Indexing the Pickup and Drop-off Locations of NYC Taxi Trips in PostgreSQL – Lessons from the Road

Jia Yu and Mohamed Sarwat

Arizona State University

Arizona State University

A Little Story…

- **August 1, 2015**: Over **1 billion** taxi trip records from 2009 to 2015 were released by New York City Taxi & Limousine Commission
- **Since then:** New taxi trip records **keep being published** on the Internet
- **As of TODAY: Millions** of new records have been added into the dataset

PickupTime	DropoffTime	TripDistance	PickupLocation	DropoffLocation	PaymentType	FareAmount	TipAmount
2009-01-01 08:01:01	2009-01-01 08:20:37	2.2	(40.7577,-73.9851)	(40.7497,-73.9882)	Credit Card	15.5	3.5

Photo credit: NYC TLC website

A Little Story… (cont.)

- People really want to do Spatial Query on this **175 GB** data in PostGIS

- People really need a Spatial Index to speed up the queries.

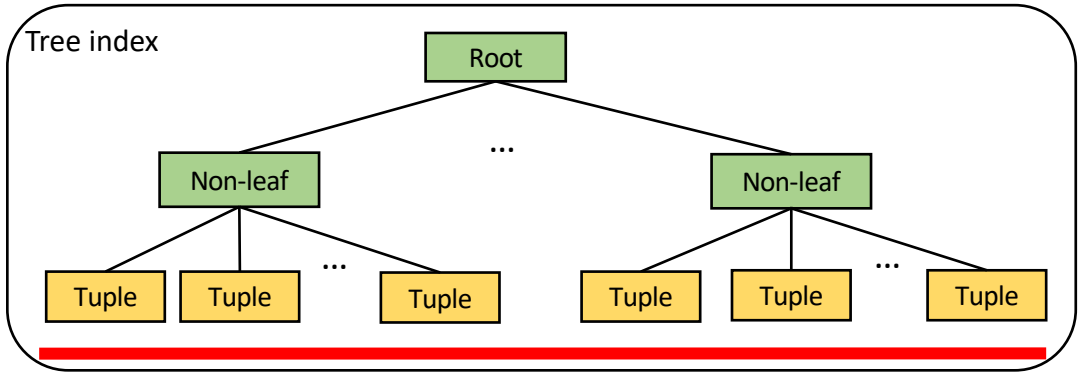
Which Spatial Index can handle these?

1 billion records, 175 GB, millions new records, keep being published

Arizona State University

Compared Approaches: GiST

- Generalized Search Tree (GiST-Spatial, Similar to R-Tree)
  - Index structure:



  - Index entry (tree node): Minimum Bounding Rectangle, Tuple pointers
  - Index search: Top-down, fast prune by checking Query Window with MBR
  - Index maintenance: Search tree, then split (if full) and merge (if too empty)

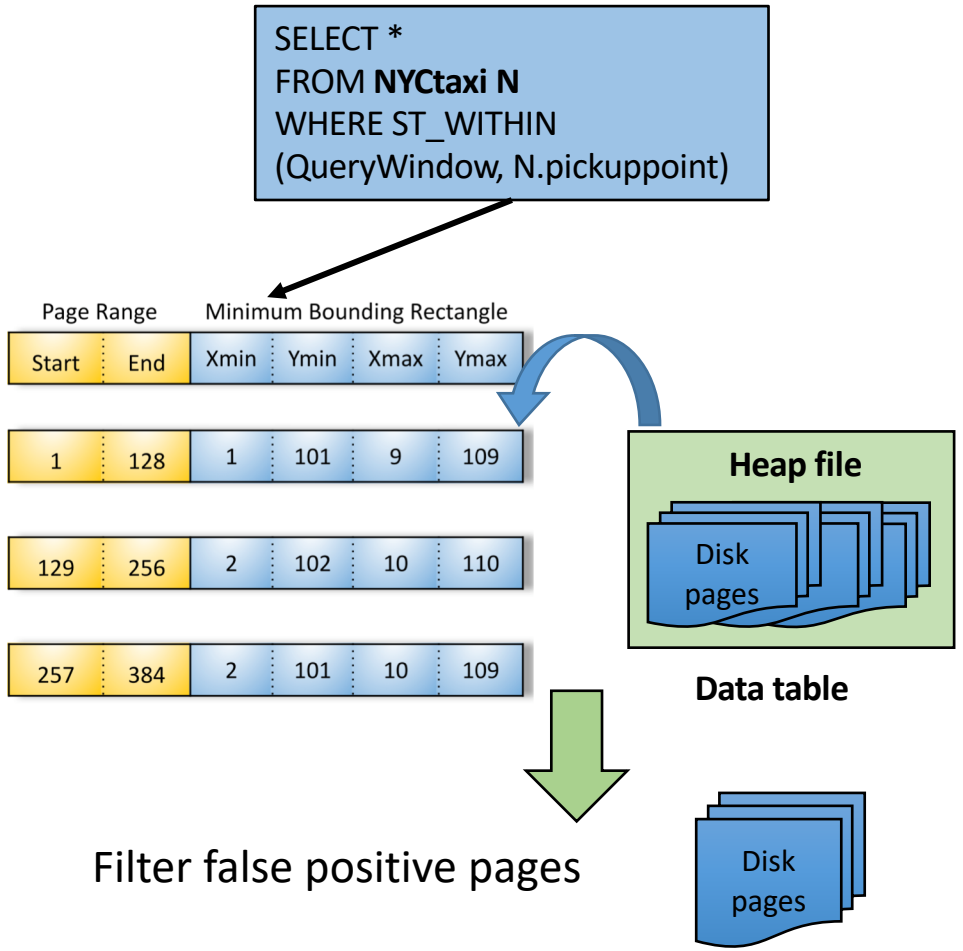
Compared Approaches: GiST

- Summary of GiST
  - Fast index search
  - Large storage overhead: 20% or more additional overhead
  - Slow maintenance: Split, merge tree nodes

Index Name	Data size	Index size	Initial. time	Insertion (0.1%)
GiST	175 GB NYC	84 GB	28 hours	6 hours

