DISTRIBUTED RESEARCH ON EMERGING APPLICATIONS & MACHINES
Department of Computational & Data Sciences
Indian Institute of Science, Bangalore

DREAM:Lab

Granite: A Distributed Engine for Scalable Path Queries over Temporal Property Graphs

Shriram R.

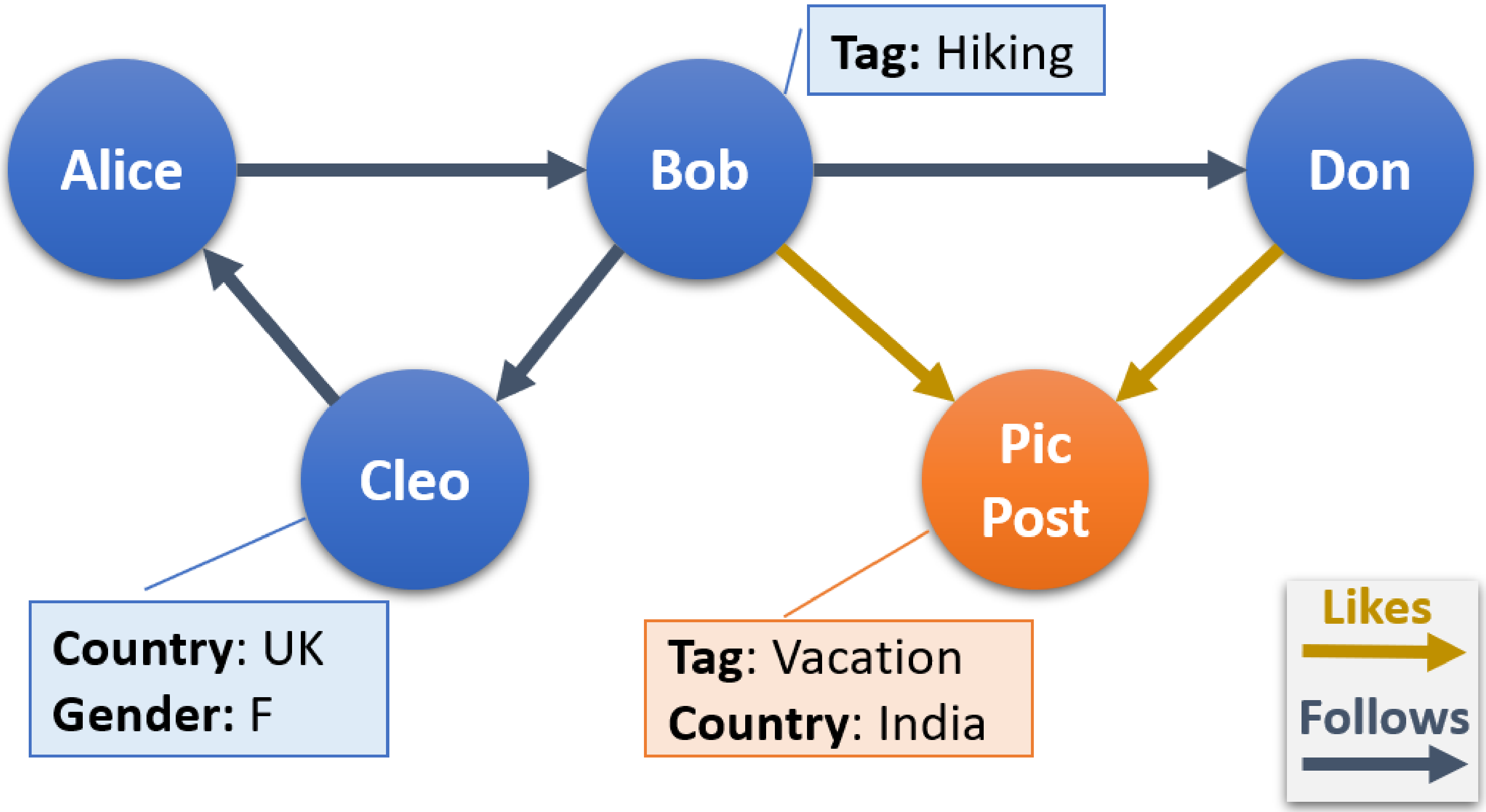
Advisor : Prof. Yogesh Simmhan

10th September 2020

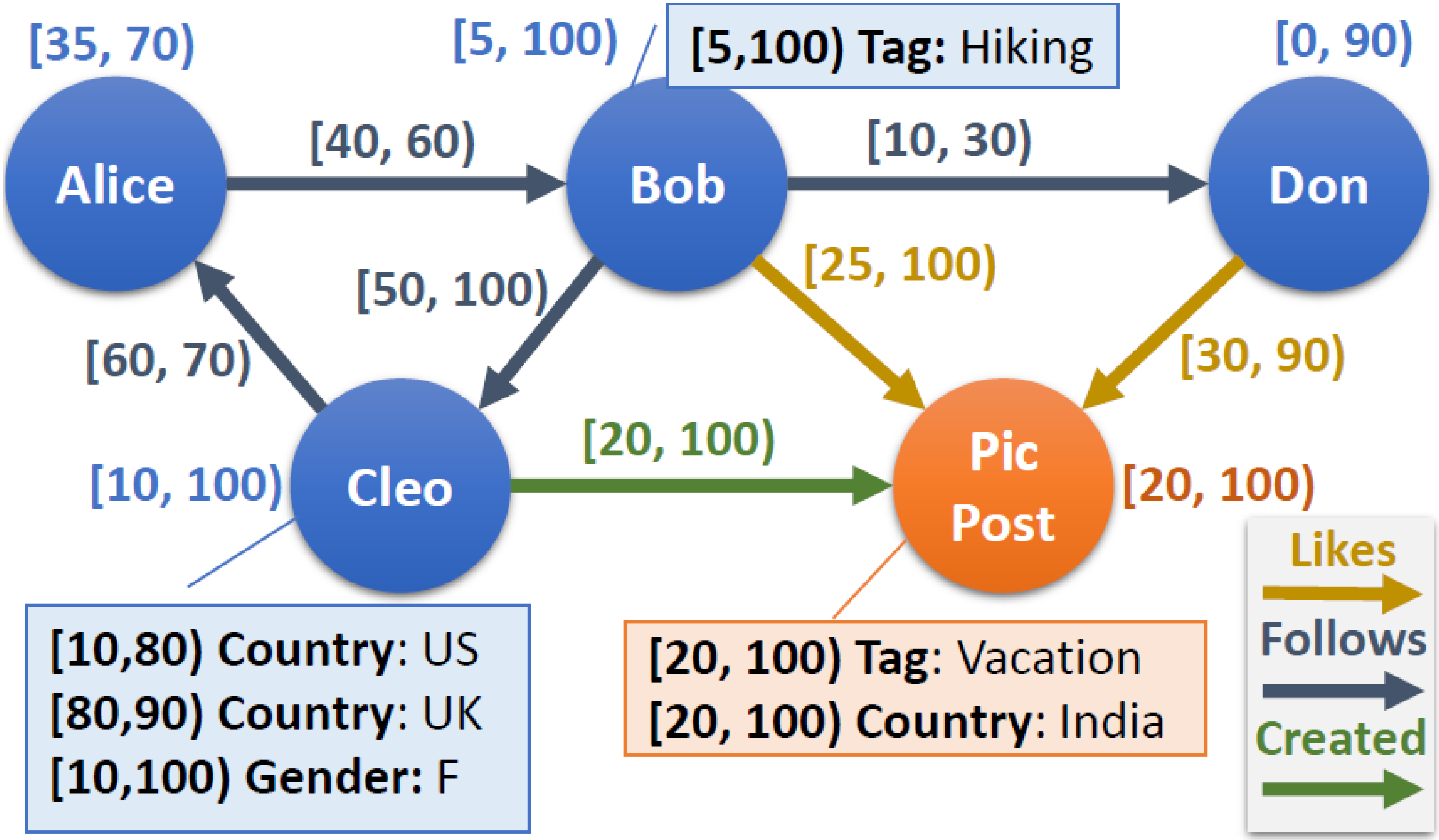
Department of Computational and Data Sciences

