Slalom: Coasting Through Raw Data via Adaptive Partitioning and Indexing

Matthaios Olma[‡] Manos Karpathiotakis[‡] Ioannis Alagiannis[*]

Manos Athanassoulis[§] Anastasia Ailamaki[‡]

Presented by Ziyu Shen

The problem

- Data generated has grown massively
  - Sensor data (ex. from management, clinical studies, Financial market)
  - Network monitoring data
  - etc

The problem

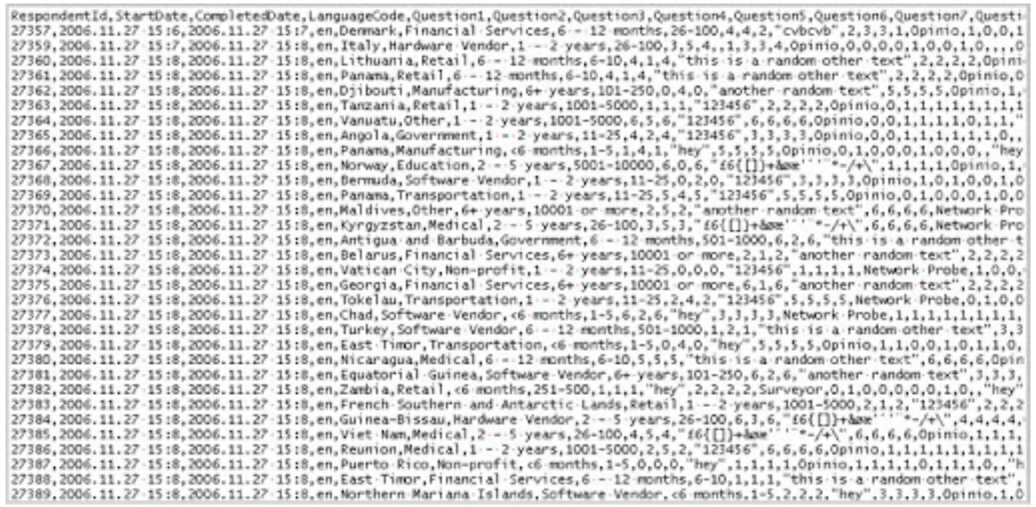
Current analytical system not built for this much data

- Time-to-insight is high
  - **Dynamic**
  - **Data-driven**

- Analyzing datasets is a costly task – Need new data management systems
  - ***Data loading*** is expensive
  - Traditional DBMSs need ***data loading*** and ***index building*** to offer interactive access
  - ***Transformation***, ***copying***, and ***preparation*** introduce delays

Current Solution

- Queries over Raw Data
  - Reduce the data-to-insight cost
  - Only logical indexes
  - Physical data is never changed
- In-situ queries
  - Only need a initial table scan
  - Processing avoids the costly phase of data loading
  - Retrieve only the relevant data
- NoDB for PostgresRaw
  - Builds positional maps



