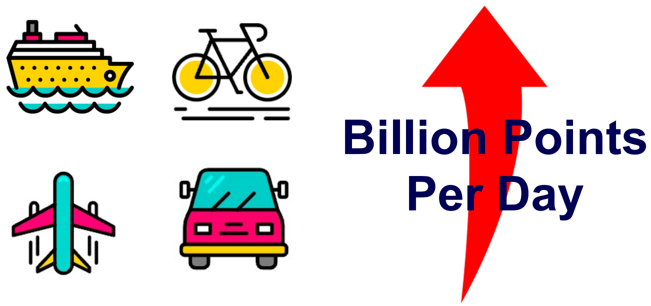
VRE: A Versatile, Robust, and Economical Trajectory Data System

Hai Lan, Jiong Xie, Zhifeng Bao, Feifei Li, Wei Tian, Fang Wang, Sheng Wang, Ailin Zhang

Outline

- Movtivation & Goals
- Existing Trajectory Systems
- VRE Architecture
- Storage Layer
- Query Processing
- Evaluation
- Conclusion

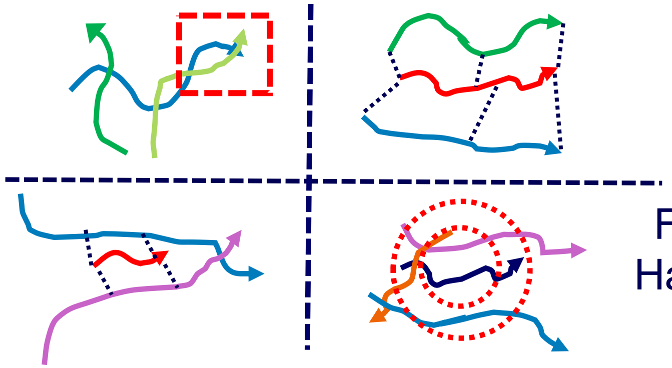
Motivation & Goals



Large-scale Trajectory Data

Properties	AIS (Vessels)	Porto (Taxi)
Avg. #Points	2,678.9	50.0
Avg. Spatial Span	(2.05, 1.49)	(0.03, 0.02)
Density of Traj.	2,310	410,326

Different Trajectory Properties



Various Queries on Trajectory Data

G1: Store the large-scale data economically

G2: Support all the typical queries and distance functions

G3: Be robust to trajectories with different properties, i.e., property-aware

Existing Trajectory Systems

Work	Basic Query			Advanced Query					Scalability		Data Properties		
	IDTQ	SRQ	STRQ	Tb-Search	Sub-Search	k-Search	Tb-Join	k-Join	Processing	Storage	NoP	SpS	DoT
Summit	✗	✓	✓	✗	✗	✗	✗	✗	✓	✓	-	-	-
MobilityDB	✗	✓	✓	✗	✗	✗	✗	✗	✓	✓	✓	✓	✗
TrajMesa	✓	✓	✓	F/H	✗	F/H	✗	✗	-	✓	✗	✗	✗
DFT	✗	✗	✗	F/H	✗	F/H	✗	✗	✓	✗	✗	✗	✗
DITA	✗	✓	✗	F/D/L/E	✗	F/D/L/E	F/D/L/E	✗	✓	✗	✗	✗	✓
REPOSE	✗	✗	✗	✗	✗	F/H/D	✗	✗	✓	✗	-	-	-
UlTraMan	✗	✓	✗	✗	✗	✗	✗	✗	✓	✗	-	-	-
VRE	✓	✓	✓	F/H/D/L/E	F/H/D/L/E	F/H/D/L/E	F/H/D/L/E	F/H/D/L/E	✓	✓	✓	✓	✓

G2 G1 G3

IDTQ: ID-Temporal Query
SRQ: Spatial Range Query
SRQ: Spatial-Temporal Range Query

Tb-Search: threshold-based similarity search, Tb-Join
Sub-Search: subtrajectory similarity search
k-Search: kNN search, k-Join