Compared Approaches: BRIN-Spatial

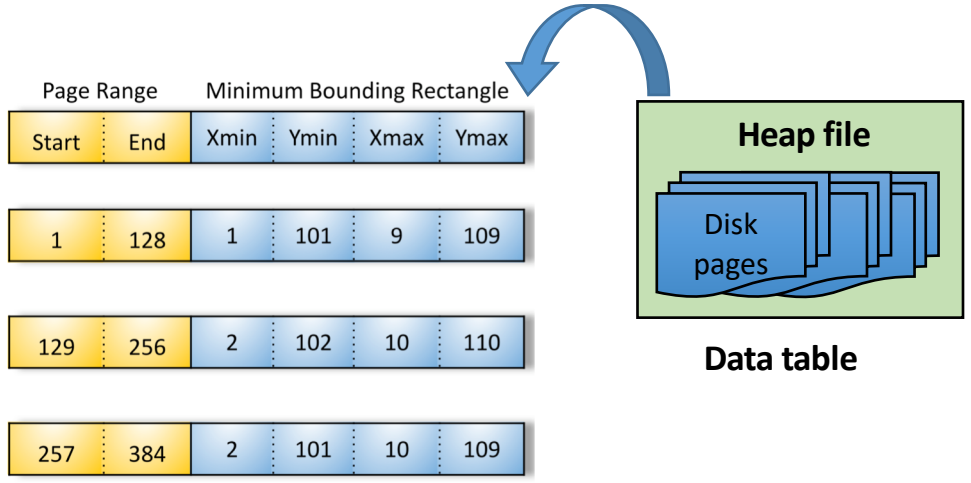
- Block Range Index (BRIN-Spatial):
  - PostgreSQL 9.5, PostGIS 2.3
  - Index heap file pages
  - Index search:
    - 1. Serial search by checking Query Window with MBR
    - 2. Filter false positive pages
  - Index maintenance:
    - Update MBR for Insertion
    - No update for deletion



Arizona State University

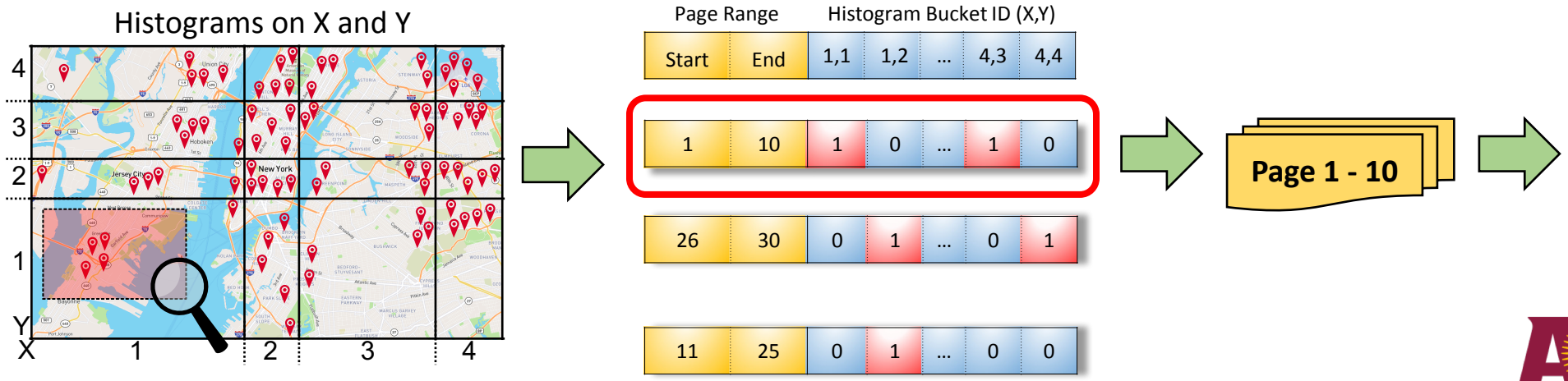
Compared Approaches: BRIN-Spatial

- Summary of BRIN-Spatial
  - Index heap file pages
  - Very small
  - Fast maintenance
  - Not good at queries



Compared Approaches: Hippo-Spatial

- Hippo-Spatial: PVLDB 2016
  - Index heap file pages
  - Index entry: dynamic page range, partial histogram
  - Index search:
    - 1. Serial search by finding overlapped buckets between Query Window and partial histogram
    - 2. Filter false positive pages

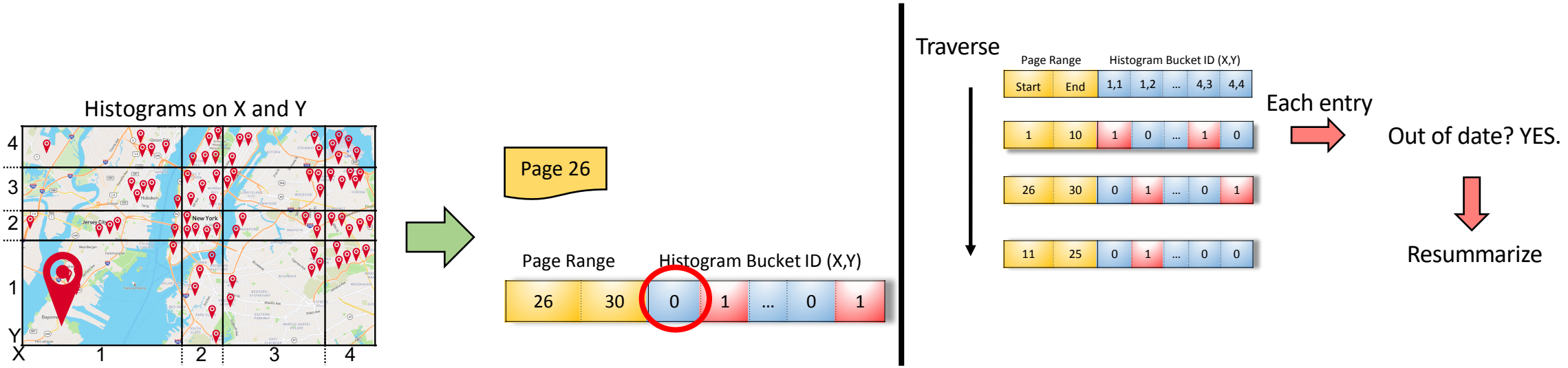


Page	False positive
1	√
2	√
3	Got results!
4	Got results!
5	√
6	Got results!
7	√
8	Got results!
9	Got results!
10	√