Introduction

Static Property Graph



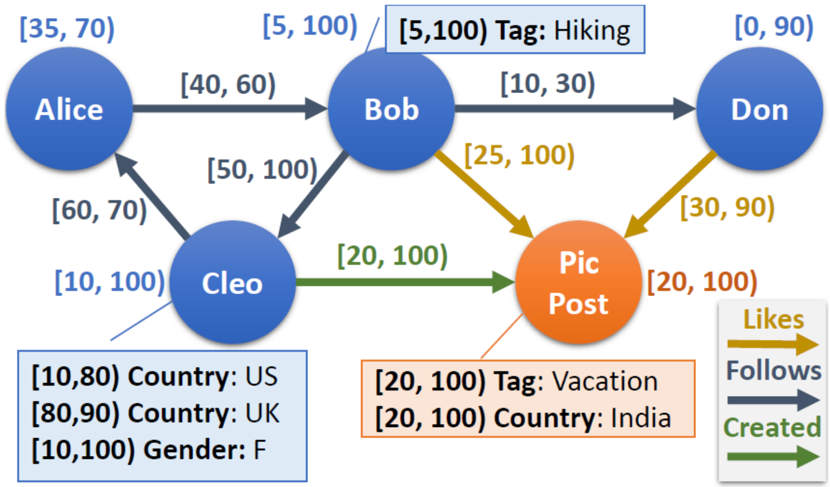
Temporal Property Graph



Query Example

- Find people tagged with 'Hiking' who liked a post tagged as 'Vacation', before the post was liked by a person named 'Don'.

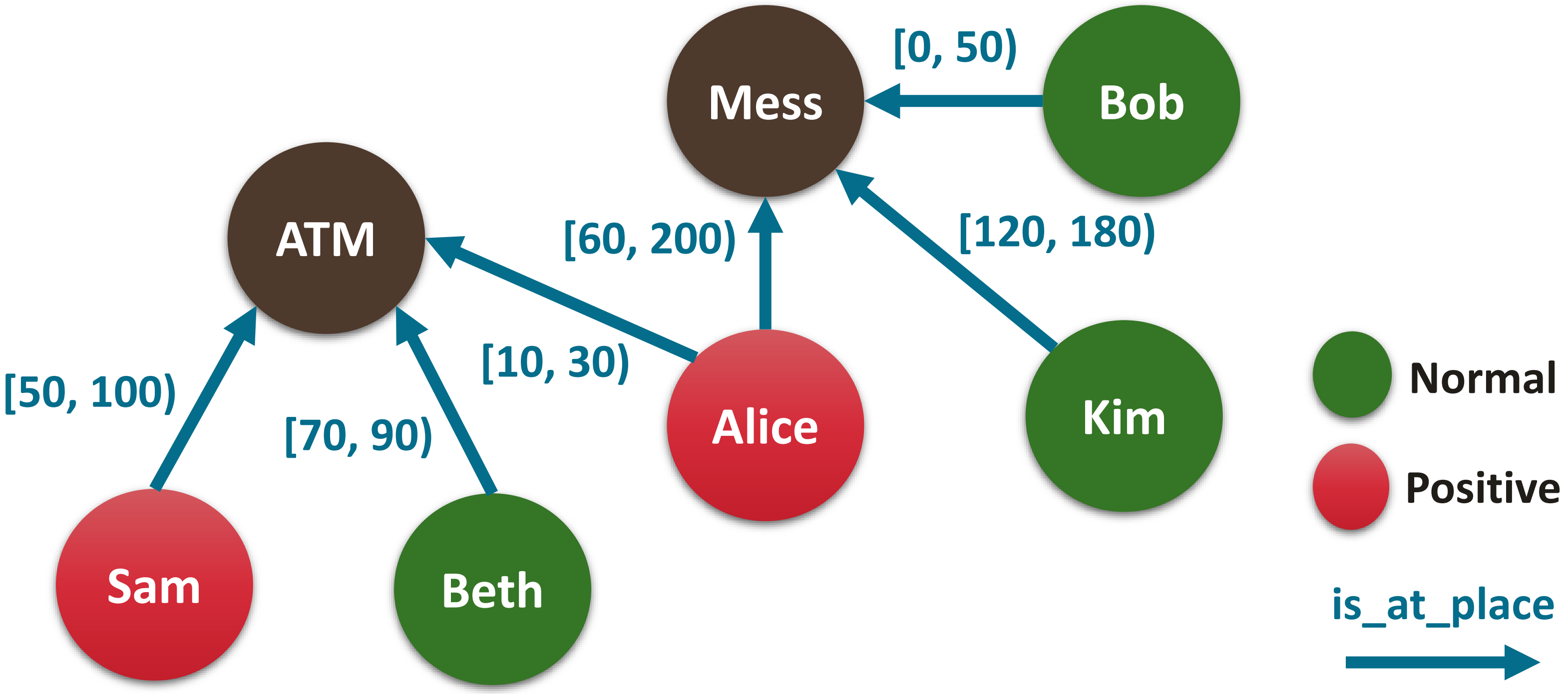
- Path (edges)
  - liked
- Properties
  - tagged with 'Hiking'
  - named 'Don'
- Temporal relations
  - before



(Bob->Pic Post<-Don)

Use Case: Querying the COVID-19 Contact Tracing Graph

For each person who is COVID positive, get the list of people who were near to them using place contacts.