Frechet, Huarsdorff, DTW, LCSS, EDR

NoP: Number of Points
SpS: Spatial Span
DoT: Density of Trajectory

Architecture

Processing Layers

Schema Definition
Table Schema
Storage Model
Index Type/Mode

Data Operation
Insert update delete

Supported Queries

IDTQ SRQ STRQ

Tb-Search
k-Search
sub-Search
Tb-Join
k-Join

Two-stage Framework

Prune with meta
Prune with points

Parallel Processing
Process Pushdown

Supported Metrics

LCSS
EDR
Fréchet
Huasdorff
DTW

Pruning Framework

Storage Layers

Point/Segment/Trajectory-based Model

ID-Temporal Index, Spatial Range Index, Spatial-temporal range index, OTS index

Secondary Index Table

Key	Value
	Dup. Cols
v1+Seg_1	…
v2+Seg_2	…

Primary Index Table

Key	Value		
	Meta	Indexed Col1	Traj.
Seg_1	Obj_id,…	v1	Points
Seg_2	Obj_id,…	v2	Points

Engine Features - Lindorm

Distributed Storage Engine
Secondary Index
Compression
Cold-Hot Data Separation
Coprocessor

Storage Layer – Storage Layout

- Storage Model
  - Point / Segment / Trajectory
- Storage Layout

■ related to primary index

■ secondary index's key

Key	Value				
Rowkey	Metadata	Point List	Indexed Column	Indexed Column	Indexed Column
key_1(SR)	$meta_1$	$List_1$<Point>	key_1(IDT)	key_1(ST)	key_1(OTS)
key_2(SR)	$meta_2$	$List_2$<Point>	key_2(IDT)	key_2(ST)	key_2(OTS)
...	...	...	...	...	...
key_n(SR)	$meta_n$	$List_n$<Point>	key_n(IDT)	key_n(ST)	key_n(OTS)

Insert automatically

Key	Value
Rowkey	Duplicated Column
key_1(IDT) + key_1(SR)	...
key_2(IDT) + key_2(SR)	...
...	...
key_n(IDT) + key_n(SR)	...

■ metadata (used for pruning)
  ■ Object id, start time, end time, start point, end point
  ■ Segment's MBR, segment order, segment signature
  ■ Segment type

Storage Layer – Indexing

- Indexing
  - Support basic queries
  - Types: ID-Temporal Index, Spatial Range Index, Spatial-temporal range index, OTS index
  - Basic Idea
    - Insert: generate the indexed key: $key = \boldsymbol{F}(x)$
    - Query: generate the key range $[key_{lower}, key_{upper}]$ for a query, and then fetch the rows in the range

Query Processing – General Steps

Stage 1

1. Fetch the candidates' metadata only from the storage layer
2. Prune the unsatisfied results with metadata in query processing layer

A pruning framework based on metadata

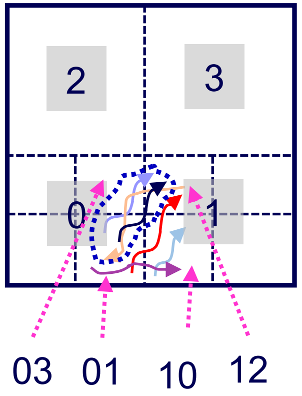
Pruning pushdown into storage layer

Stage 2

3. Fetch the full trajectories of the remaining candidates
4. Verify the candidates with the full trajectories

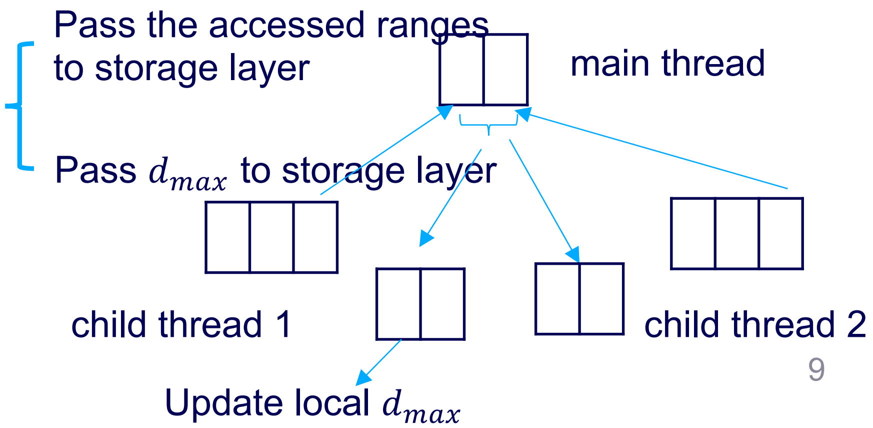
Query Processing – k-Search as an example

- k-Search
  - Given a query trajectory q, a distance function f, and an integer k, k-Search returns a subset K with size k (from $\mathcal{T}$), whose distances to q are less than the other trajectories in $\mathcal{T} - K$ to q.