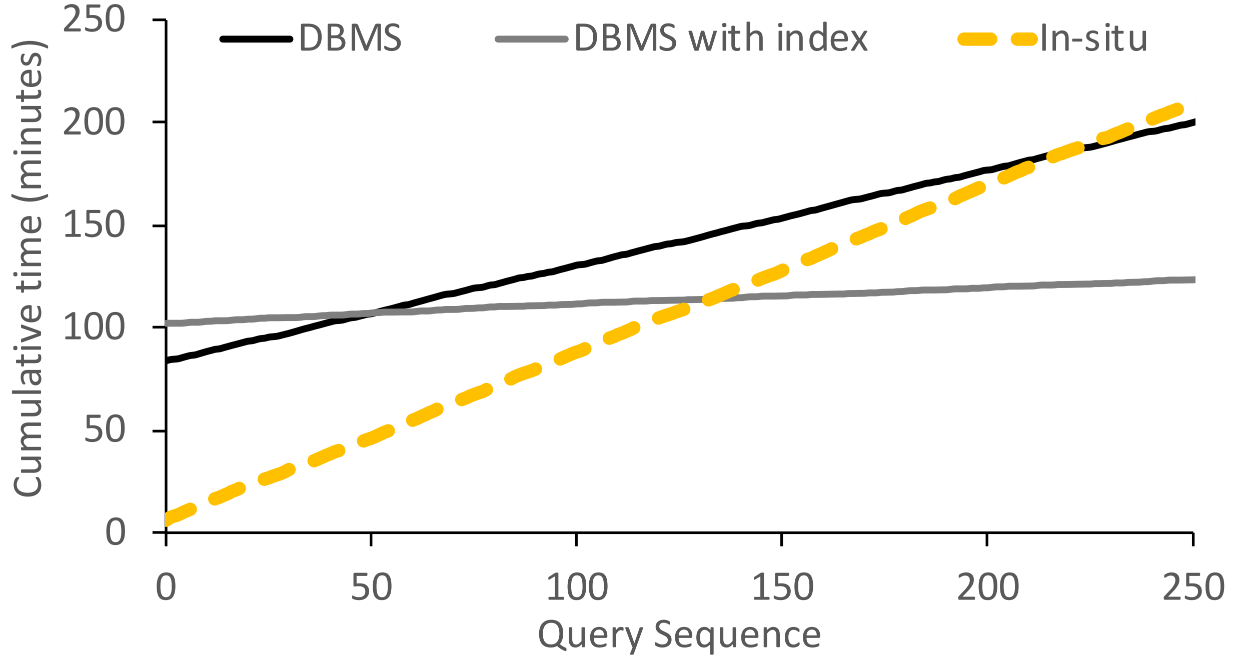
```
RespondentId,StartDate,CompletedDate,LanguageCode,Question1,Question2,Question3,Question4,Question5,Question6,Question7,Questi
27357,2006.11.27 15:6,2006.11.27 15:7,en,Denmark,Financial Services,6 - 12 months,26-100,4,4,2,"cvbcvb",2,3,3,1,Opinio,1,0,0,1
27359,2006.11.27 15:7,2006.11.27 15:8,en,Italy,Hardware Vendor,1 - 2 years,26-100,3,5,4,,1,3,3,4,Opinio,0,0,0,0,1,0,0,1,0,,,,0
27360,2006.11.27 15:8,2006.11.27 15:8,en,Lithuania,Retail,6 - 12 months,6-10,4,1,4,"this is a random other text",2,2,2,2,Opini
27361,2006.11.27 15:8,2006.11.27 15:8,en,Panama,Retail,6 - 12 months,6-10,4,1,4,"this is a random other text",2,2,2,2,Opinio,0
27362,2006.11.27 15:8,2006.11.27 15:8,en,Djibouti,Manufacturing,6+ years,101-250,0,4,0,"another random text",5,5,5,5,Opinio,1,
27363,2006.11.27 15:8,2006.11.27 15:8,en,Tanzania,Retail,1 - 2 years,1001-5000,1,1,1,"123456",2,2,2,2,Opinio,0,1,1,1,1,1,1,1
27364,2006.11.27 15:8,2006.11.27 15:8,en,Vanuatu,Other,1 - 2 years,1001-5000,6,5,6,"123456",6,6,6,6,Opinio,0,0,1,1,1,1,0,1,1,"
27365,2006.11.27 15:8,2006.11.27 15:8,en,Angola,Government,1 - 2 years,11-25,4,2,4,"123456",3,3,3,3,Opinio,0,0,1,1,1,1,1,1,0,,
27366,2006.11.27 15:8,2006.11.27 15:8,en,Panama,Manufacturing,<6 months,1-5,1,4,1,"hey",5,5,5,5,Opinio,0,1,0,0,0,1,0,0,0,,"hey
27367,2006.11.27 15:8,2006.11.27 15:8,en,Norway,Education,2 - 5 years,5001-10000,6,0,6,"£6{[]}+&æ'`'"*-/+\",1,1,1,1,Opinio,1,
27368,2006.11.27 15:8,2006.11.27 15:8,en,Bermuda,Software Vendor,1 - 2 years,11-25,0,2,0,"123456",3,3,3,3,Opinio,1,0,1,0,0,1,0
27369,2006.11.27 15:8,2006.11.27 15:8,en,Panama,Transportation,1 - 2 years,11-25,5,4,5,"123456",5,5,5,5,Opinio,0,1,0,0,0,1,0,0
27370,2006.11.27 15:8,2006.11.27 15:8,en,Maldives,Other,6+ years,10001 or more,2,5,2,"another random text",6,6,6,6,Network Pro
27371,2006.11.27 15:8,2006.11.27 15:8,en,Kyrgyzstan,Medical,2 - 5 years,26-100,3,5,3,"£6{[]}+&æ'`'"*-/+\",6,6,6,6,Network Pro
27372,2006.11.27 15:8,2006.11.27 15:8,en,Antigua and Barbuda,Government,6 - 12 months,501-1000,6,2,6,"this is a random other t
27373,2006.11.27 15:8,2006.11.27 15:8,en,Belarus,Financial Services,6+ years,10001 or more,2,1,2,"another random text",2,2,2,2
27374,2006.11.27 15:8,2006.11.27 15:8,en,Vatican City,Non-profit,1 - 2 years,11-25,0,0,0,"123456",1,1,1,1,Network Probe,1,0,0,
27375,2006.11.27 15:8,2006.11.27 15:8,en,Georgia,Financial Services,6+ years,10001 or more,6,1,6,"another random text",2,2,2,2
27376,2006.11.27 15:8,2006.11.27 15:8,en,Tokelau,Transportation,1 - 2 years,11-25,2,4,2,"123456",5,5,5,5,Network Probe,0,1,0,0
27377,2006.11.27 15:8,2006.11.27 15:8,en,Chad,Software Vendor,<6 months,1-5,6,2,6,"hey",3,3,3,3,Network Probe,1,1,1,1,1,1,1,1,
27378,2006.11.27 15:8,2006.11.27 15:8,en,Turkey,Software Vendor,6 - 12 months,501-1000,1,2,1,"this is a random other text",3,3
27379,2006.11.27 15:8,2006.11.27 15:8,en,East Timor,Transportation,<6 months,1-5,0,4,0,"hey",5,5,5,5,Opinio,1,1,0,0,1,0,1,1,0,
27380,2006.11.27 15:8,2006.11.27 15:8,en,Nicaragua,Medical,6 - 12 months,6-10,5,5,5,"this is a random other text",6,6,6,6,Opin
27381,2006.11.27 15:8,2006.11.27 15:8,en,Equatorial Guinea,Software Vendor,6+ years,101-250,6,2,6,"another random text",3,3,3,
27382,2006.11.27 15:8,2006.11.27 15:8,en,Zambia,Retail,<6 months,251-500,1,1,1,"hey",2,2,2,2,Surveyor,0,1,0,0,0,0,0,1,0,,"hey"
27383,2006.11.27 15:8,2006.11.27 15:8,en,French Southern and Antarctic Lands,Retail,1 - 2 years,1001-5000,2,1,2,"123456",2,2,2
27384,2006.11.27 15:8,2006.11.27 15:8,en,Guinea-Bissau,Hardware Vendor,2 - 5 years,26-100,6,3,6,"£6{[]}+&æ'`'"*-/+\",4,4,4,4,
27385,2006.11.27 15:8,2006.11.27 15:8,en,Viet Nam,Medical,2 - 5 years,26-100,4,5,4,"£6{[]}+&æ'`'"*-/+\",6,6,6,6,Opinio,1,1,1,
27386,2006.11.27 15:8,2006.11.27 15:8,en,Reunion,Medical,1 - 2 years,1001-5000,2,5,2,"123456",6,6,6,6,Opinio,1,1,1,1,1,1,1,1,1
27387,2006.11.27 15:8,2006.11.27 15:8,en,Puerto Rico,Non-profit,<6 months,1-5,0,0,0,"hey",1,1,1,1,Opinio,1,1,1,1,0,1,1,1,0,,"h
27388,2006.11.27 15:8,2006.11.27 15:8,en,East Timor,Financial Services,6 - 12 months,6-10,1,1,1,"this is a random other text",
27389,2006.11.27 15:8,2006.11.27 15:8,en,Northern Mariana Islands,Software Vendor,<6 months,1-5,2,2,2,"hey",3,3,3,3,Opinio,1,0
```

Current Solution Disadvantages

Querying Raw Data Files Is Not Enough!