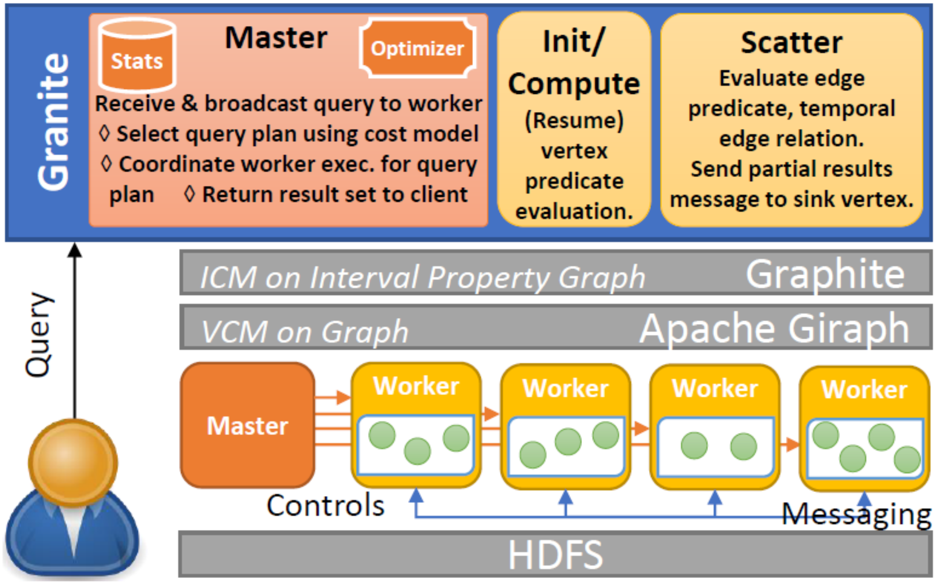
Our Contributions

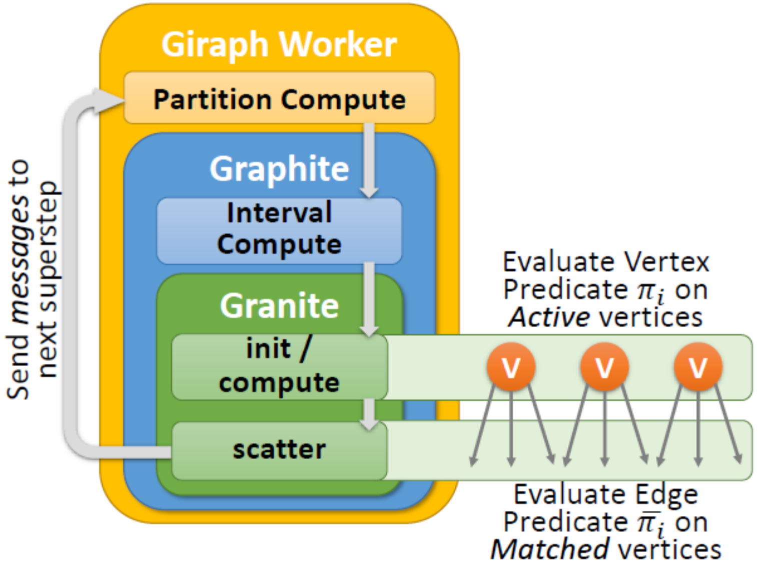
- Temporal property graph query model
- Distributed execution engine for our query model with optimizations
- Cost model to select the best execution plan using statistics
- Detailed evaluation of performance and scalability using LDBC workload
- Publications based on our work:
  - S. Ramesh, A. Baranawal, Y. Simmhan, A distributed path query engine for temporal property graphs, IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing **(CCGrid), 2020**.
  - S. Ramesh, A. Baranawal, Y. Simmhan, Granite: A distributed engine for scalable path queries over temporal property graphs, *under review* at **The Journal of Parallel and Distributed Computing (JPDC)**.

Query Engine

Distributed Execution Model

- Relaxed Interval Centric Model (ICM)*
- One ICM superstep per query hop.
  - Vertex Predicates – *compute*
  - Edge Predicates – *scatter*