1. Maintain a priority queue gq for these grids based on their distance to query q , $gq = <10,01,12,03,\ldots>$, a priority queue $tq = <>$ for results, $d_{max} = \infty$
2. Pop an item g from gq, if $|tq| = k$ and $f_{qg}(q, g) \geq d_{max}$, return tq.
3. Fetch the candidates C with a spatial range query with range g.
4. Partition C randomly and process each partition in parallel.
5. Goto step 2

- Sort the candidates based on their bounds first
- Two-stage processing
- Local bound synchronization



Query Processing – Pruning Framework

- Given a query $Q = \{q_1, q_2, \ldots, q_n\}$ with a distance threshold τ , after getting the candidates, we group these segments based by their tid. Each group is formed as $G = \{S_1, S_2, \ldots, S_{|G|}\}$, where $S_i = \{t_1, t_2, \ldots, t_{|S_i|}\}$ denotes a segment.

Metric	Completeness	LB_SES	LB_PartialSim	LB_Pivots	LB_SIG
Hausdorff	✓	0	$\max_{S_i \in G} \max_{q_j \in Q} \{f_{p\to p}(q_j, t_1), f_{p\to p}(q_j, t_{\|S_i\|})\}$	$\max_{q_j \in P} \min_{S_i \in G} \{f_{p\to r}(q_j, mbr_{S_i})\}$	$\max_{r_i \in sig_Q} \min_{r_j \in sig_G} \{f_{r\to r}(r_i, r_j)\}$[3]
Fréchet	✓	$\max\{f_{p\to p}(q_1, t_1),$[1] $f_{p\to p}(q_{\|Q\|}, t_{\|S_{\|G\|}\|})\}$	$\max_{S_i \in G} \max_{q_j \in Q} \{f_{p\to p}(q_j, t_1), f_{p\to p}(q_j, t_{\|S_i\|})\}$	$\max_{q_j \in P} \min_{S_i \in G} \{f_{p\to r}(q_j, mbr_{S_i})\}$	$\max_{r_i \in sig_Q} \min_{r_j \in sig_G} \{f_{r\to r}(r_i, r_j)\}$
DTW	✓	$\max\{f_{p\to p}(q_1, t_1), f_{p\to p}(q_{\|Q\|}, t_{\|S_{\|G\|}\|})\}$	$\sum_{S_i \in G} \|S_i\| \times \min_{q_j \in Q} \{f_{p\to r}(q_j, mbr_{S_i})\}$[2]	$\sum_{q_j \in P} \min_{S_i \in G} \{f_{p\to r}(q_j, mbr_{S_i})\}$	$\sum_{r_i \in sig_Q} \|r_i\| \times \min_{r_j \in sig_G} \{f_{r\to r}(r_i, r_j)\}$
EDR	✗	$\max\{f_{p\to p}(q_1, t_1), f_{p\to p}(q_{\|Q\|}, t_{\|S_{\|G\|}\|})\}$	$\sum_{S_i \in G} \|S_i\| \times \min_{q_j \in Q} \{f_{p\to r}(q_j, mbr_{S_i})\}$	$\sum_{q_j \in P} \min_{S_i \in G} \{f_{p\to r}(q_j, mbr_{S_i})\}$	$\sum_{r_i \in sig_Q} \|r_i\| \times \min_{r_j \in sig_G} \{f_{r\to r}(r_i, r_j)\}$
LCSS	✗	$\max\{f_{p\to p}(q_1, t_1), f_{p\to p}(q_{\|Q\|}, t_{\|S_{\|G\|}\|})\}$	-	-	-