Arizona State University

Compared Approaches: Hippo-Spatial

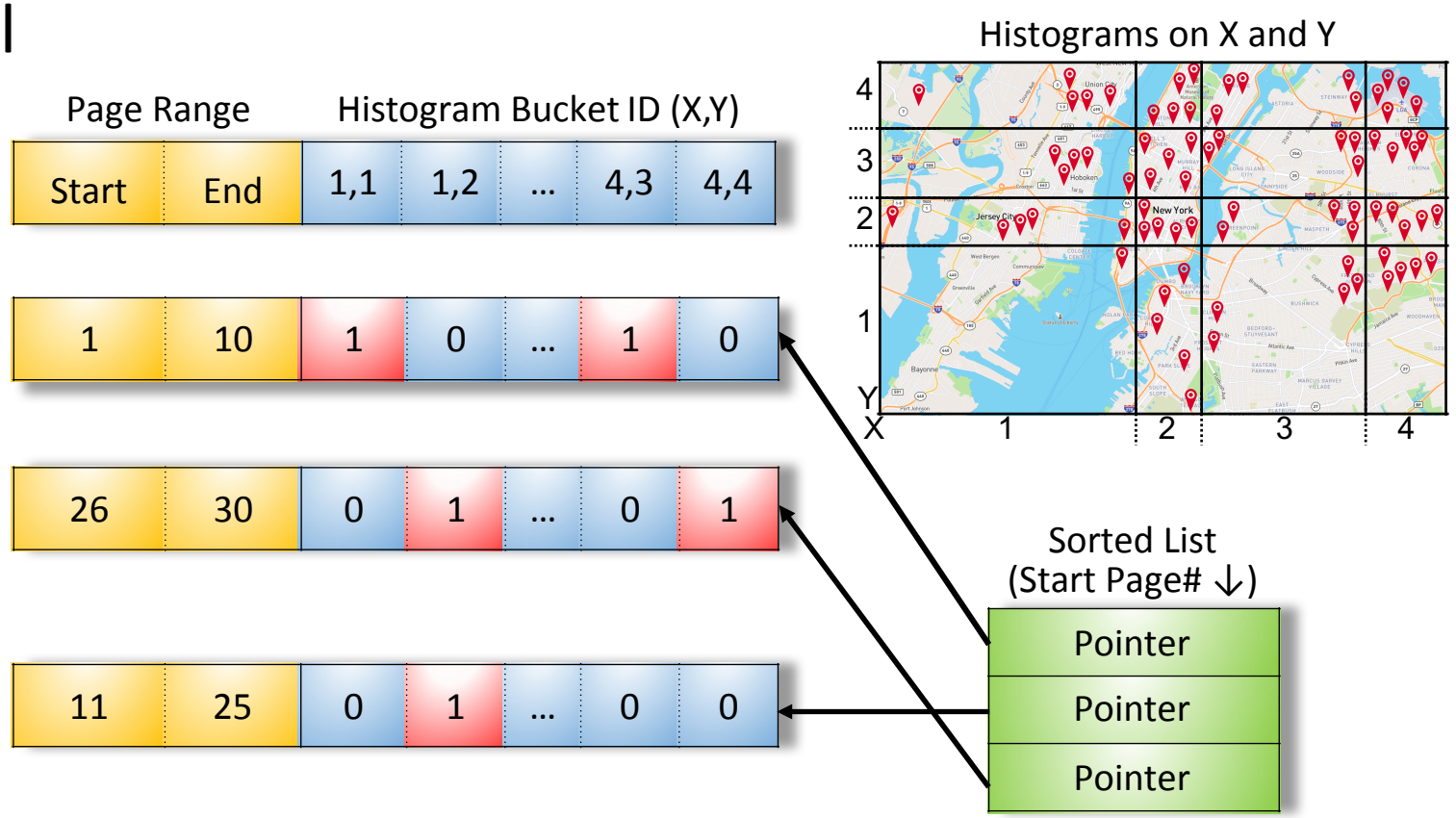
- Hippo-Spatial:
  - Index maintenance
    - Data insertion: eager update on partial histogram
    - Data deletion: lazy update on partial histogram



Arizona State University

Compared Approaches: Hippo-Spatial

- Summary of Hippo-Spatial
  - Index heap file pages
  - Still small
  - Fast maintenance
  - Good at common queries