In-situ not suitable for large data size!

Reducing data to query time



* 60GB smart meter dataset, selectivity 1%, 128GB RAM, 1 thread

So how can we combine the performance of an indexed system with the in-situ data-to-query time?

Figure 1 shows the cumulative time of a DBMS system and a in-situ system during query sequence.

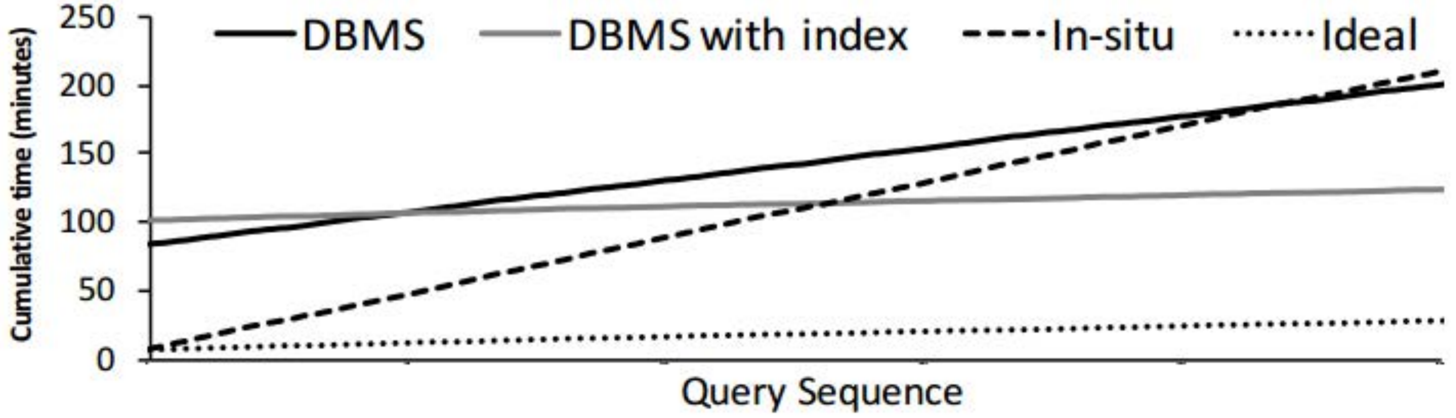


Figure 1: Ideally, in-situ data analysis should be able to retrieve only the relevant data for each query after the initial table scan (ideal - dotted line). In practice today, in-situ query processing avoids the costly phase of data loading (dashed line), however, as the number of the queries increases, the initial investment for full index on a DBMS pays off (the dashed line meets the grey line).

Enhancing in situ querying

- **Reduce** the amount of data accessed
  - **Partition** data to a favorable state
  - Build appropriate **indexes**

- Enable in situ data updates

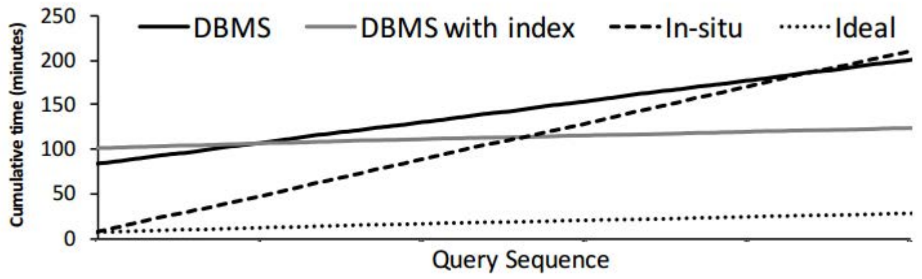
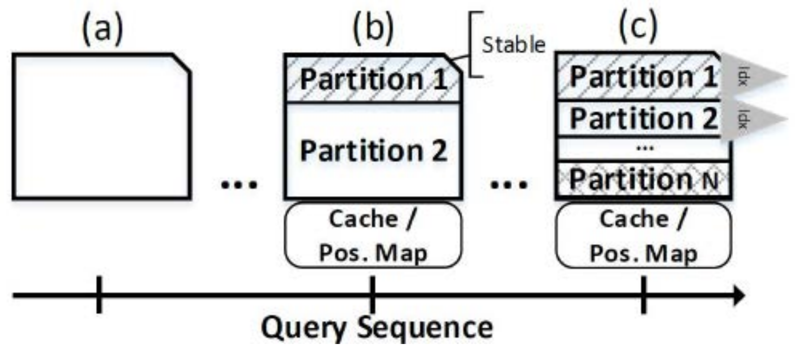
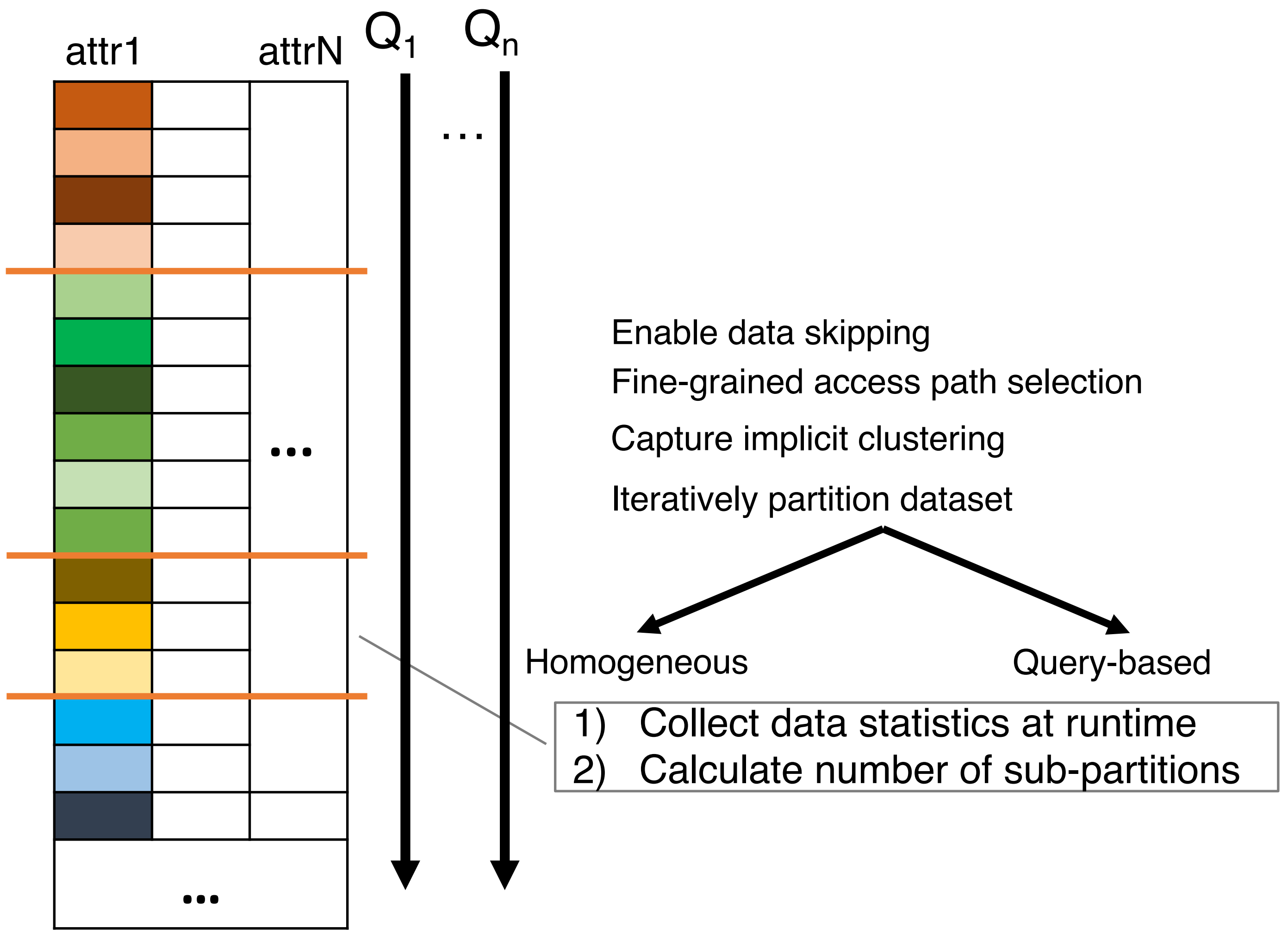