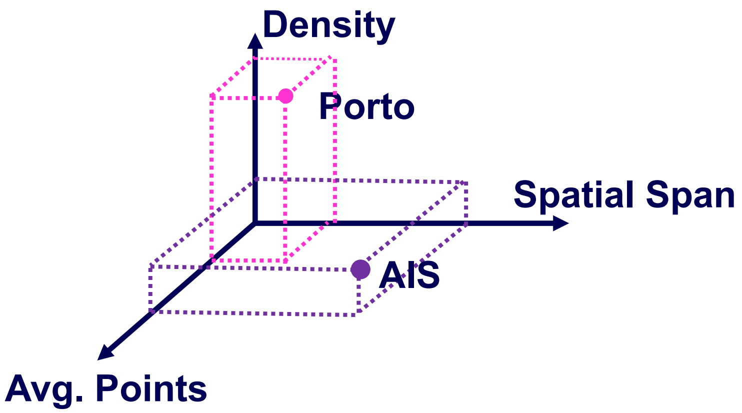
[1] $f_{p\to p}(p, q)$ denotes the Euclidean distance between a point p and a point q. In EDR and LCSS, it denotes the discrete distance defined by themselves.
[2] $f_{p\to r}(p, r) = \min_{p' \in r} f_{p\to p}(p, p')$ denotes the distance between a point p and a region r.
[3] $f_{r\to r}(r_1, r_2) = \min_{p_i \in r_1, p_j \in r_2} f_{p\to p}(p_i, p_j)$ denotes the region distance between r_1 and r_2.

- Completeness: whether a complete trajectory can be recovered from all the segments in G.
- LB_SES: the lower bound by considering the start and end segments.
- LB_PartialSim: lower bound based on the collected segments' metadata, i.e., partial segments.
- LB_Pivots: lower bound based on pivots from query Q.
- LB_SIG: lower bound based on their signatures.

Evaluation – Setup

- Dataset statistics

	AIS	Porto	Beijing	OSM
# Trajectories	42,446	1,645,908	11,114,613	96,648,669
Size (GB)	5.34	1.94	10.4	201
Avg # Points (NoP)	2,678.9	50.0	22.2	49.8
Avg Spatial Span (SpS)	(2.0508,1.4909)	(0.0322,0.0221)	(0.1,0.374)	(0.016,0.03)
Density of Trajs (DoT)	2,310	410,326	827,359	1,491



- Parameters

Parameter	Value
Time Window	12h, **1d**, 1w, 2w, 1m
Spatial Window	0.001, 0.002, **0.003**, 0.004, 0.005
Threshold τ	0.001, 0.002, **0.003**, 0.004, 0.005
k	1, 2, 5, **10**, 20
Data Size (%)	25, 50, **100**, 200, 400
# of Cores	1, 2, 4, 8, 16, **32**

Evaluation – Different Storage Schemas on AIS

- Storage & Bulkload(AIS)
  - With secondary index, storage cost is reduced significantly.
  - Metadata only takes 4% of total storage cost.
  - Insertion time is proportional to the storage size.
- Query Performance (AIS)
  - One schema cannot be best in all cases!
    - Related to query types & result types
    - For example, trajectory-based is more suitable for Tb-Search than segment-based model while k-Search has a different result.

Evaluation – Proposed Optimizations

- Impact of Secondary Index

	IDTQ (Porto)	*STRQ* (Porto)	*IDTQ* (AIS)	*STRQ* (AIS)
PK (ms)	33.16	91.32	11.75	6.91
SK (ms)	33.56	218	9.75	7.32

  - The candidate size of $STRQ$ on Porto is much larger. Random access leads to a high latency.
  - Add metadata into secondary index or answer $STRQ$ with SR index.

- Efficiency of Two-Stage Framework

	SRQ (Porto)	*Tb-Search* (Porto)	*SRQ* (AIS)	*Tb-Search* (AIS)
one-stage (s)	14.94	out-of-memory	2.24	2.64
two-stage (s)	6.37	1.86	0.47	0.22

  - Two-stage has smaller latency with much fewer data transferred from storage layer

- Impact of Pushdown (Tb-search as example)

dataset	total (s)	rs	cs	tc (s)	tp (s)	tf (s)	td (s)
Porto	0.89	23	474	0.83	0.0	0.0	0.0
Porto (w/o pushdown)	2.17	23	1,105,945	1.82	0.0	0.0	0.0
AIS	0.1	3	3	0.09	0.0	0.0	0.0
AIS (w/o pushdown)	0.06	3	491	0.05	0.0	0.0	0.0

  - With pushdown strategies, we can reduce the candidate size to be verified in processing layer.
  - On AIS, pushdown is not a good choice.
    - Overhead in evoking coprocessor and pruning