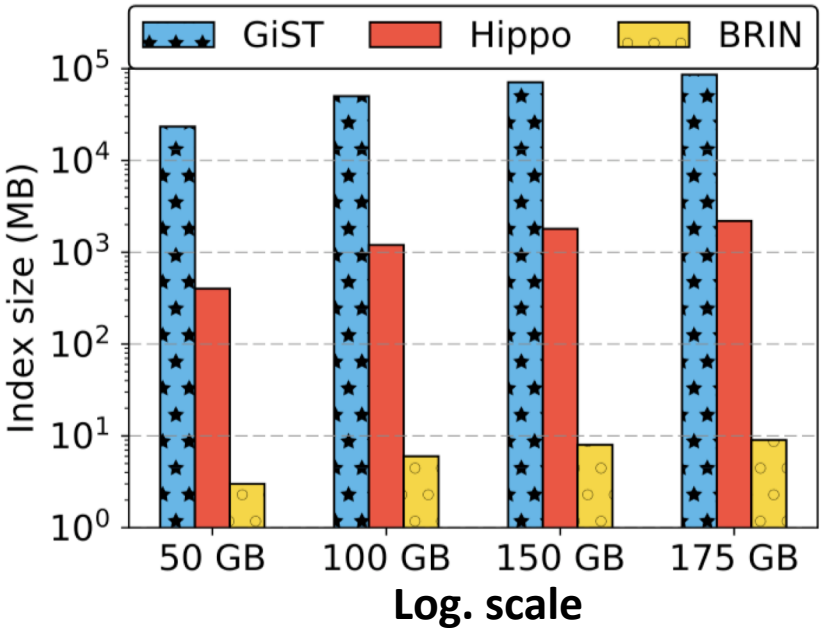
Arizona State University

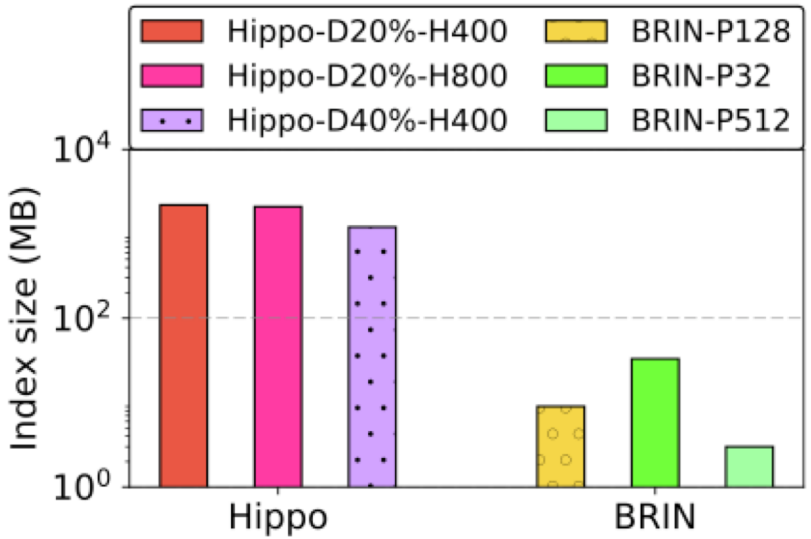
Experimental Environment

- Datasets
  - NYC Taxi Trips 175 GB
- Parameter setting
  - Hippo: Histogram bucket (H) 400, Partial histogram density(D) 20%
  - BRIN: Page per range (P) 128

Indexing Overhead

- Index size
  - Hippo: 100x < GiST
  - BRIN: 100x < Hippo
- Reason
  - Index pages not tuples
  - Partial histogram > MBR

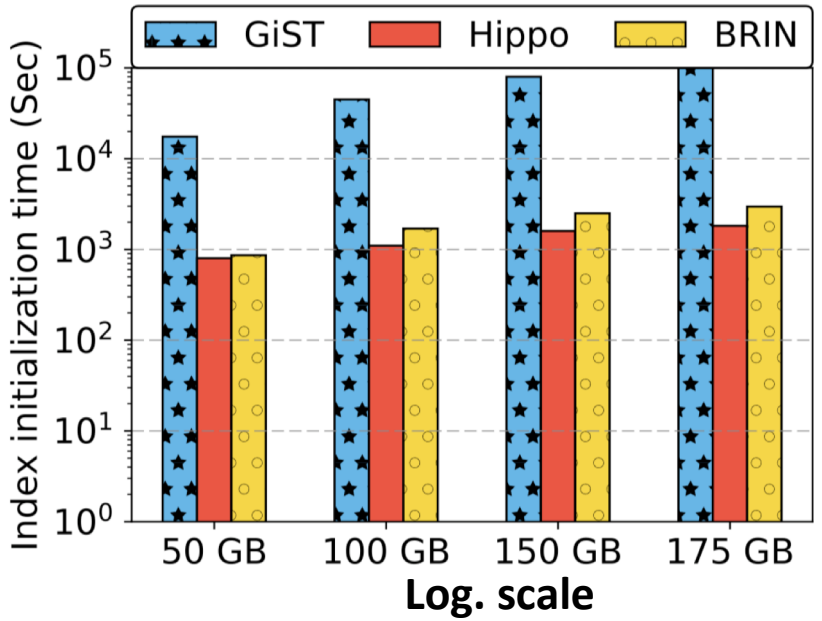


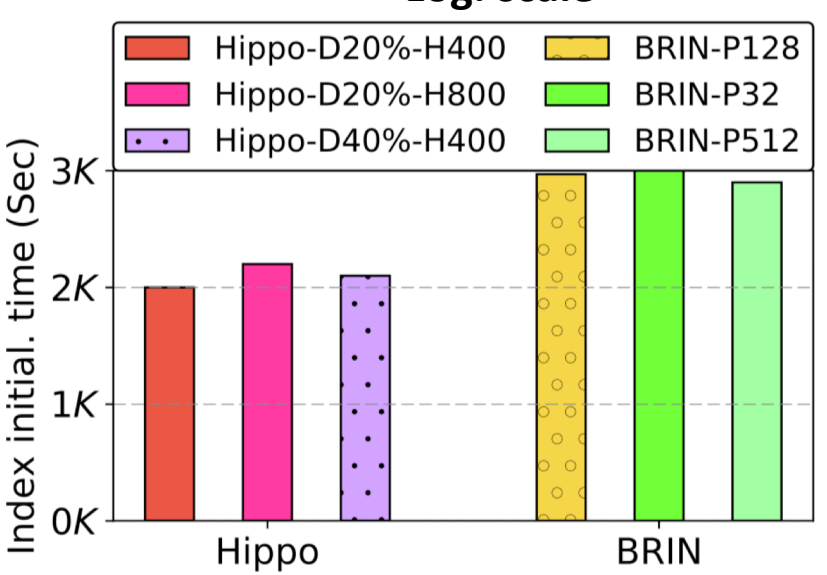


Arizona State University

Indexing Overhead (cont.)

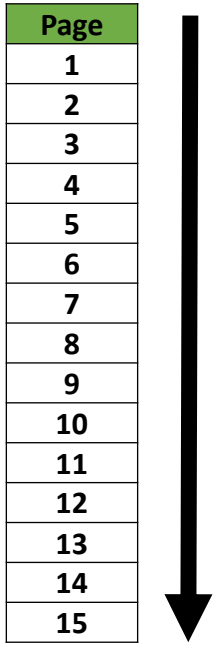
- Index initialization time
  - Hippo, BRIN-Spatial 100x < GiST
  - Hippo takes 60% time of BRIN
- Reason
  - Hierarchy > flat index structure
  - GiST writes lots of temporary disk files
  - BRIN in-memory entry is updated frequently