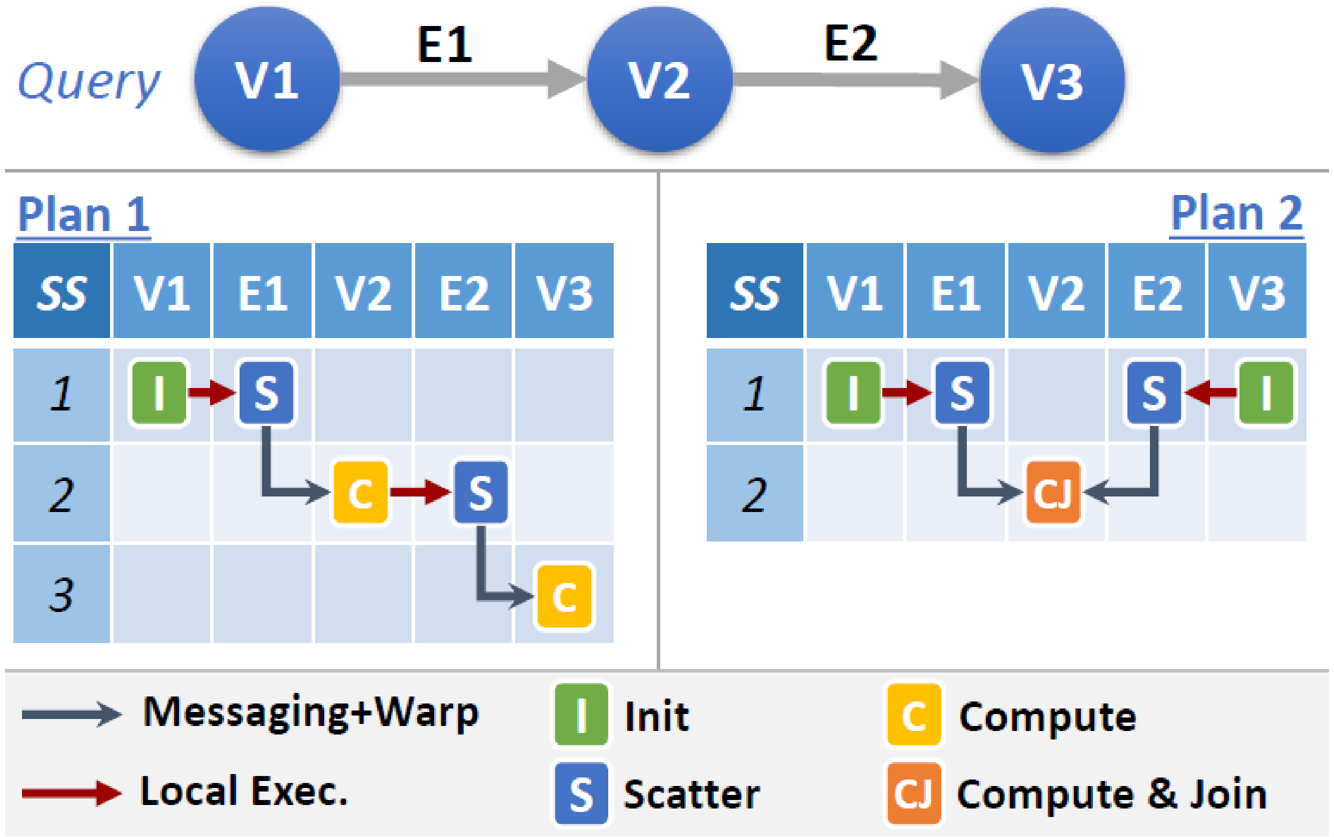




* S. Gandhi and Y. Simmhan, "An interval-centric model for distributed computing over temporal graphs," ICDE. IEEE, 2020.