Figure 1: Ideally, in-situ data analysis should be able to retrieve only the relevant data for each query after the initial table scan (ideal - dotted line). In practice today, in-situ query processing avoids the costly phase of data loading (dashed line), however, as the number of the queries increases, the initial investment for full index on a DBMS pays off (the dashed line meets the grey line).

Partition manager

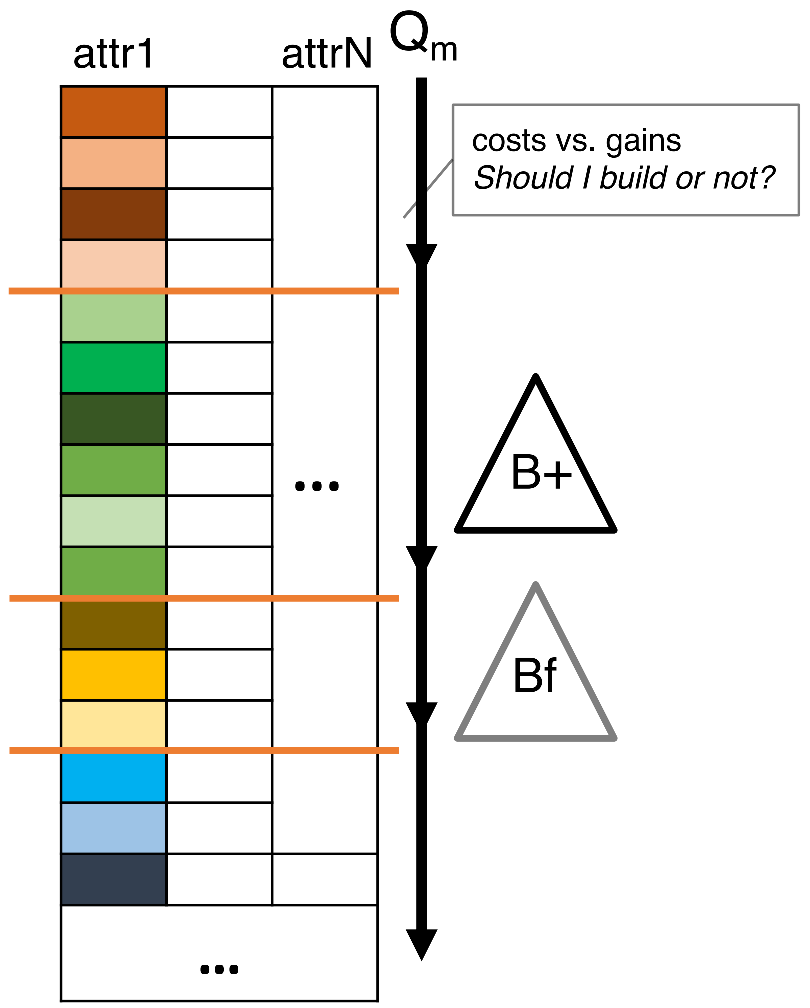
- Only logical partitions
- Contiguous and non-overlapping
- Iterative refinement
- Incremental splits
- Stops when partition is stable
- Splits into many smaller partitions




attr1
attrN
Q_1
Q_n
...
Enable data skipping
Fine-grained access path selection
Capture implicit clustering
Iteratively partition dataset
Homogeneous
Query-based
1) Collect data statistics at runtime
2) Calculate number of sub-partitions

Index manager

- Only applied to stable partitions
- Value existence
  - Bloom filters
  - Zone maps
- Value position
  - B+ Tree



Index tuning on partition level

Choose what & when to build

What
- Value-Existence (i.e., Bloom filters)
- Value-Position (i.e., B+ Trees)