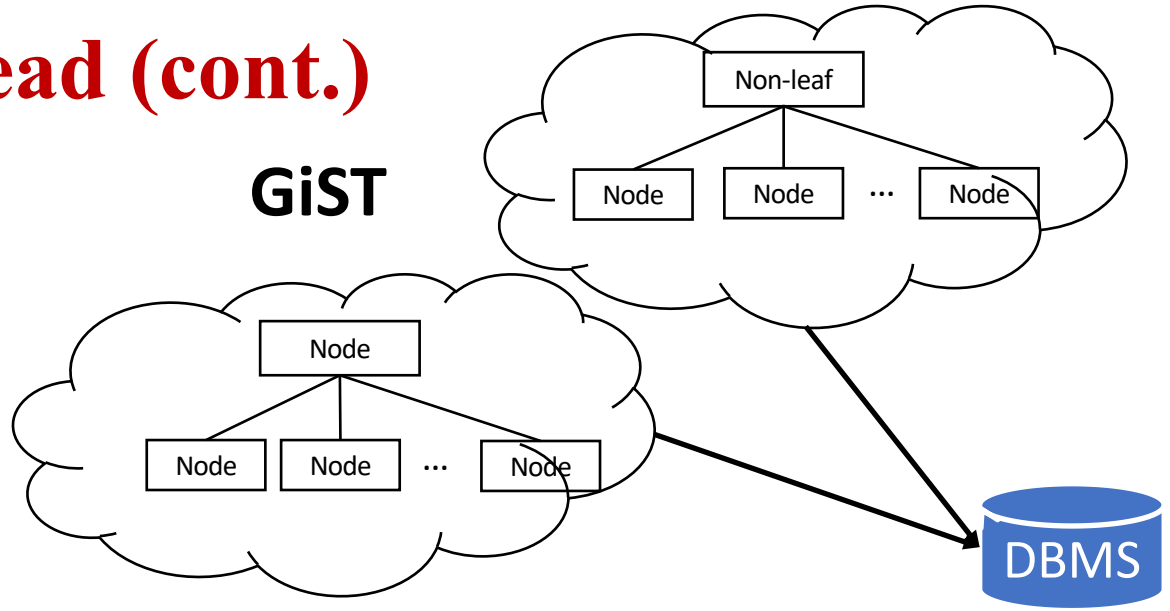


Arizona State University

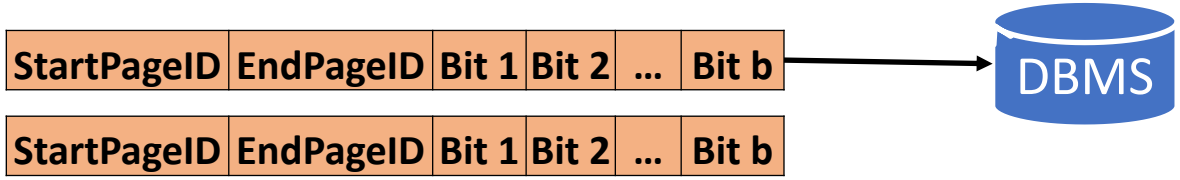
Indexing Overhead (cont.)



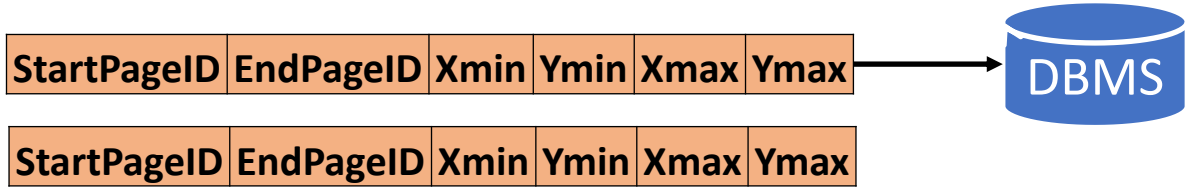
GiST



Hippo



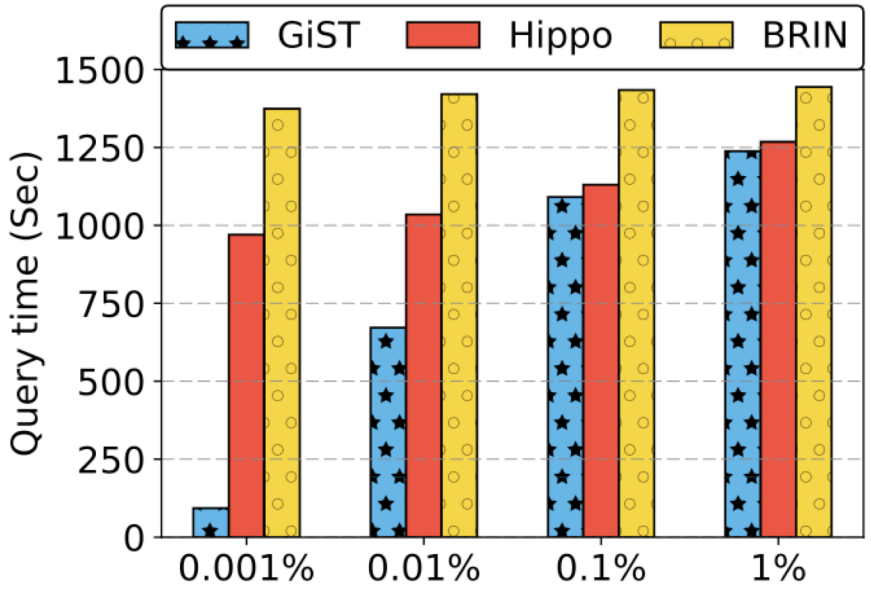
BRIN



Arizona State University

Query Response Time: vary query selectivity factor

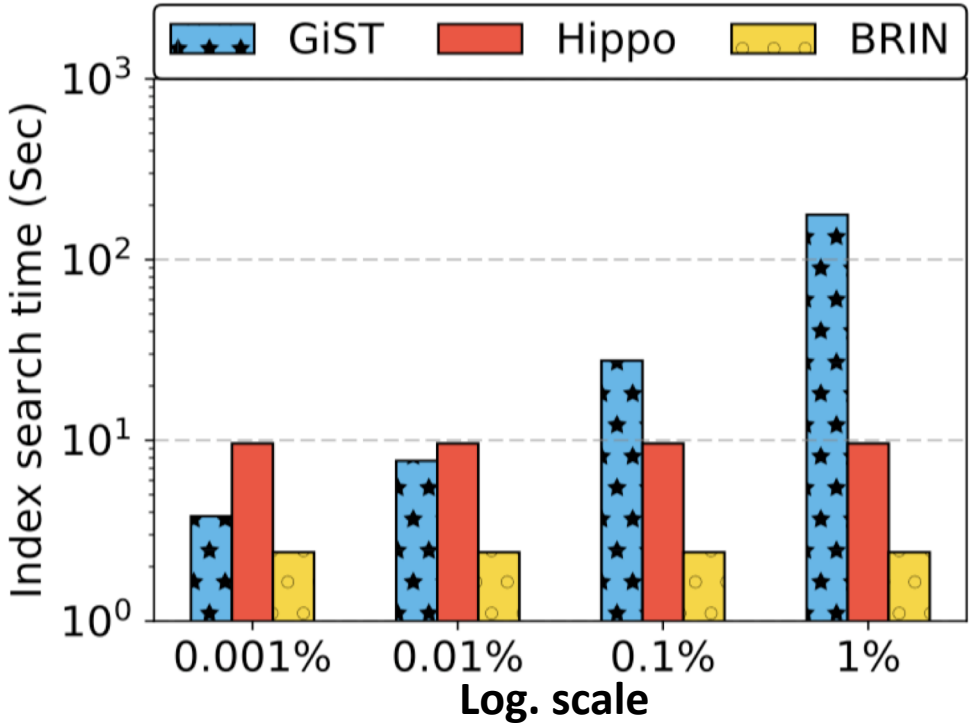
- Hippo ≈ GiST at 0.1% and 1% selectivity
- BRIN is always the worst