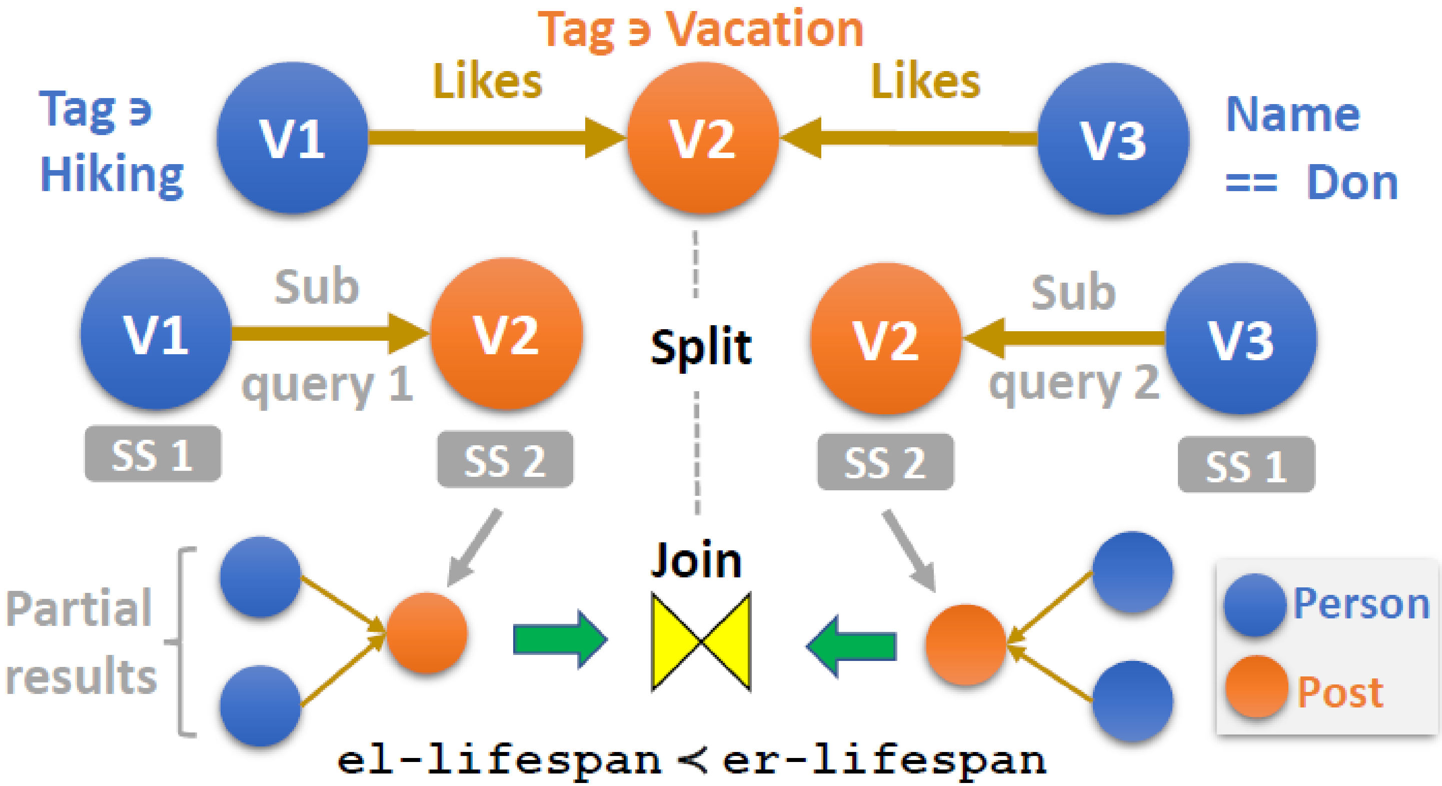
Query Execution Plans

- Superstep wise execution of different stages



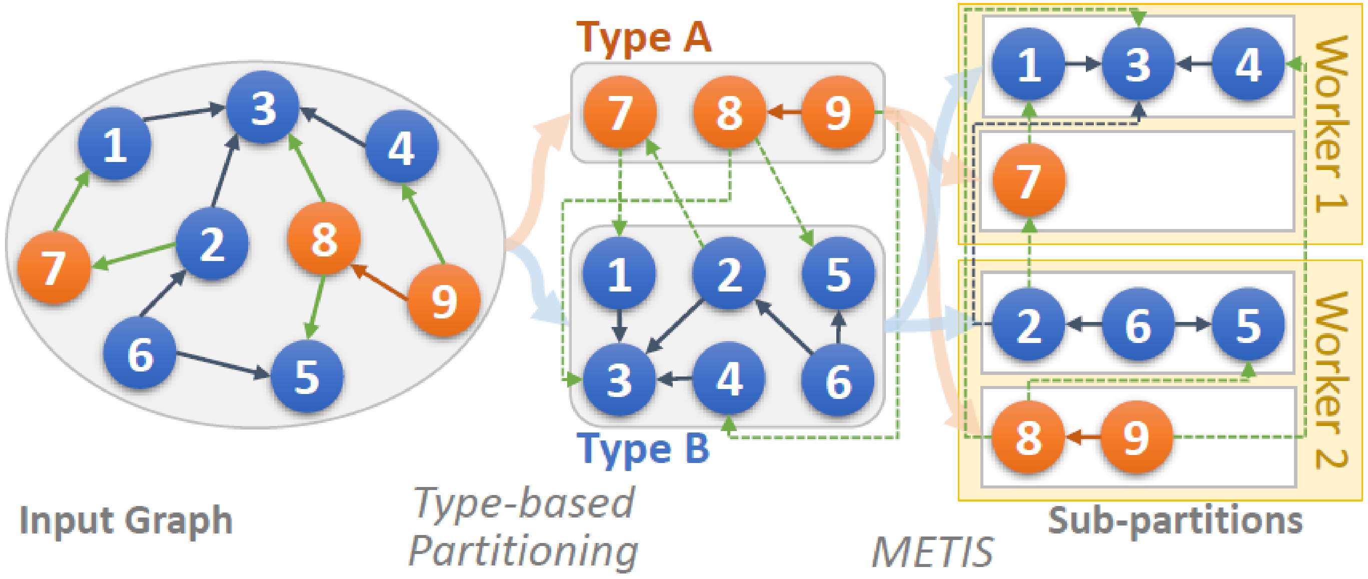
Query Execution Plans

- Illustration of split and join operation



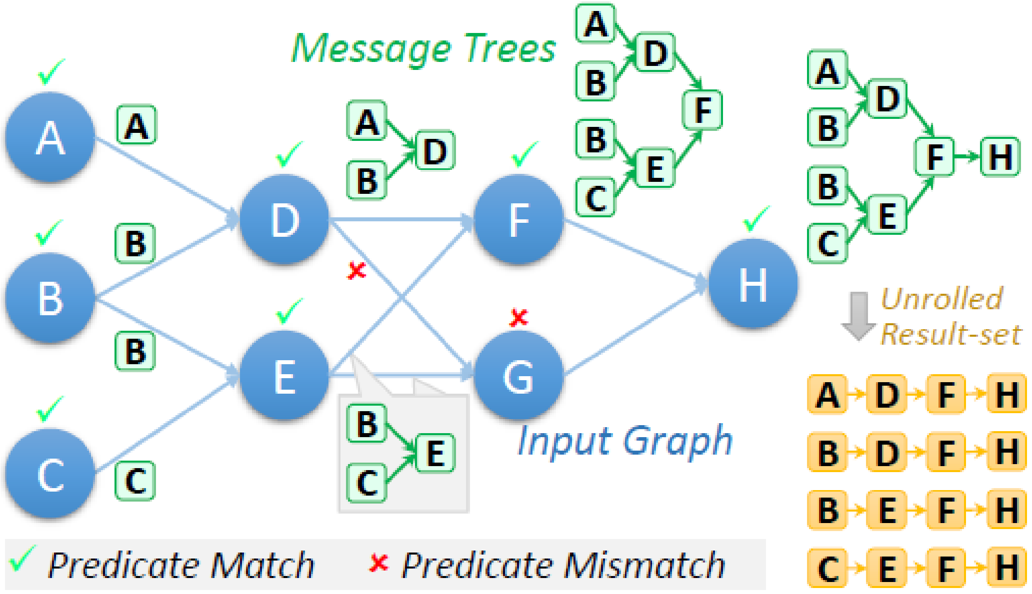
System Optimizations

- Type-based Graph Partitioning
  - 5.8x speedup due to type-based over hash based partitioning
  - 32% improvement from METIS