When
- Based on randomized algorithm
- Cost of scan vs. cost of build + gain

Build and drop based on budget

Append and updates manager

Store partition state

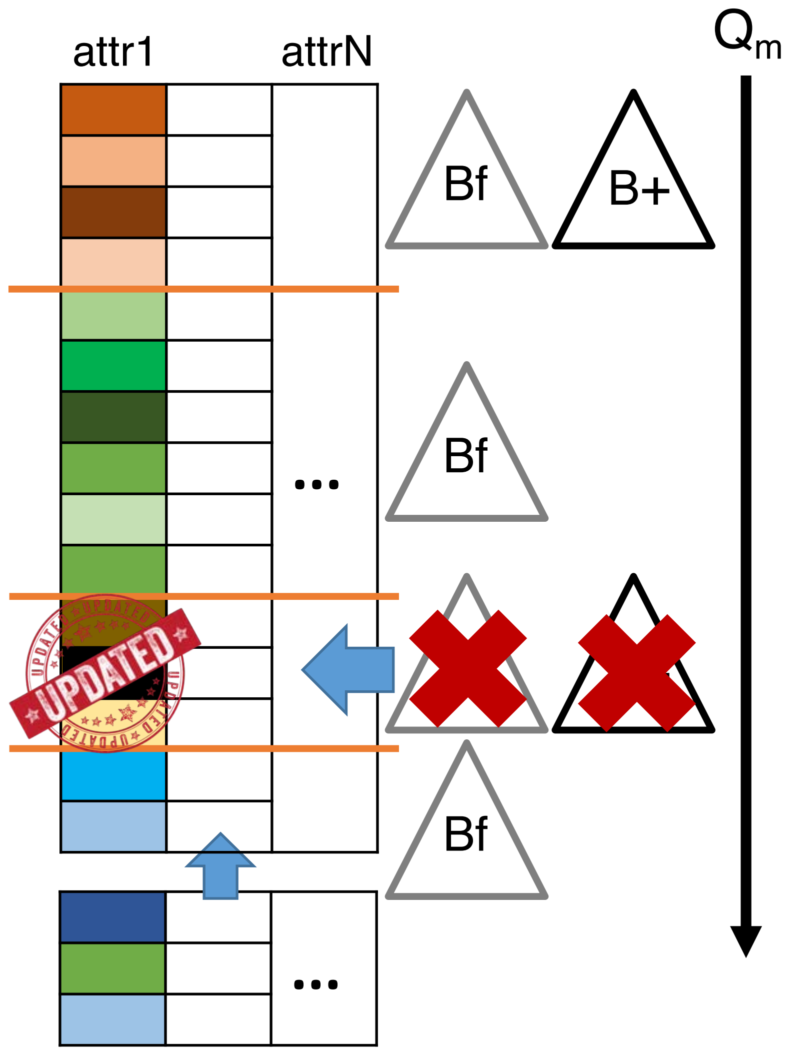
- Calculate hash value (MD5)

