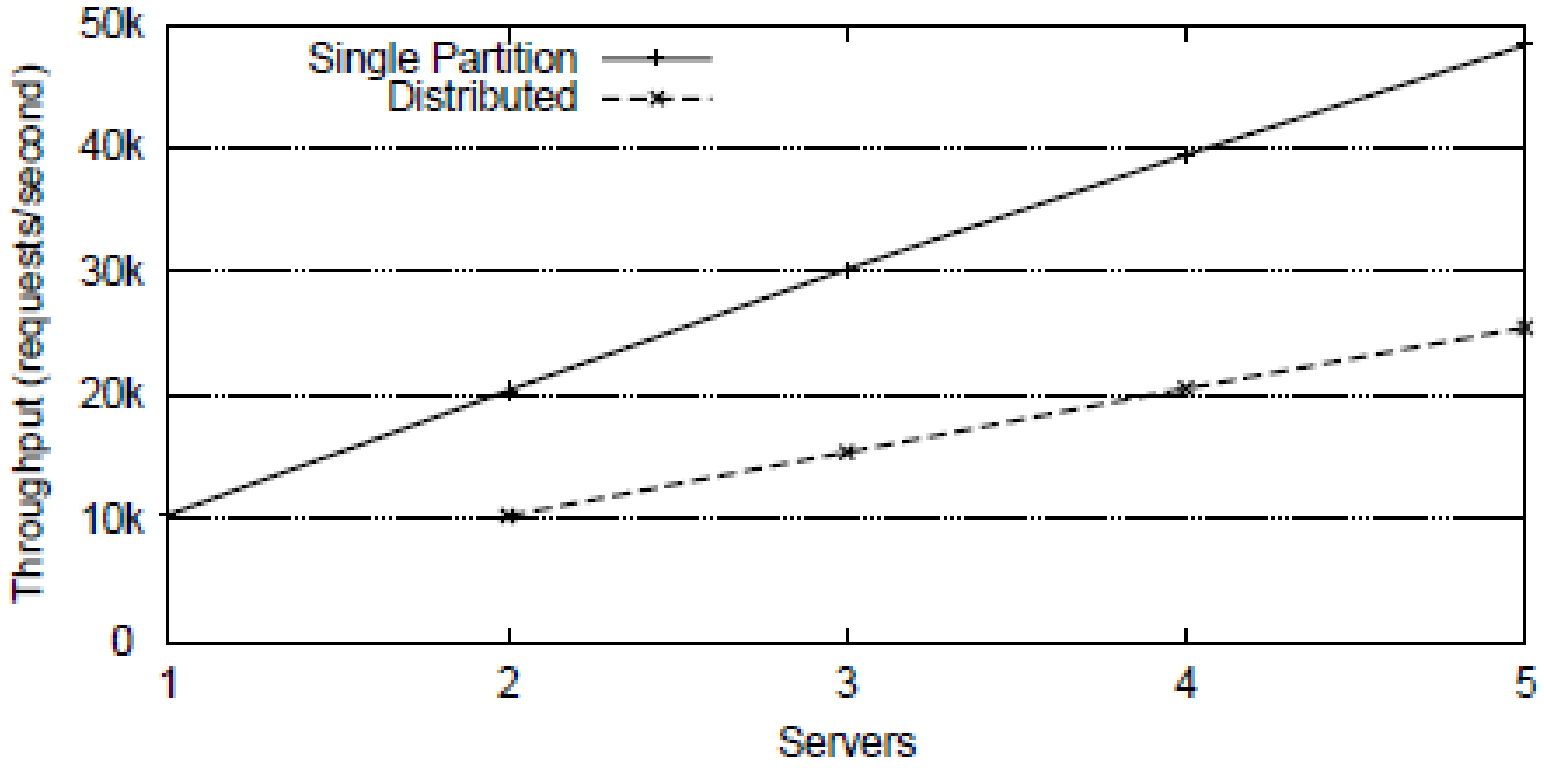
- **A novel graph-based, data driven partitioning system for transactional workloads.**
  - Data pre-processing
    - Input: trace of transactions & DB
    - Read and write sets
  - Creating the graph
    - Nodes: tuples – Edges: transactions
  - Partitioning the graph
    - Balanced min-cut partitioning & replication
  - Explaining the partition
    - Decision tree on frequent attribute set
  - Final validation
    - The best strategy?