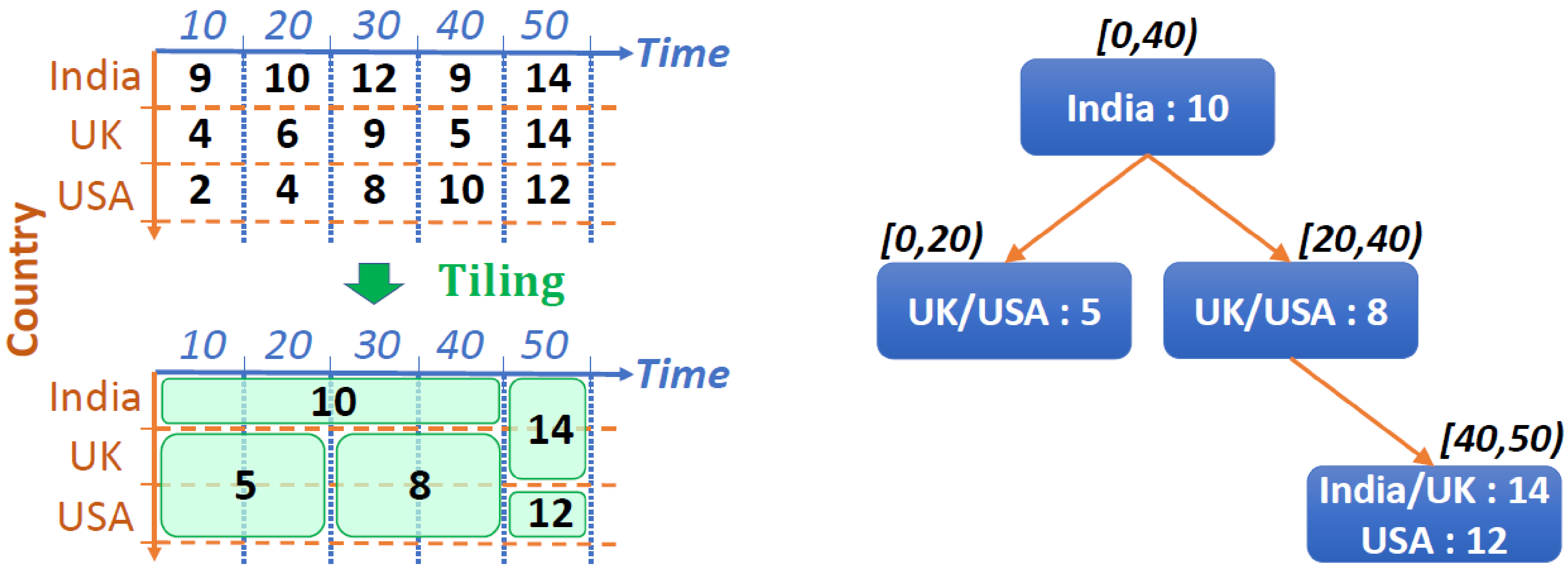
System Optimizations

- Message Trees
  - O(hxn) to O(2n-1) for a full binary tree



Query Optimization

Statistics - 2D Histograms



(a) 2-D Histogram of Statistics

(b) Interval Tree for Statistics

Cost Model - Illustration

Plan	SS	a_i	f_i	m_i	$\overline{a_i}$	$\overline{f_i}$	$\overline{m_i}$	T_i (ms)
1	1	$100k$	3.7×10^{-2}	$3.7k$	$\boxed{6.2M}$	$35M$	$1.3M$	$\boxed{531}$
	2	$1.3M$	7.7×10^{-4}	$1k$	–	–	–	132
2	1	$\boxed{51M}$	7.7×10^{-4}	$39k$	$273k$	$88M$	$67k$	$\boxed{4147}$
	2	$67k$	3.7×10^{-2}	$2.5k$	–	–	–	35

Init ($\mathcal{I}$)			Compute ($\mathcal{C}$)				Scatter ($\mathcal{S}$)		
a_0	m_0	cons.	a_i	m_i	$\overline{m_{i-1}}$	cons.	$\overline{a_i}$	$\overline{m_i}$	cons.
$\boxed{9.4e-5}$	-3.1e-5	3.83	7.2e-5	3.3e-5	1.8e-5	1.63	$\boxed{7.9e-5}$	0	-3.81

$$T = (\iota + s_1 + cc_1 + ic_1) + \sum_{i=2}^{n} c_i + s_i + cc_i + ic_i$$

Evaluation

Workload Generation

- LDBC Business Intelligence (BI) and Interactive Workload (IW)
- 100 query instances per query template

Query	LDBC ID	Hops	Prop. Preds.	Time Preds.	ER Pred.*
Q1	BI/Q9	3	4	1	Yes
Q2	BI/Q10	2	6	1	No
Q3	BI/Q16	3	6	1	Yes
Q4	BI/Q17	4	3	2	Yes
Q5	–	5	7	3	Yes
Q6	–	5	7	1	Yes
Q7	BI/Q23	4	5	3	Yes
Q8	IW/Q11	3	3	1	Yes

*The ldbc social network benchmark (version 0.3.2), Linked Data Benchmark Council, Tech. Rep., 2019.

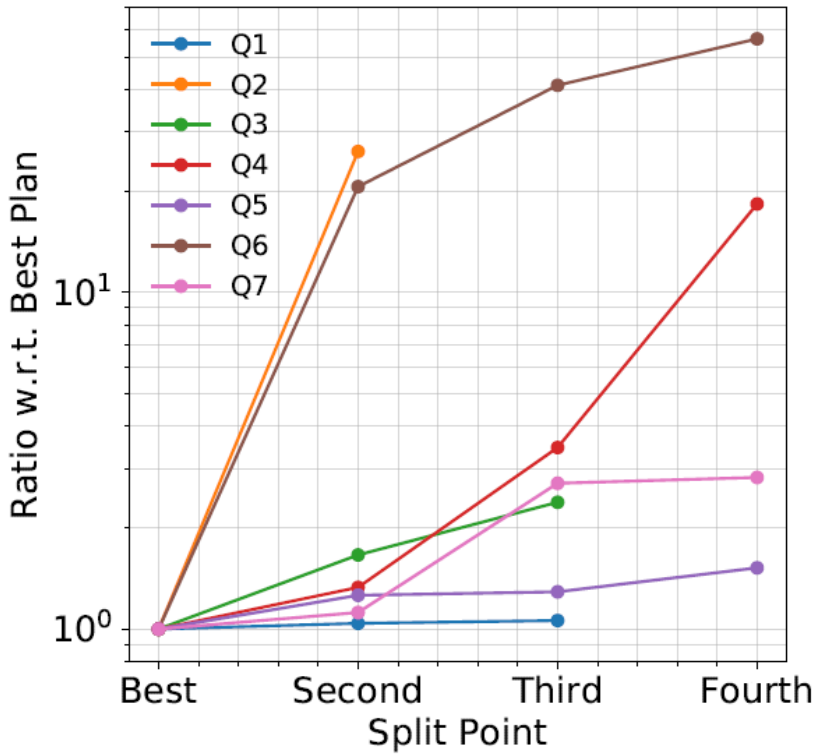
Graph/Workload Generation

- LDBC Benchmark Graph*
- 3-year Time Period

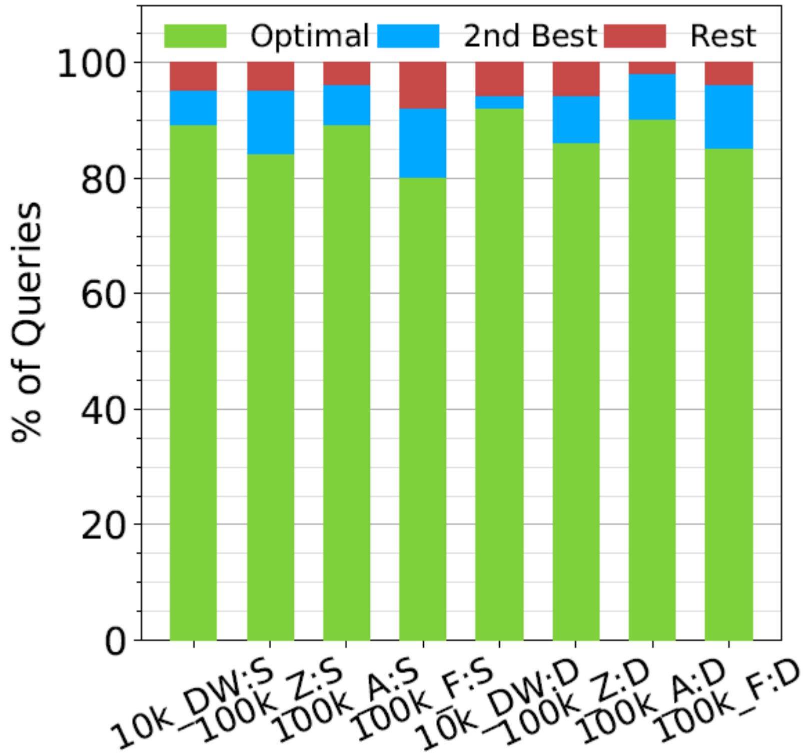
Graph	|V|	|E|	*Frequent Vertex Types* Persons	Posts	Comments	Forums
Static Temporal Graphs						
10k:DW-S	5.5M	20.8M	8.9k	1.1M	4.3M	82k
100k:Z-S	12.1M	23.9M	89.9k	7.4M	2.3M	815k
100k:A-S	25.4M	78.2M	89.9k	8.7M	15.7M	816k
100k:F-S	52.1M	217.6M	100k	12.6M	38.3M	996k
Dynamic Temporal Graphs						
10k:DW-D	6.6M	29.3M	10k	1.4M	5.1M	100k
100k:Z-D	15.2M	37.1M	100k	9.3M	4.8M	995k
100k:A-D	32.0M	112.2M	100k	10.8M	20.1M	995k
100k:F-D	52.0M	216.5M	100k	12.6M	38.2M	995k

*The ldbc social network benchmark (version 0.3.2), Linked Data Benchmark Council, Tech. Rep., 2019.