Arizona State University

Index Probe Time: vary query selectivity factor

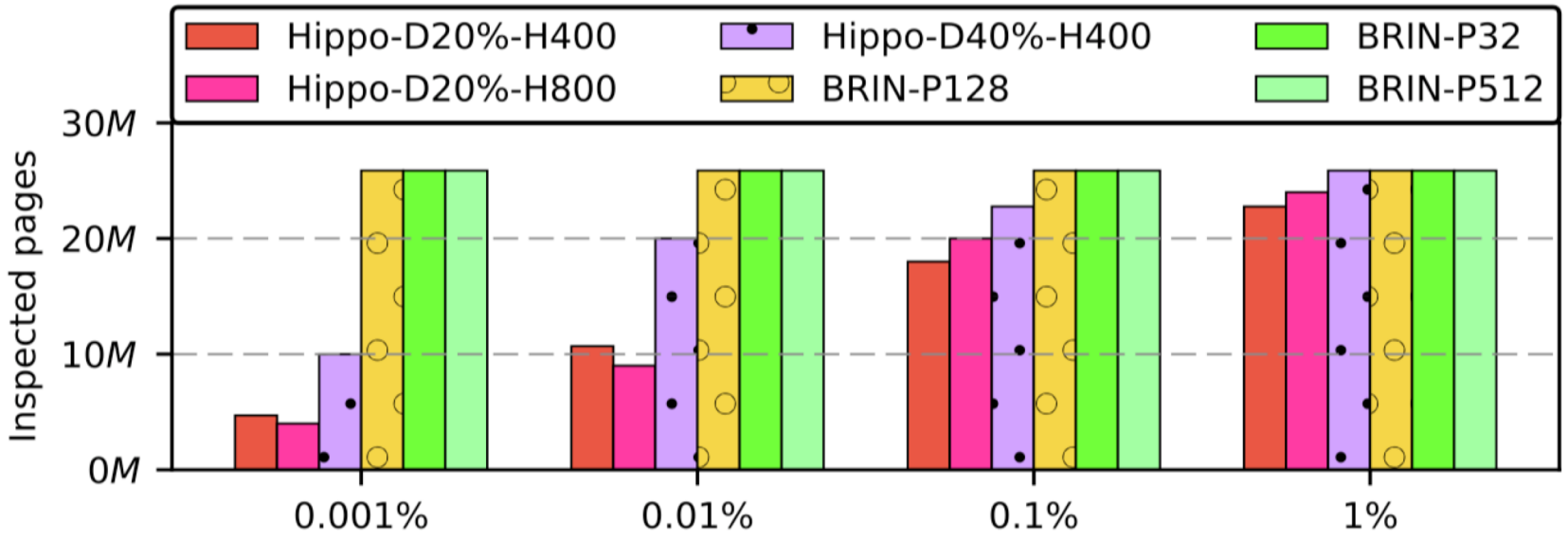
- Hippo and BRIN have constant index probe time
  - Search all index entries for a given query
- GiST index probe time increases along with selectivity factor



Arizona State University

Inspected Pages: vary query selectivity factor

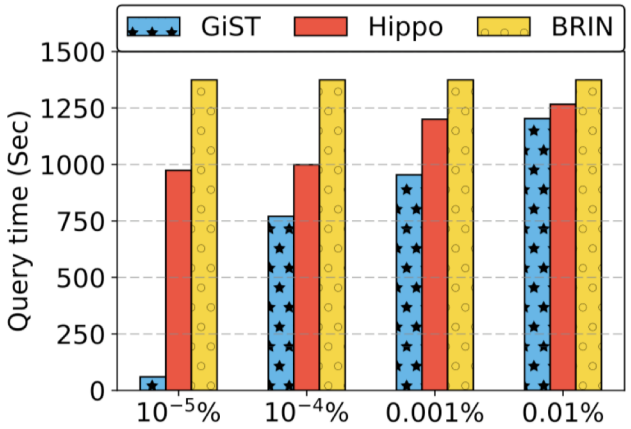
- Hippo inspects 5 times less disk pages than BRIN
- BRIN searches too many pages with 32, 128, 512 pages per range
- Higher density makes Hippo inspect more pages



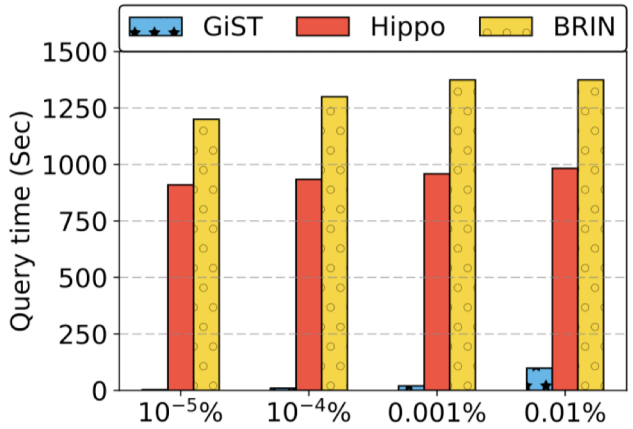
Arizona State University

Query Response Time: vary query areas

- Setting
  - Area: percent of NYC region area
  - Dense locations, Time Square, JFK,…; Random locations, random within NYC
- Hippo works better in dense locations, medium selectivity factors
- GiST works better in random locations, highly selective queries