Evaluation – k-Search

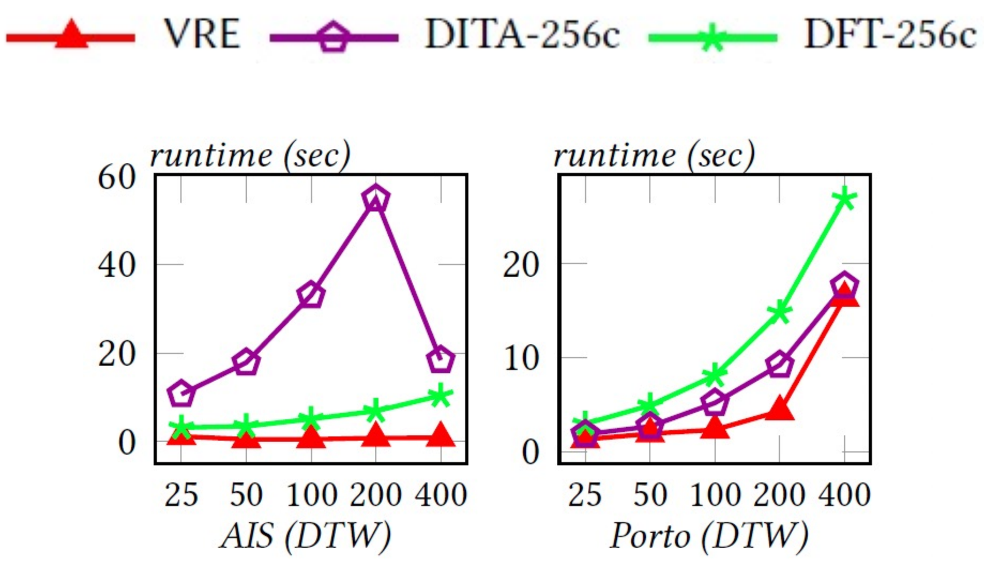
Runtime (s) of *k-Search*

Distance Function	System	*AIS*					*Porto*					*OSM*				
		1	2	5	10	20	1	2	5	10	20	1	2	5	10	20
Hausdorff	DFT	6.68	2.86	3.42	3.81	3.83	16.32	15.16	15.55	16.65	15.80	-	-	-	-	-
	VRE	**0.89**	**1.42**	**2.09**	**2.76**	**3.42**	**1.75**	**1.55**	**1.62**	**1.67**	**1.77**	**0.10**	**0.13**	**0.15**	**0.27**	**0.55**
Fréchet	DFT	2.02	2.36	2.81	2.89	**2.94**	11.04	12.87	11.60	12.06	12.08	-	-	-	-	-
	DITA	2.34	3.02	4.12	6.84	11.26	2.69	3.17	2.82	2.99	3.27	-	-	-	-	-
	VRE	**0.52**	**0.73**	**1.30**	**2.01**	3.31	**1.40**	**1.29**	**1.31**	**1.32**	**1.34**	**0.09**	**0.14**	**0.16**	**0.29**	**0.60**
DTW	DFT	3.84	4.20	5.02	5.05	5.72	7.68	7.57	7.68	7.62	7.71	-	-	-	-	-
	DITA	3.61	32.45	32.84	33.01	38.67	4.98	4.93	4.72	5.18	5.70	-	-	-	-	-
	VRE	**1.15**	**1.27**	**1.81**	**2.86**	**4.02**	**2.07**	**2.07**	**2.17**	**2.28**	**2.52**	**0.10**	**0.13**	**0.16**	**0.22**	**0.50**
EDR	DITA	**3.17**	**4.88**	**4.98**	**5.47**	**6.29**	14.75	15.99	15.56	16.43	16.46	-	-	-	-	-
	VRE	4.36	5.50	6.04	8.27	9.28	**1.86**	**3.82**	**4.15**	**4.47**	**4.77**	**0.16**	**0.18**	**0.19**	**0.22**	**0.25**
LCSS	VRE	**1.09**	**1.46**	**1.49**	**1.57**	**1.58**	**7.42**	**7.30**	**7.34**	**7.35**	**7.35**	**0.28**	**0.27**	**0.27**	**0.27**	**0.29**