COST OF DISTRIBUTED TRANSACTIONS

- **Transactions access data on a single node**
  - No additional overhead
- **Distributed transactions are expensive:**
  - Contention: Overheads of locking
  - Distributed deadlocks
  - Complex statements need to access data from multiple servers
- **Experiment:**
  - Single transaction→two rows; issuing two statements
    - Every transaction is run on a server
    - Every transaction is distributed

COST OF DISTRIBUTED TRANSACTIONS

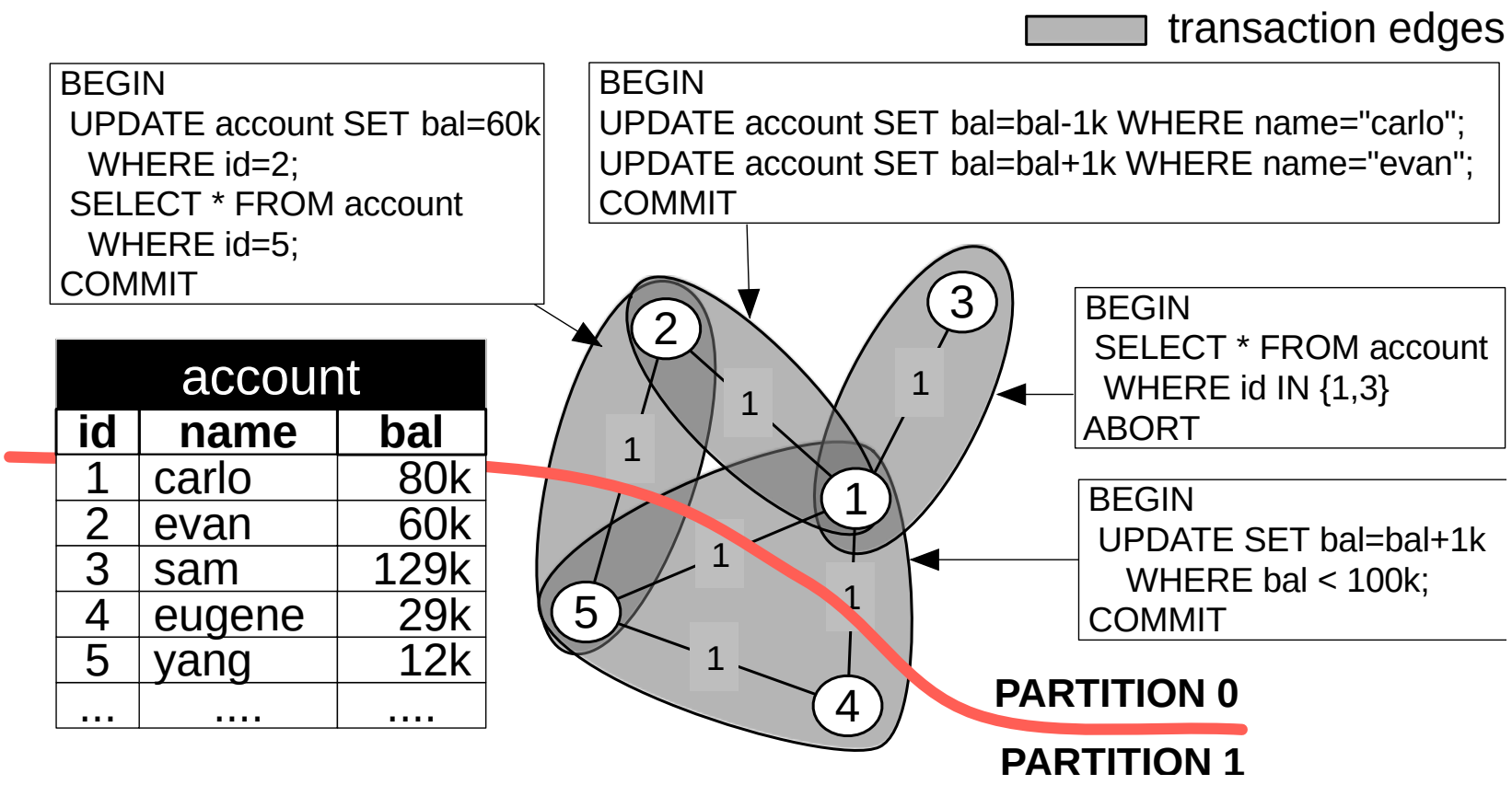
- **Reducing throughput by a factor of 2**
- **Double the average latency**