Cost Model Effectiveness

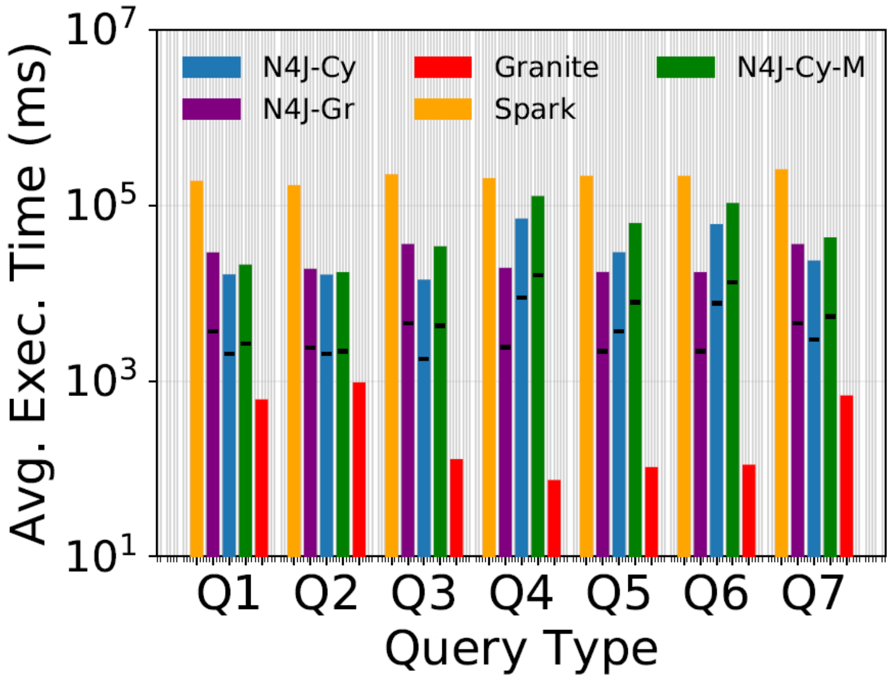


Ratio of estimated average execution cost of the other plans relative to the optimal plan, for all query types of 100k:A-S graph

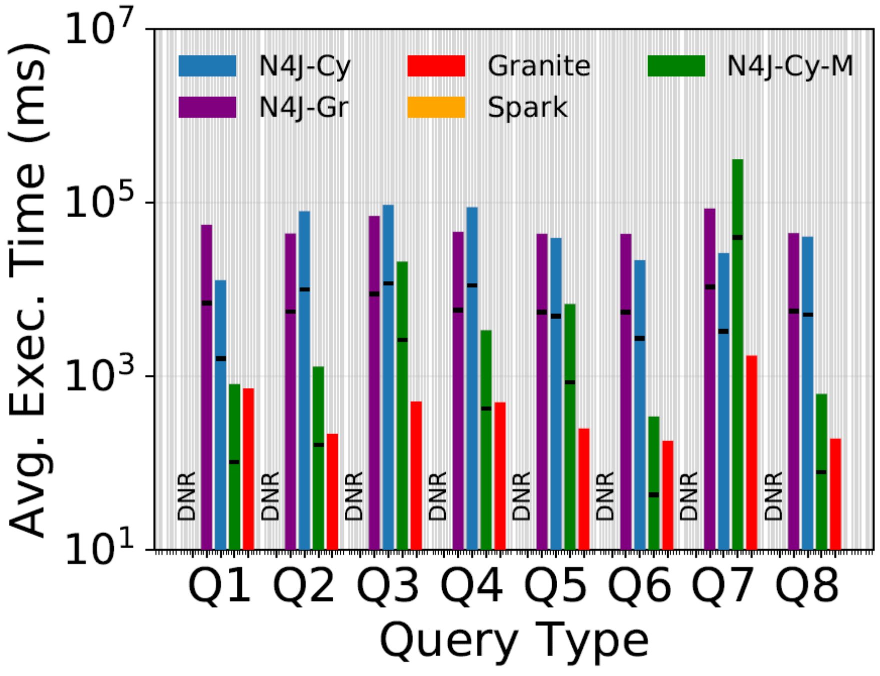


Cost Model Accuracy. % of times the optimal plan, 2nd best plan and other plans were selected by our model for all graphs