Monitor file for modifications

Recognize updated partitions

Fix modified partitions

- Drop/Re-build cache/index



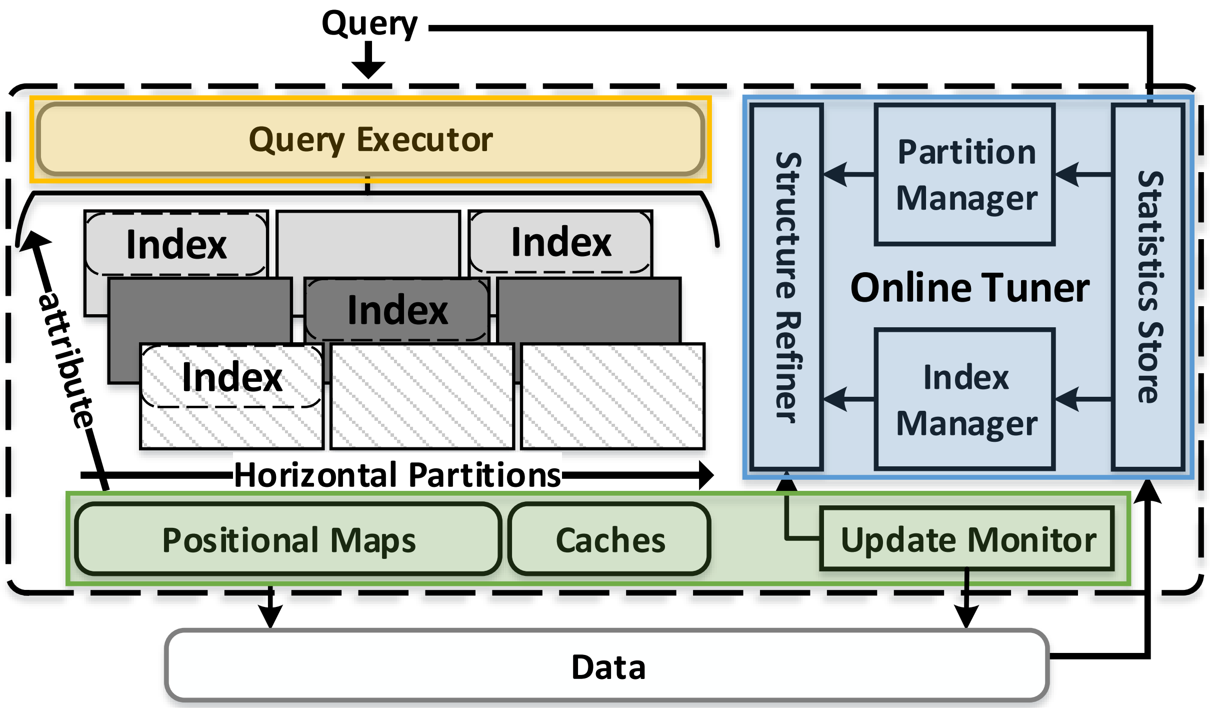
Minimize overhead of updates

THE SLALOM SYSTEM

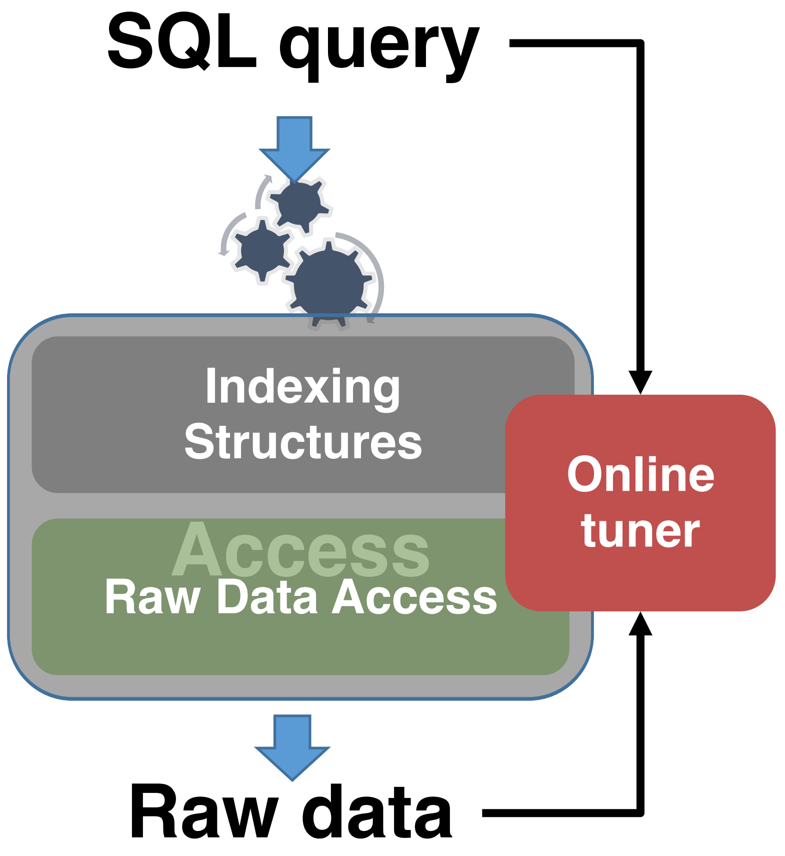
- Dynamic partitions
  - Logical partition only
  - Created at runtime
- Dynamic indexes
  - Bloom filters
  - Zonemaps
  - Bitmaps
  - B+ Trees

THE SLALOM SYSTEM

- Figure 2 shows the architecture of Slalom: Slalom combines an online partitioning and indexing tuner with a query executor featuring in-situ querying techniques.



Experiments Architecture



- Combine ***Online Tuning*** with ***Adaptive Indexing***
- ***Adapt data access*** to queries and data at ***runtime***

Experimental Setup

Hardware:

- Xeon CPU E5-2660 @ 2.20GHz, 2TB HDD - 7200RPM, 128GB RAM

Systems:

- Disk-based: PostgreSQL
- In-Memory: DBMS X
- In situ: PostgresRAW, Slalom with Stochastic Cracking

Slalom query latency evolution with time

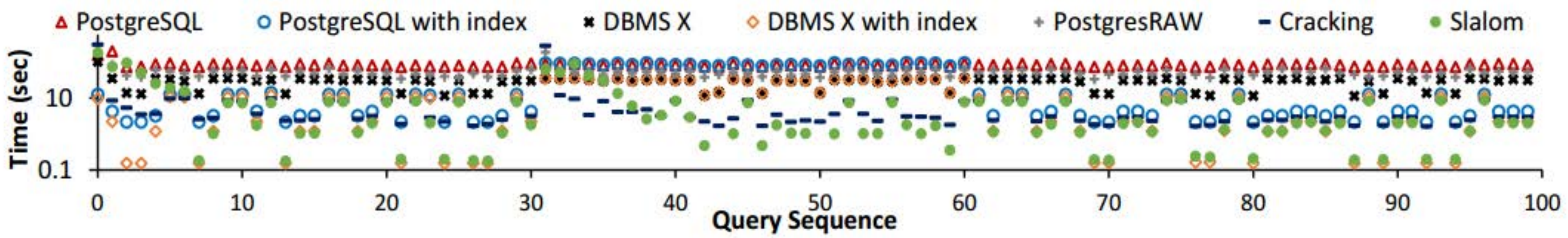


Figure 4: Sequence of 100 queries. Slalom dynamically refines its indexes to reach the performance of an index over loaded data.

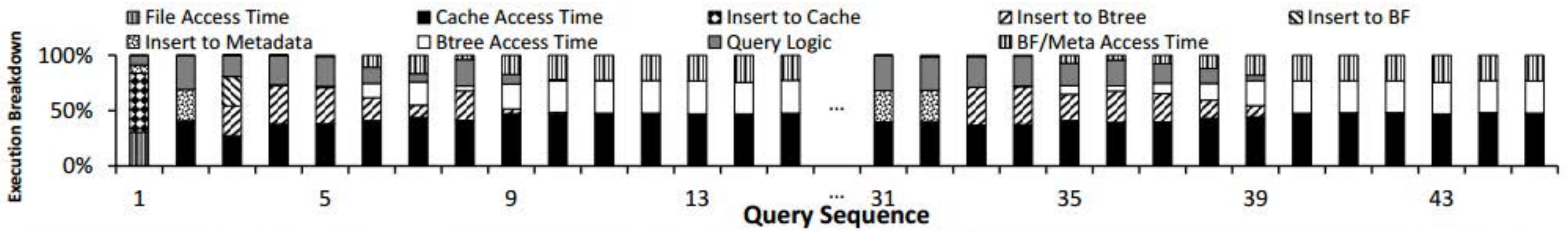


Figure 5: A breakdown of the operations taking place for Slalom during the execution of a subset of the 1000 point query sequence.

Slalom converges quickly to the fastest alternative configuration!