GRAPH REPRESENTATION

- **Graph representation: build a graph from transaction traces**
  - Node: tuple
  - Edges: usage of the tuples within a transaction
  - Edge weights: #transactions that co-access a pair of tuples
- **Hypergraphs?**
- **Extension: replicated tuples**

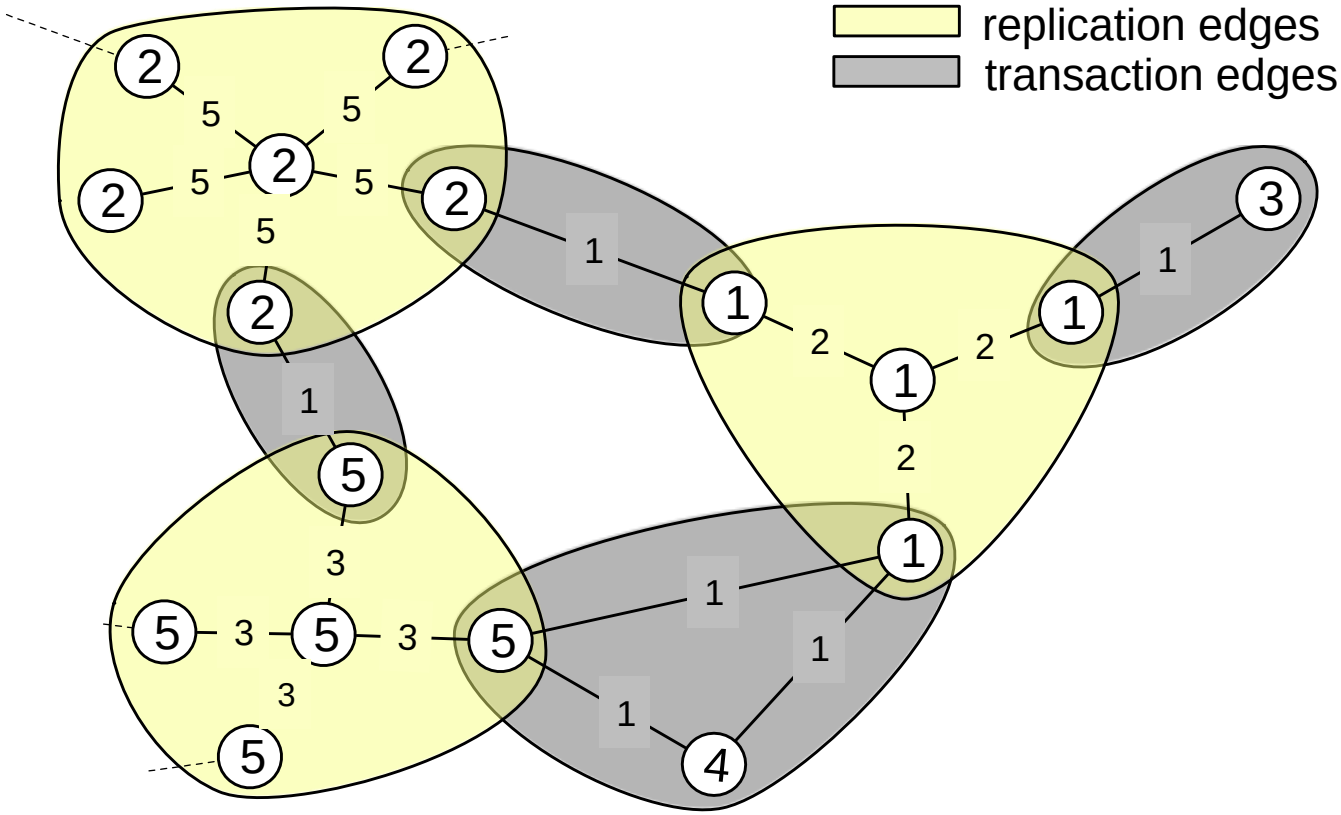
GRAPH REPRESENTATION



GRAPH WITH REPLICATION

- **Extension to basic graph representation:**
  - Tuple-level replication
  - A singel node: a singel tuple (basic graph)
- **Star-shaped configuration:**
  - n+1 nodes: a single tuple
  - n: #transactions that access the tuple
  - Replication edge weights: #transactions that update the tuple in the workload