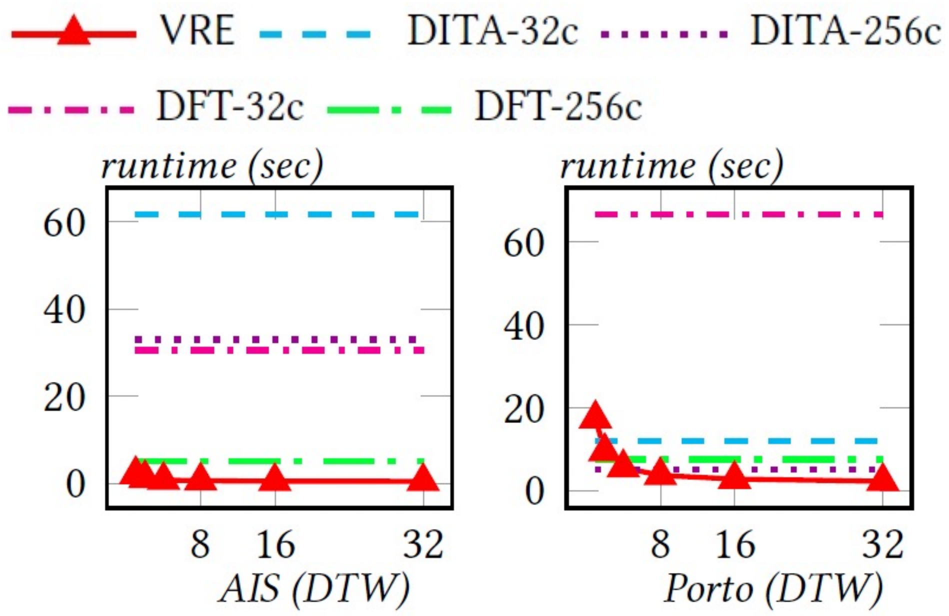
-: DFT or DITA crashed since it consumes too much memory on big datasets.

– VRE beats other systems or is competative in all cases.

Evaluation – k-Search



Different dataset sizes



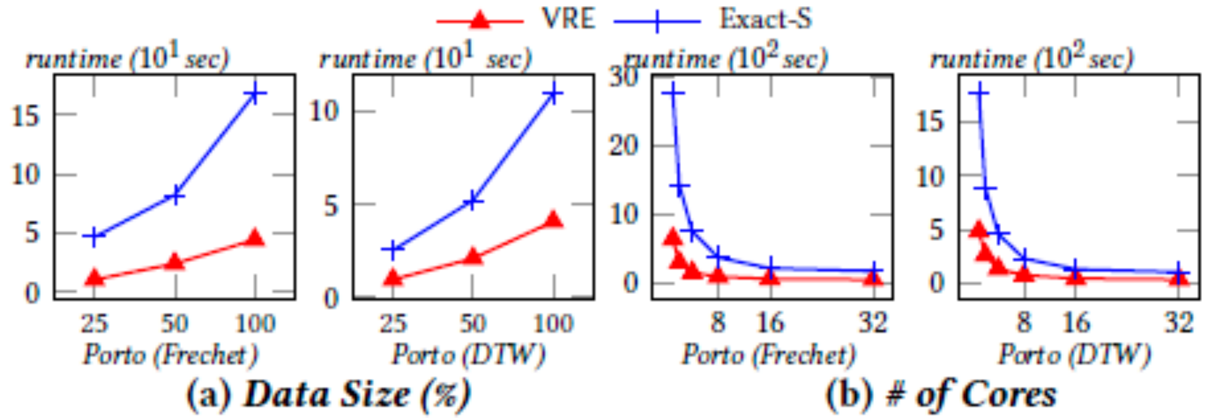
Different number of cores

- VRE has good scalability on dataset size.

- VRE is better or competative to DITA and DFT with fewer cores.