(a) Queries in dense locations

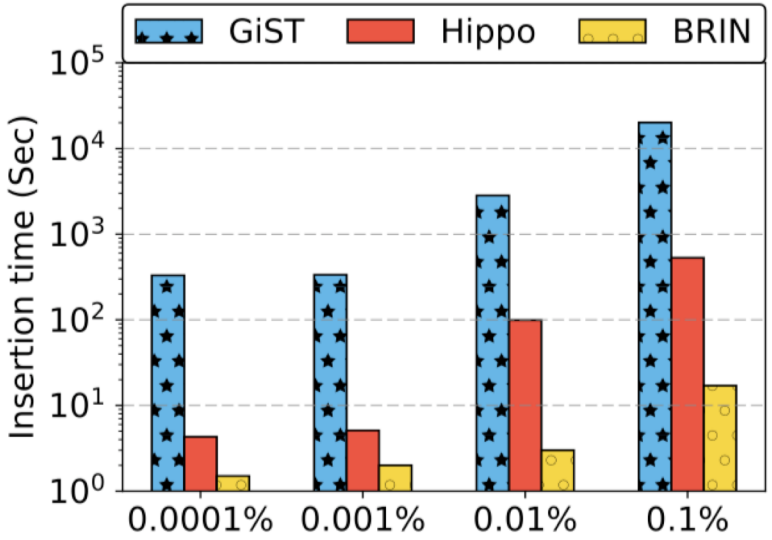


(b) Queries in random locations

Arizona State University

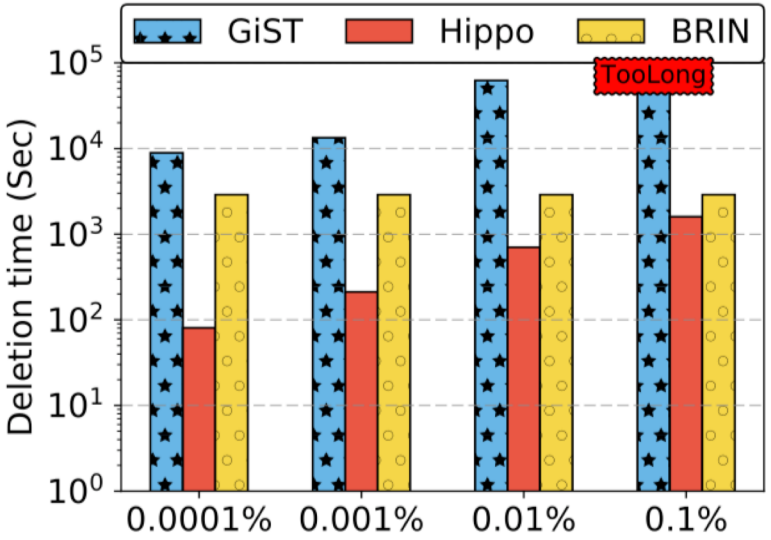
Maintenance time: vary update ratio

- Insertion:
  - Hippo 100x < GiST, flat index structure
  - BRIN 50x < Hippo, Hippo updates partial histogram
- Deletion: Hippo 100x < GiST; BRIN > Hippo, BRIN has to re-build



(a) Data insertion time

Log. scale

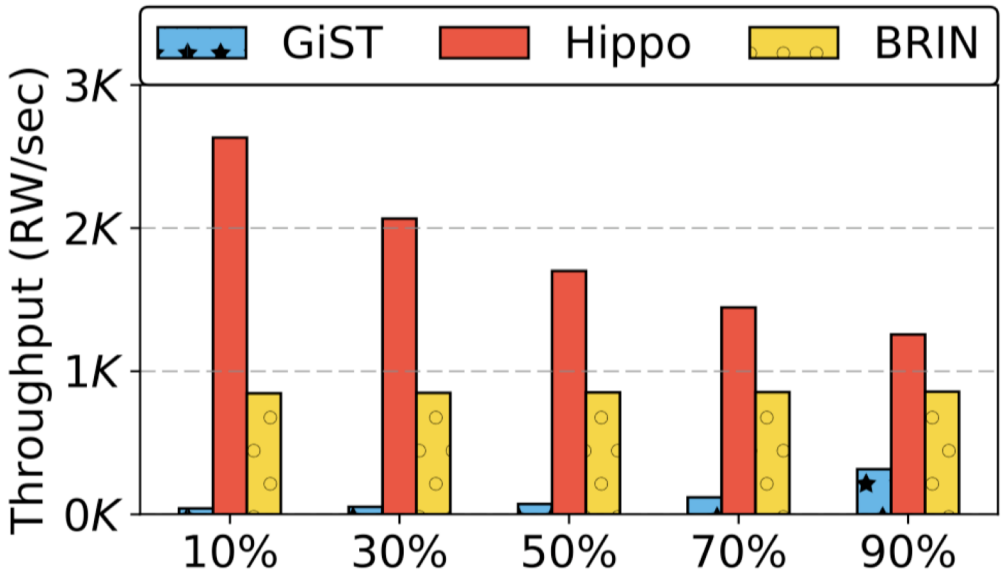


(b) Data deletion time

Arizona State University

Throughput: Hybrid workloads

- Queries + Updates
- Update-intensive workloads (10%-50%), Hippo is 100x > GiST
- Query-intensive workloads (70%-90%), Hippo ≈ GiST



Arizona State University

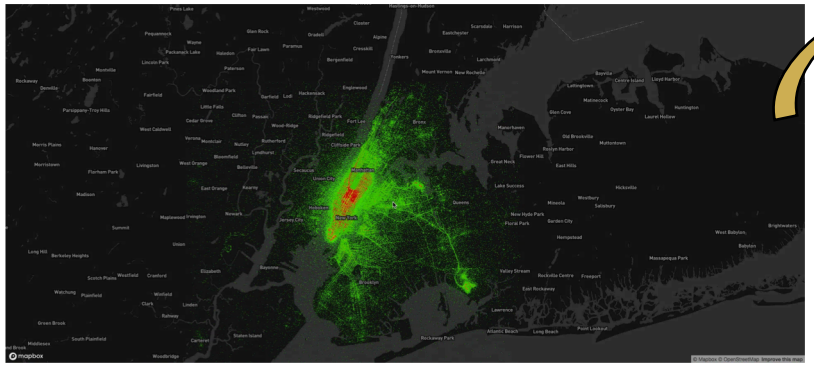
Summary of Results

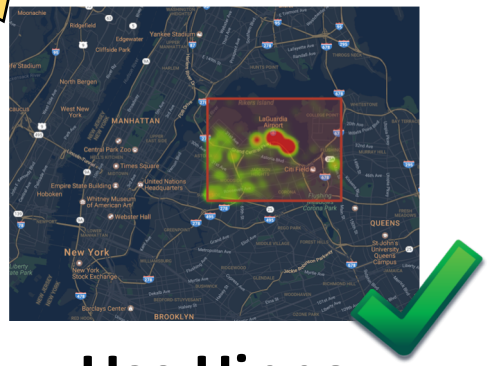
Metric	GiST-Spatial	Hippo-Spatial	BRIN-Spatial
Storage overhead	84 GB	2 GB	10 MB
Initialization time	28 hours	30 minutes	45 minutes
Favored selectivity query	0.001% selectivity	selectivity between 0.01% and 1%	X
Favored dense area query	10^{-5}% range query area	range query area ≥ 10^{-4}%	X
Index insertion	6 minutes for inserting 10^{-4}% data	4 seconds for inserting 10^{-4}% data	1 second for inserting 10^{-4}% data
Index deletion	2 hours for deleting 10^{-4}% data	2 min for deleting 10^{-4}% data	Index rebuilt
Hybrid workload	Query-intensive	Balanced Workload and Update-intensive	Update-intensive