GRAPH WITH REPLICATION

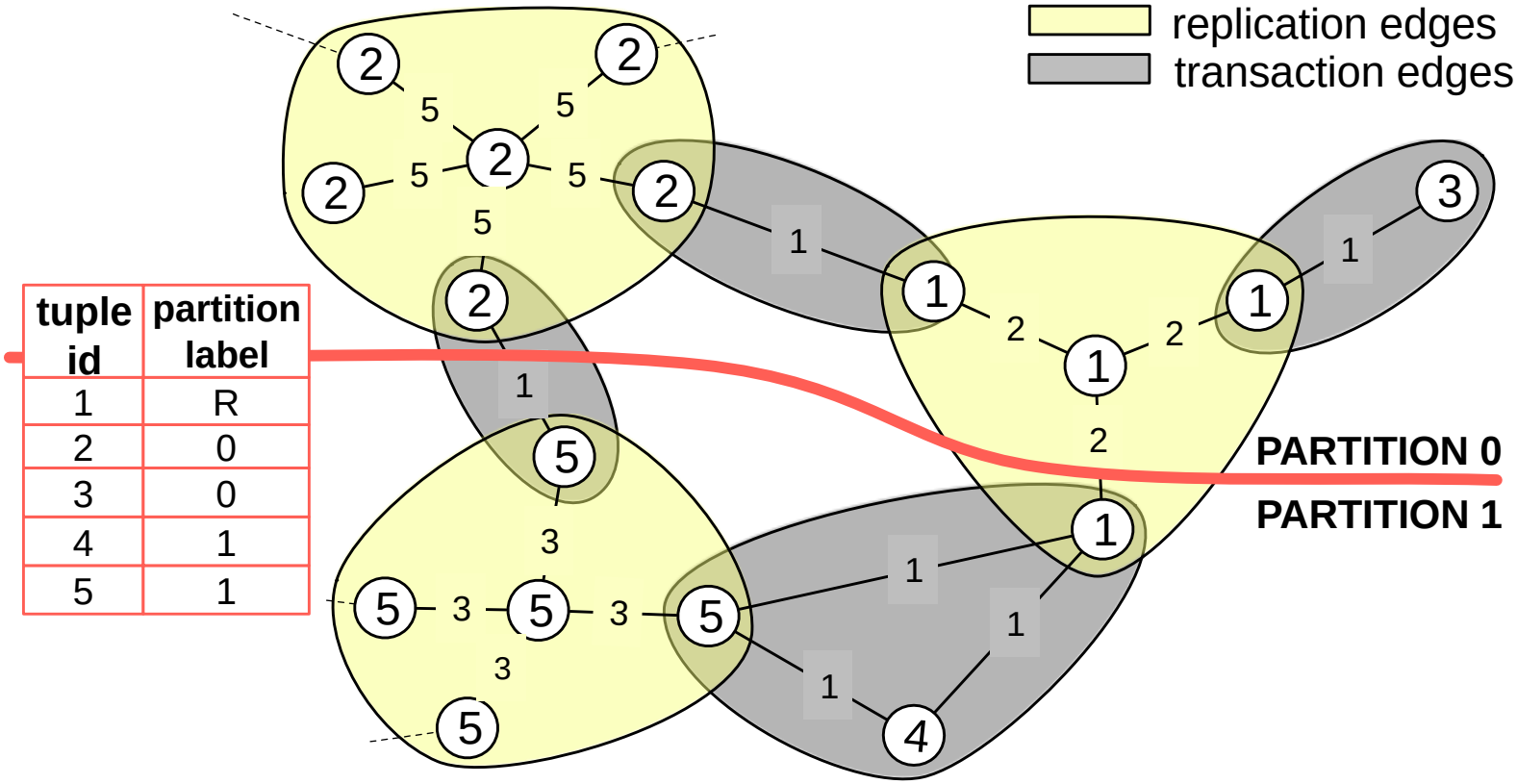


GRAPH PARTITIONING

- **Splits graph into k non-overlapping partitions:**
  - Overall cost of the cut edges is minimized (min-cut)
  - Keep the weight of partitions within a constant factor of perfect balance
  - Decide replication of tuple and distributed updates or place it in a single partition and distributed transactions?

- **Use *METIS* to partition the graph**
  - Assign nodes to partitions

GRAPH PARTITIONING



tuple id	partition label
1	R
2	0
3	0
4	1
5	1

- **Fine-grained mapping between nodes and partitions**
  - Look-up table on attributes that frequently appears in WHERE clauses.

GRAPH PARTITIONING

- **Look-up tables:**
  - Stored in RAM?
  - Efficient maintanence for updates
  - Not ideal for large DB or insert-heavy workloads

- **Another phase of Schism: Explanation**
  - Predicate based partitioning

EXPLAINING THE PARTITION

- **Find a compact model/rules that represent the partitions**
- **Decision Trees**
  - Values: tuples
  - Labels: partitions
  - Replicated tuples are labeled by *replication identifier*

$$
\begin{aligned}
(id = 1) &\rightarrow \quad partitions = \{0, 1\} \\
(2 \leq id < 4) &\rightarrow \quad partition = 0 \\
(id \geq 4) &\rightarrow \quad partition = 1
\end{aligned}
$$

EXPLAINING THE PARTITION

- **The explanation is useful if;**
  - It is base on frequent attributes
  - It does not reduce the partitioning quality too much
  - It avoids over-fitting
    - prunning