Experiments

- Figure 6 shows the full full workload of 1000 queries, from the raw data to result.
- In-situ adaptive indexing achieves interactive access

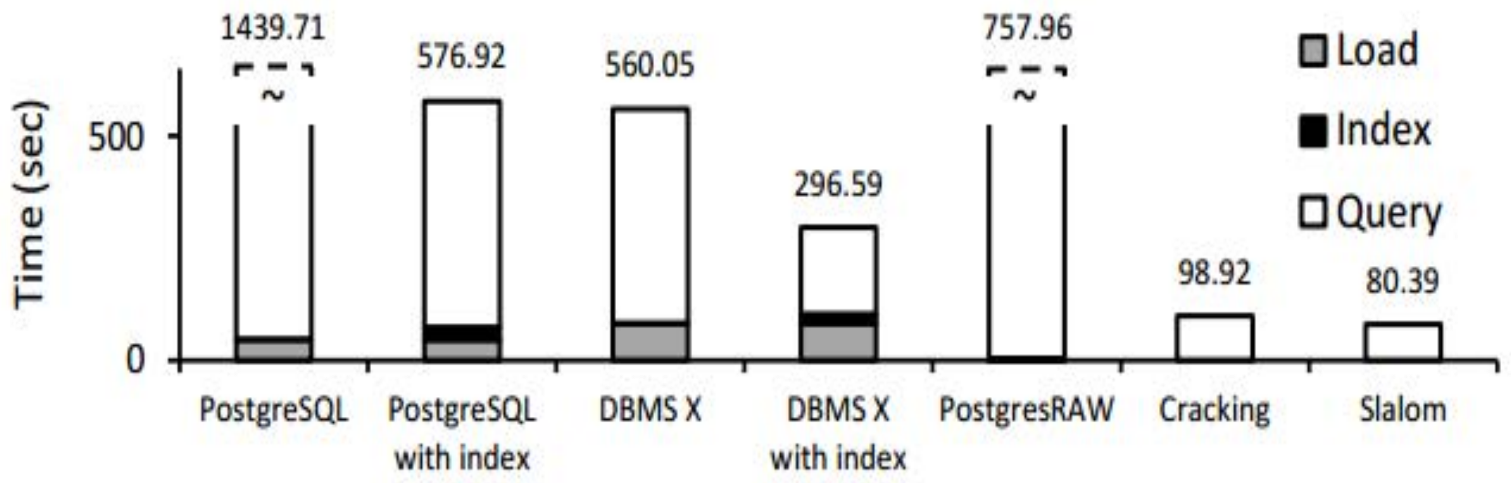


Figure 6: Sequence of 1000 queries. Slalom does not incur loading cost and dynamically builds indexes.

Overall, Cracking and Slalom offer comparable raw-data to results response time for this workload.

Experiments

- Figure 7 plots the Memory consumption of Slalom vs. a single fully-built B+ Tree for PostgreSQL and DBMS-X.

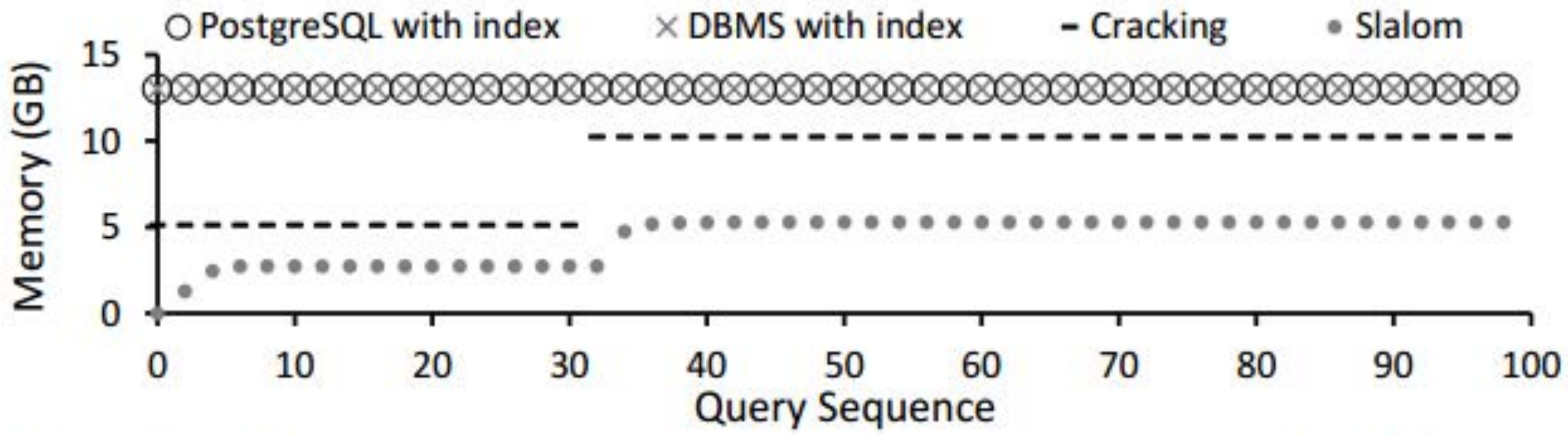


Figure 7: Memory consumption of Slalom vs. a single fully-built B+ Tree for PostgreSQL and DBMS-X. Slalom uses less memory because its indexes only target specific areas of a raw file.

Slalom consumes the minimum space among the four systems.

Experiments

- Figure 8 presents the number of tuples that Slalom accesses for each query in this experiment.