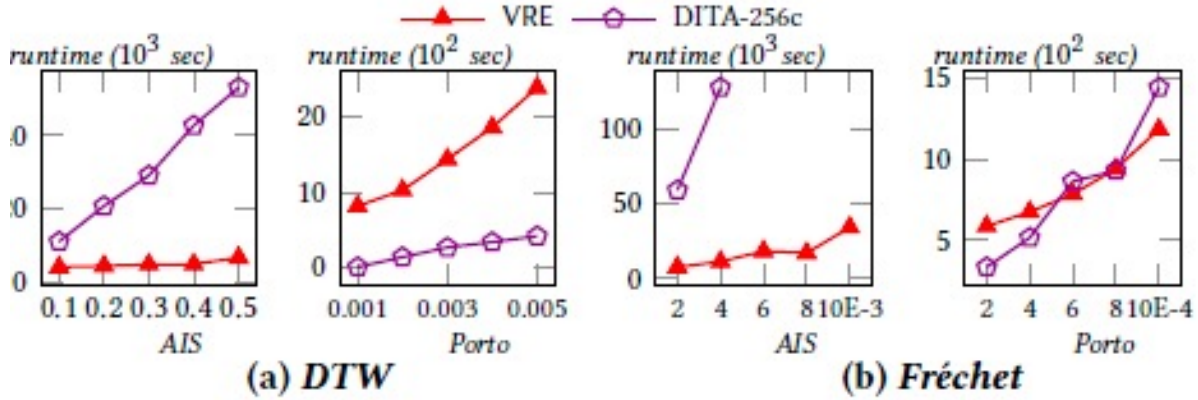
Evaluation – Sub-Search and Tb-Join

- Sub-Search



  - VRE achieves a better scalability in terms of data size.
  - VRE has better performance under the same core number setting.

- Tb-Join



  - VRE outperforms DITA on AIS in both DTW and Fréchet is competitive to DITA on Porto in Fréchet.
  - DITA beats VRE on Porto in DTW.
    - 32 cores vs. 256 cores

More…

- Algorithms for all advanced queries
- Results for basic queries, Tb-Search, and other joins
- Insights for VRE

Conclusion & Future Work

- Conclusion
  - First system that supports all typical basic and advanced query types and distance functions
  - Deployed as part of Lindorm Ganos in Alibaba
- Future Work
  - Select the right execution plan with a cost model
  - Able to handle the queries across multiple nodes
  - Benchmark for trajectory processing

Q&A
Thanks!

Storage Layer – Different Schemas

whether use secondary index

metadata type

storage model

Name	Schema	System
VRE_{PXM}	$key_p + meta_p + point$	GeoMesa, VRE
VRE_{TXM}	$key_t + meta_t + traj$	TrajMesa, VRE
VRE_{TXM_1}	$key_t + meta_{\{s\}} + traj$	VRE
VRE_{SXM}	$key_s + meta_s + seg$	VRE
VRE_{TXM_2}	$key_t + traj; key_s + meta_s$	VRE
VRE_{*S*}	See above	VRE

Evaluation – Different Storage Schemas

- Storage Size (AIS)

Storage Cost (MB) and Insertion Time (s)

	Disk	TSM	SSM	SXM	SSX	TSM_1	TSM_2	TXM	PXX
Cost	795	1167	1508.9	5535	1440.9	1291.6	1566.1	3455	7362
Ratio	1.0	1.5	1.9	6.7	1.8	1.62	2.0	4.34	9.6
Insert	-	83.8	99.1	214	94.7	81.8	112.4	183.8	624.5

  - With secondary index, storage cost redcues significantly.
  - Metadata only takes 4% of total storage cost.
  - Inserttion time is proportional to the storage size.

- Query Performance (AIS)

Table 11: Time Breakdown of *Tb-Search* (Storage Schemas)

schema	total (ms)	tc (ms)	tp (ms)	tf (ms)	td (ms)
VRE_{TSM}	202	86	7	5	98
VRE_{TSM_1}	233	122	7	3	95
VRE_{SSM}	397	134	152	10	96
VRE_{TSM_2}	390	130	166	5	95

Time Breakdown of *k-Search* (Storage Schemas)

schema	iter	total (ms)	tt_1	tc_1	tp_1	tf_1	td_1	avg	post-pro
VRE_{TSM}	2	7726.0	7482	107	91	901	6472	157	0
VRE_{TSM_1}	2	4362	3990	127	269	483	2673	294	0
VRE_{SSM}	2	4537	3637	297	523	241	2463	759	88
VRE_{TSM_2}	2	4459	3574	305	197	227	2504	753	76