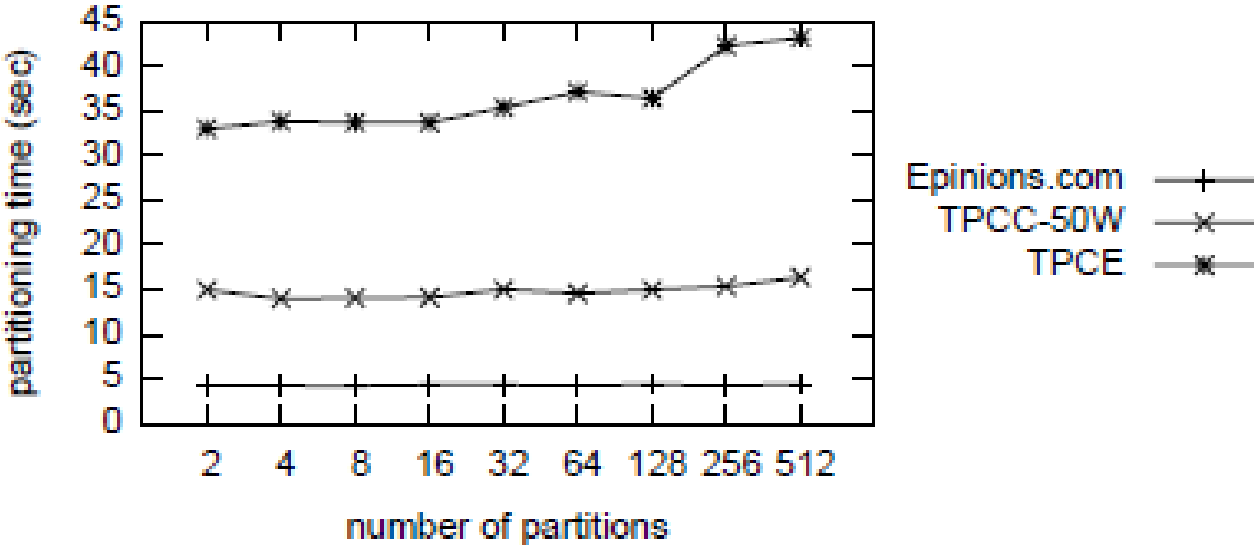
FINAL VALIDATION

- **Compare solutions**
- **Bestsolution:**
  - Provides the smallest number of distributed transactions.
    - Fine-grained per tuple partitioning
    - Range predicate partitioning
    - Hash partitioning
    - Full replication
  - Tie? Lowest complexity
    - Hash vs predicate?

OPTIMIZATIONS

- **Scalability:** Graph partitioning scale well in terms of the number of partitions, but running time increases substantially with graph size.

Dataset	Tuples	Transactions	Nodes	Edges
Epinions	2.5M	100k	0.6M	5M
TPCC-50	25.0M	100k	2.5M	65M
TPC-E	2.0M	100k	3.0M	100M



OPTIMIZATIONS

- **Reducing the size of graph with a limited impact on quality:**
  - Transaction-level sampling
    - Reducing #edges
  - Tuple-level sampling
    - Reducing #nodes
  - Tuple-coalescing
    - Represents tuples that are always accessed together