Arizona State University

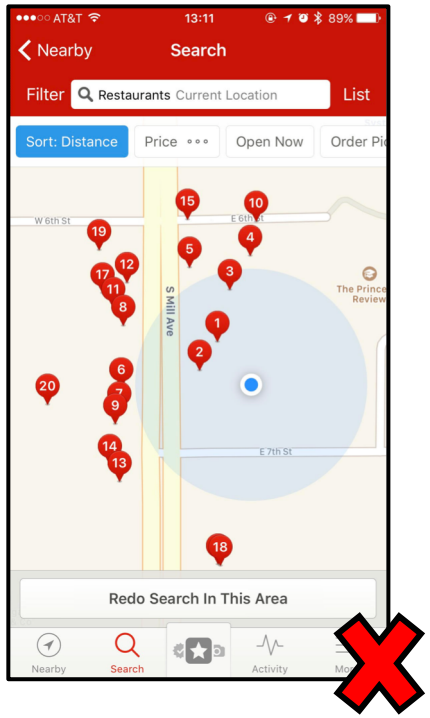
Take-home Lesson

- **Do not use GiST** (spatial tree index) if limited storage
- **Do not use BRIN or Hippo** for Yelp-like applications.
- **Use Hippo** for spatial analytics applications over dynamic and dense spatial data.
  - query selectivity is 0.1% - 1%, update-intensive workloads





Use Hippo



Use GiST

Arizona State University

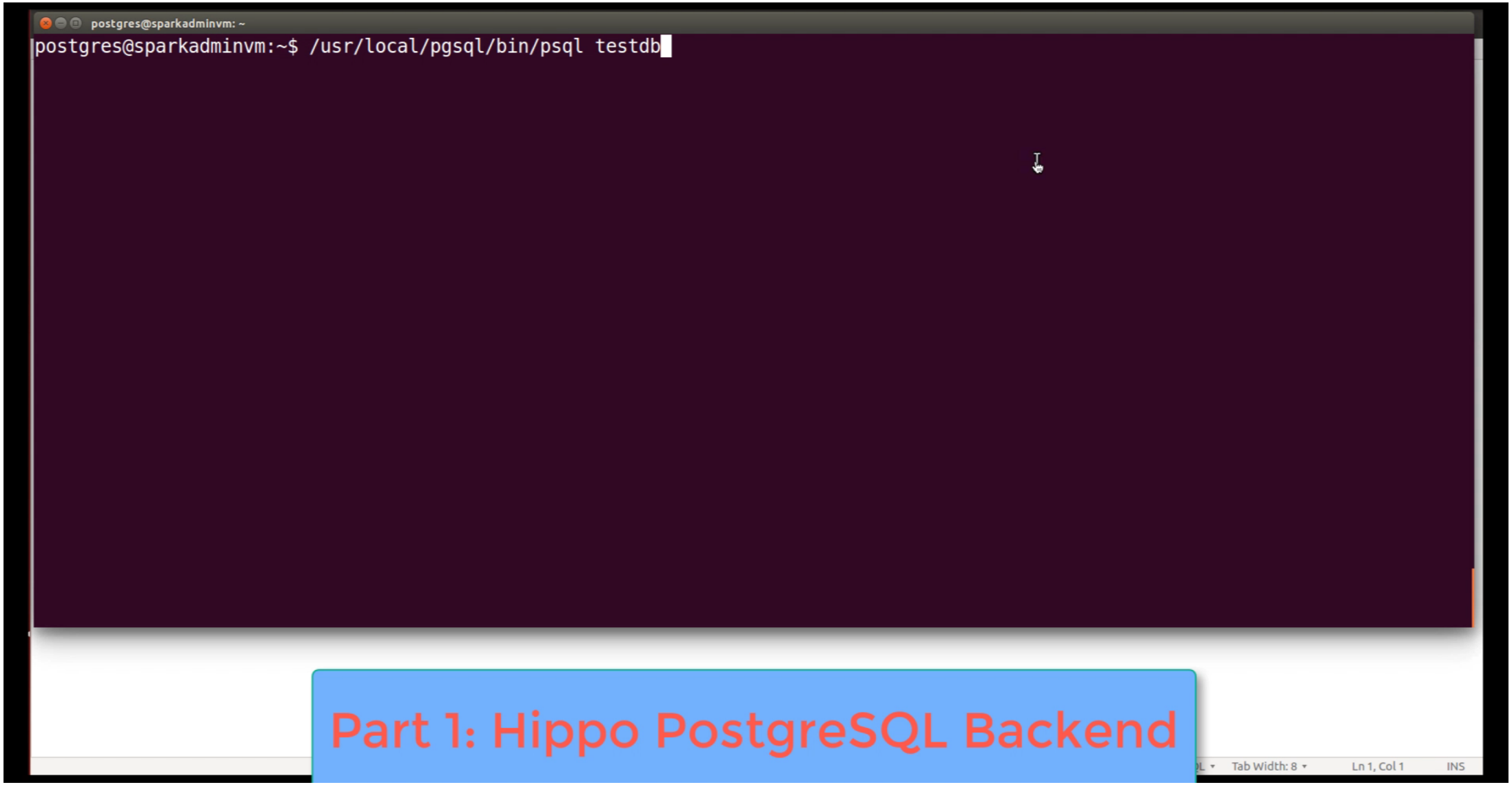
Questions?

https://github.com/DataSystemsLab/hippo-postgresql

Build Hippo

CREATE INDEX hippo_idx ON hippo_tbl USING **hippo** (randomNumber) WITH (density = 20);

Arizona State University



Arizona State University