  - One schema cannot be best in all cases!
    - Related to query type

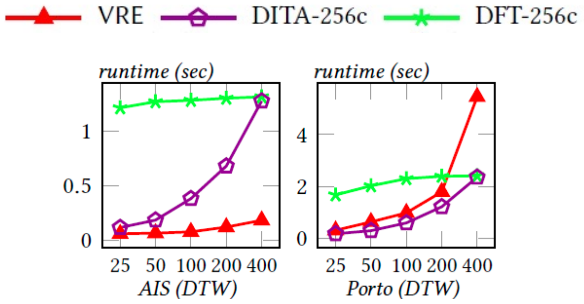
Evaluation – Tb-Search

Runtime (s) of *Tb-Search*

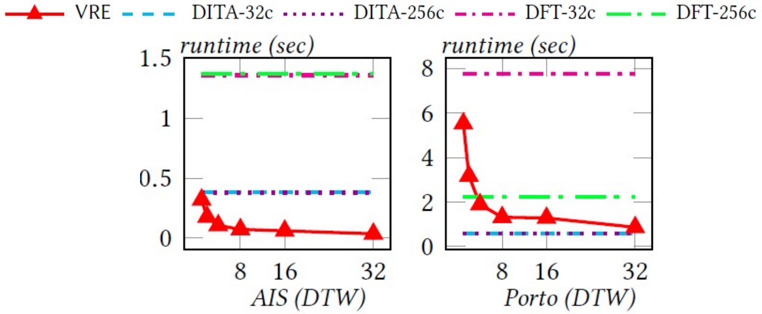
Distance Function	System	AIS 0.1	AIS 0.2	AIS 0.3	AIS 0.4	AIS 0.5	Porto 0.001	Porto 0.002	Porto 0.003	Porto 0.004	Porto 0.005	OSM 0.01	OSM 0.02	OSM 0.03	OSM 0.04	OSM 0.05
Hausdorff	DFT	1.37	1.40	1.38	1.43	1.54	2.48	2.39	2.37	2.66	3.31	-	-	-	-	-
	VRE	**0.19**	**0.25**	**0.38**	**0.55**	**0.64**	**0.73**	**0.76**	**0.80**	**0.87**	**0.96**	**0.11**	**0.12**	**0.15**	**0.18**	**0.21**
Fréchet	DFT	1.32	1.37	1.40	1.36	2.38	2.43	2.34	2.28	2.45	2.96	-	-	-	-	-
	DITA	0.60	0.82	0.98	1.11	1.26	**0.63**	**0.62**	**0.68**	**0.72**	**0.81**	44.69	45.85	46.64	46.60	46.91
	VRE	**0.24**	**0.26**	**0.41**	**0.56**	**0.67**	0.72	0.75	0.76	0.80	0.81	**0.09**	**0.11**	**0.15**	**0.17**	**0.19**
DTW	DFT	1.29	1.39	1.38	1.33	2.09	2.34	2.29	2.23	2.37	2.90	-	-	-	-	-
	DITA	0.35	0.36	0.38	0.37	0.43	**0.57**	**0.57**	**0.58**	**0.59**	**0.63**	49.40	42.71	43.74	43.78	43.10
	VRE	**0.20**	**0.17**	**0.17**	**0.18**	**0.18**	0.72	0.75	0.77	0.79	0.80	**0.10**	**0.09**	**0.09**	**0.09**	**0.01**
EDR		10	20	40	80	200	1	2	3	4	5	1	2	3	4	5
	DITA	1.76	1.71	1.77	1.80	1.96	**0.66**	**0.71**	**1.20**	**1.28**	**1.20**	131.72	390.00	-	-	-
	VRE	**0.31**	**0.18**	**0.18**	**0.17**	**0.18**	3.39	3.26	3.30	3.34	3.34	**0.09**	**0.08**	**0.08**	**0.08**	**0.08**
LCSS		0.80	0.85	0.90	0.95	1.0	0.80	0.85	0.90	0.95	1.0	0.80	0.85	0.90	0.95	1.0
	VRE	**0.23**	**0.19**	**0.18**	**0.17**	**0.07**	**4.86**	**4.50**	**4.43**	**4.41**	**3.34**	**0.11**	**0.09**	**0.09**	**0.09**	**0.09**

-: DFT or DITA crashed since it consumes too much memory on big datasets.

- Except EDR on Porto, VRE beats other systems or is competative.



Different dataset sizes



Different number of cores

- VRE has good scalability on dataset size.
- VRE is better or competative to DITA and DFT with fewer cores.

VLDB 2022