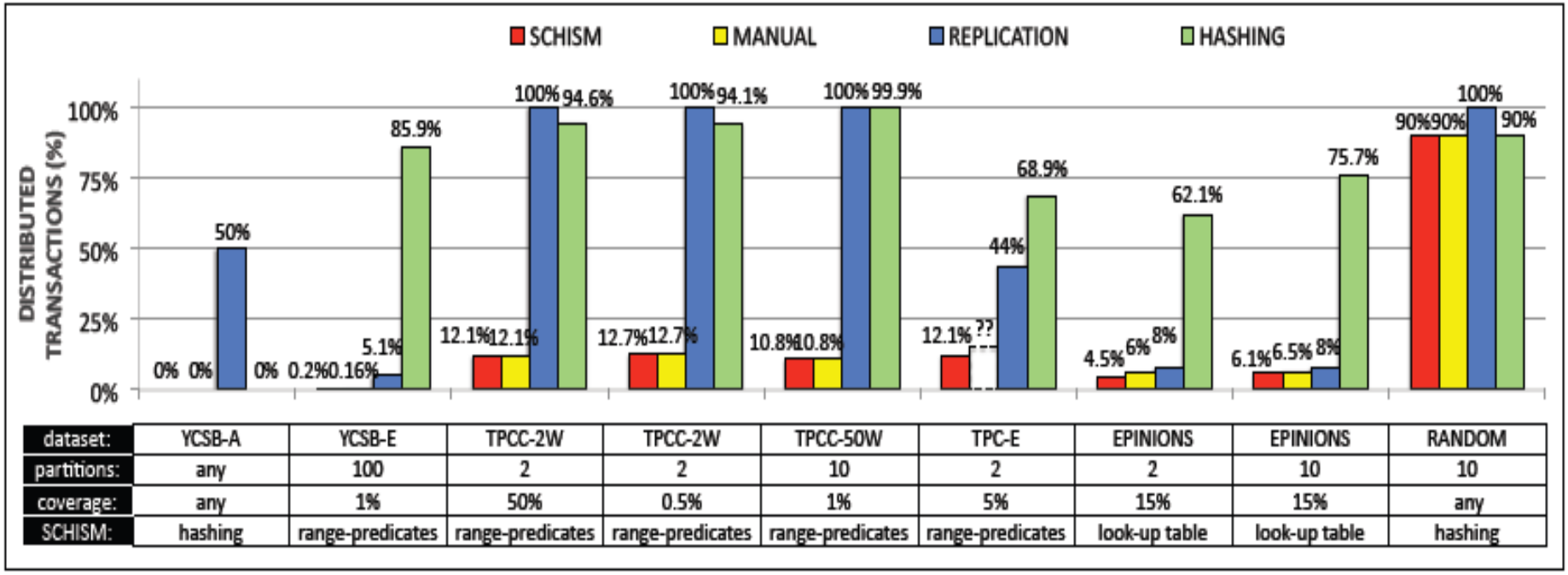
EXPERIMENTS

- **The experiments compare #transactions produced by;**
  - Schism
    - Fine-grained per tuple
    - Range predicates
  - Best manual partitioning
  - Replication of all tables
  - Hash partitioning
- **The fraction of the sampled dataset & #partitions**
- **Final validation**

EXPERIMENTS

- **Datasets:**
  - Yahoo Cloud Serving Benchmark
    - Workload A: reads – updates (%50-%50)
    - Workload E: short scan – one tuple update (%95-%5)
  - TPC-C: write intensive OLTP workload
    - Sampling, #partitions
    - 2W
    - 50W
  - TPC-E: read intensive OLTP workload
    - Complex (33 tables, 188 columns, 10 kinds of transactions)
  - Epinions.com:
    - Social website, n-to-n relations in the schema

EXPERIMENTS



dataset:	YCSB-A	YCSB-E	TPCC-2W	TPCC-2W	TPCC-50W	TPC-E	EPINIONS	EPINIONS	RANDOM
partitions:	any	100	2	2	10	2	2	10	10
coverage:	any	1%	50%	0.5%	1%	5%	15%	15%	any
SCHISM:	hashing	range-predicates	range-predicates	range-predicates	range-predicates	range-predicates	look-up table	look-up table	hashing

CONCLUSION

- **Schism;**
  - System for fine-grained partitioning of OLTP DB
  - Represents DB and transactions as a graph
  - Supports tuple-level replication
  - Uses classification techniques to represent partitions
  - Uses graph-partitioning algorithm
  - Proposes sampling to reduce graph size

DISCUSSIONS

- **Schism overcome the partitioning challenges**
  - Distributed transactions
  - Many-to-many relations

- **The quality of sampling & decision tree?**
  - Prunning?

- **What is the running time of Schism including all steps?**
  - Overhead of complexity - Choose the simplest.
  - Overhead of fine-grained partitioning?

DISCUSSIONS

- **The provided scalability is a result of METIS graph partitioning**

- **Schism focuses on using classification techniques to transform fine-grained partitioning into range partitions.**
  - How to use fine-grained partitioning

- **Hypergraph vs collection of edges**

DISCUSSIONS

- **Statements that access tuples using partitioning attributes are sent to those partitions**
- **Access table using other attributes?**
  - Broadcast the statement to all partitions
- **More complex statements: access multiple tables using non-partition attributes?**
  - Not currently handled.

THANK YOU