Comparison With Baselines



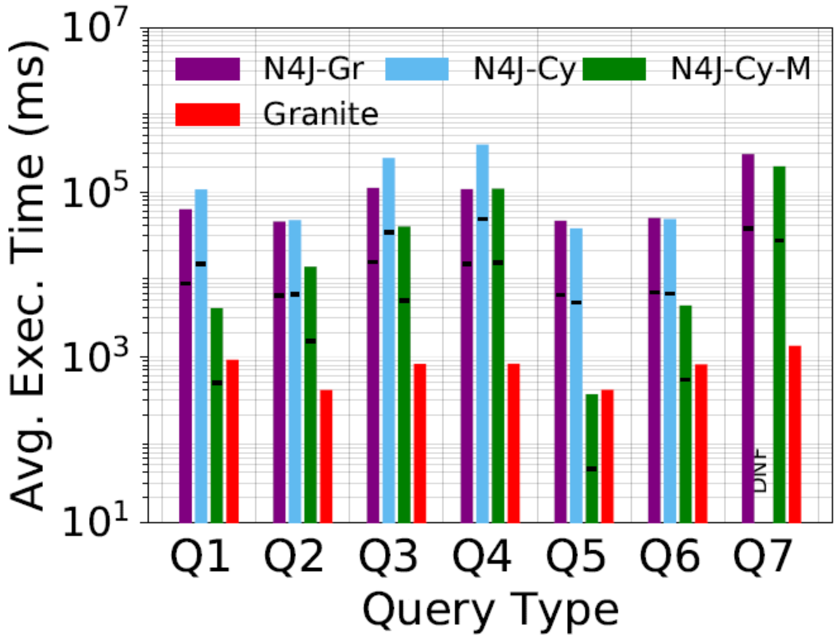
100k:Z-S



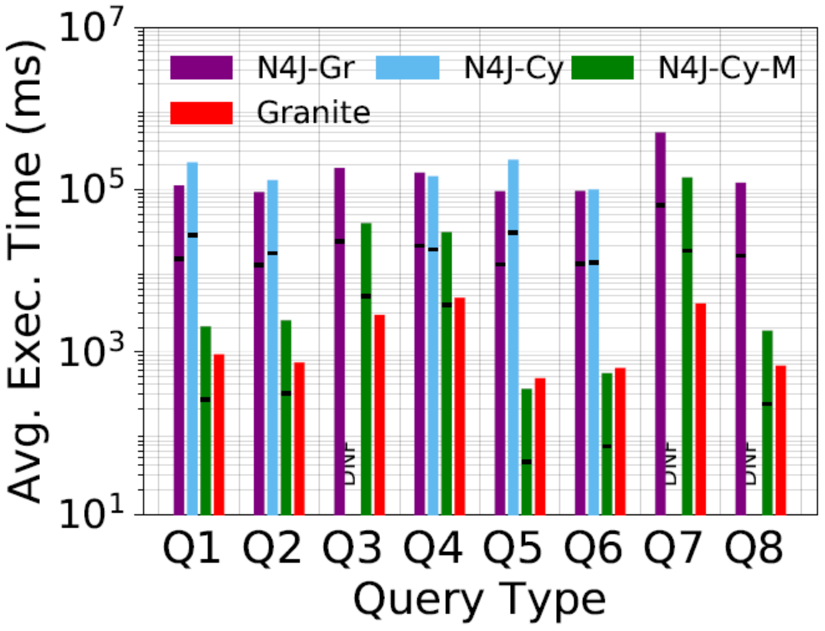
100k:Z-D

1. Spark – Distributed and N4J-Cy, N4J-Gr and N4J-Cy-M – Single Machine

Comparison with Baselines – Temporal Aggregate Queries



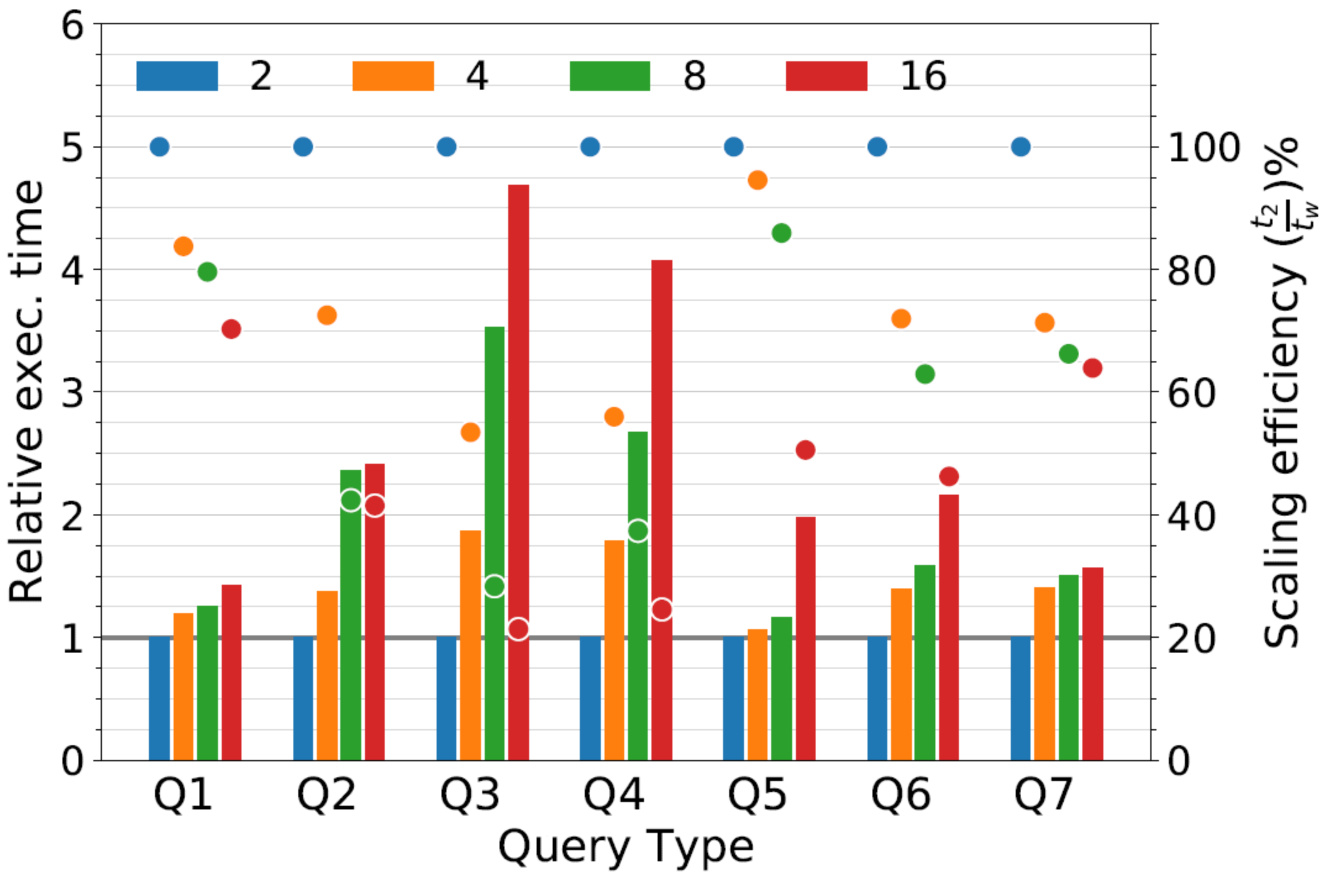
100k:A-S



100k:A-D

Comparison of average execution time of Granite with baseline systems for Temporal Aggregate query types.

Weak Scaling



Relative execution time (left axis, bar) and Scaling efficiency% (right axis, circle) for Worker counts w = 4; 8; 16, relative to w = 2 for Weak Scaling runs with (w x 6:25k):F-S graphs

The End