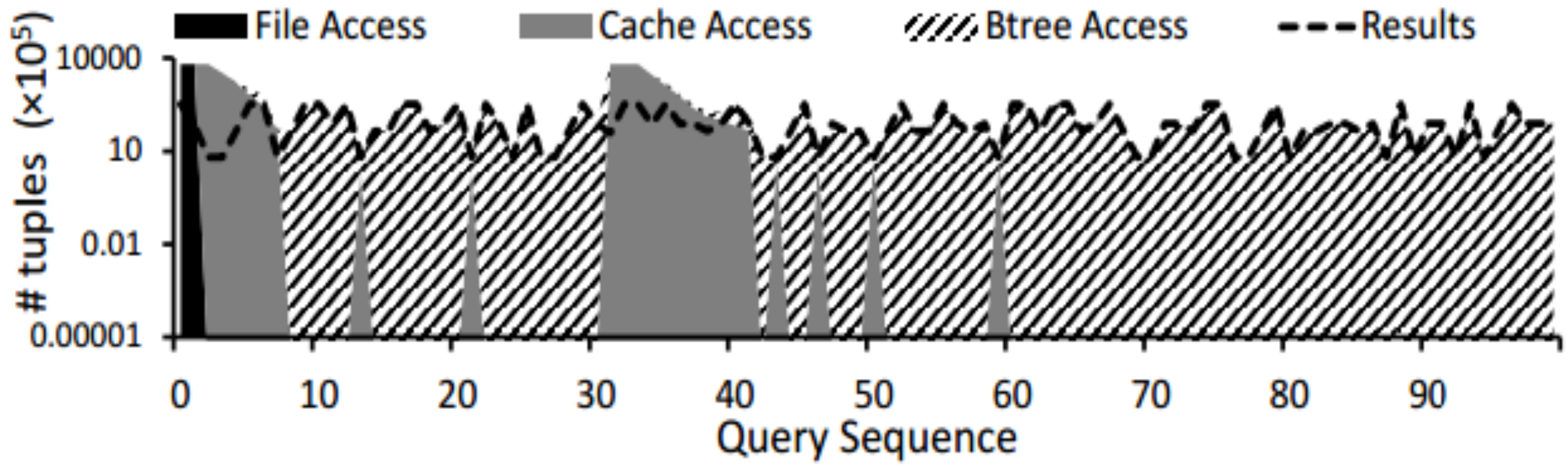


Figure 8: Sequence 100 queries. Number of accessed tuples using different access paths. Slalom uses indexes and data skipping to reduce data access.

The experiments shows the partitioning and indexing schemes of Slalom converge.

Experiments

- Figure 9 shows how the minimized data access translates to reduced response time and the efficiency of data skipping and indexing for different data distribution and different query types, which tested on the workload same as Figure 4.

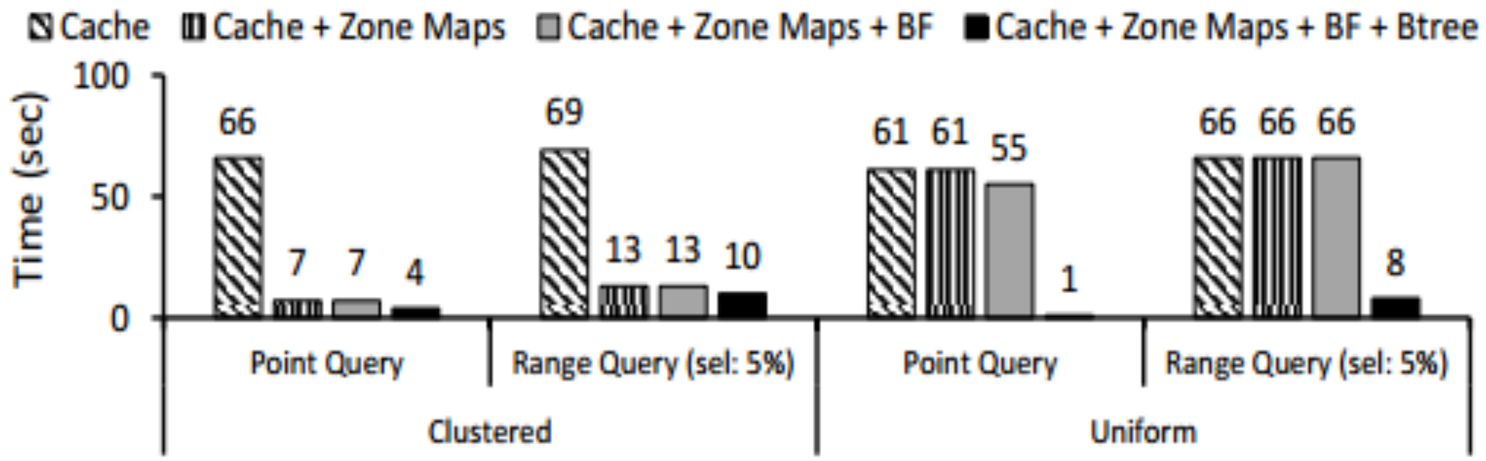


Figure 9: The effect of different indexes on point and range queries over uniform and clustered datasets.

Slalom avoids full access of the partition
slalom also reduces memory access or disk I/O if the partition is cached or not respectively.

Experiments

- Figure 10 shows the query execution times of Slalom for the workload given the three different memory budgets, consider working under memory constraints.

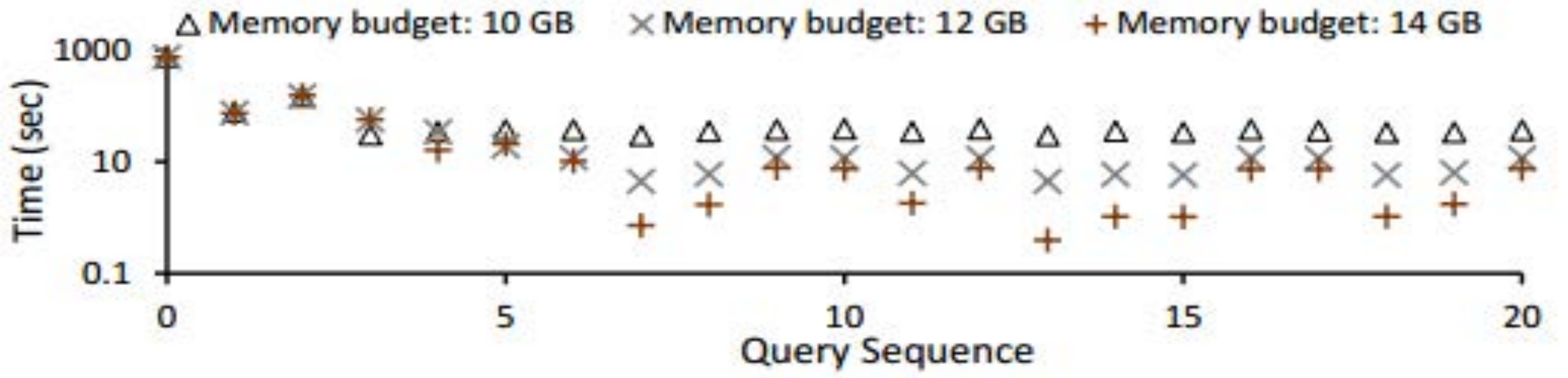


Figure 10: Slalom performance using different memory budgets. Slalom performance varies with alloted memory.

The results shows that as the query sequences increase, future queries can benefit from the additional B+-trees, using the available memory budget.

Experiments

- Figure 11 presents the breakdown of memory allocation for the same query sequence when Slalom is given a 12GB memory budget.

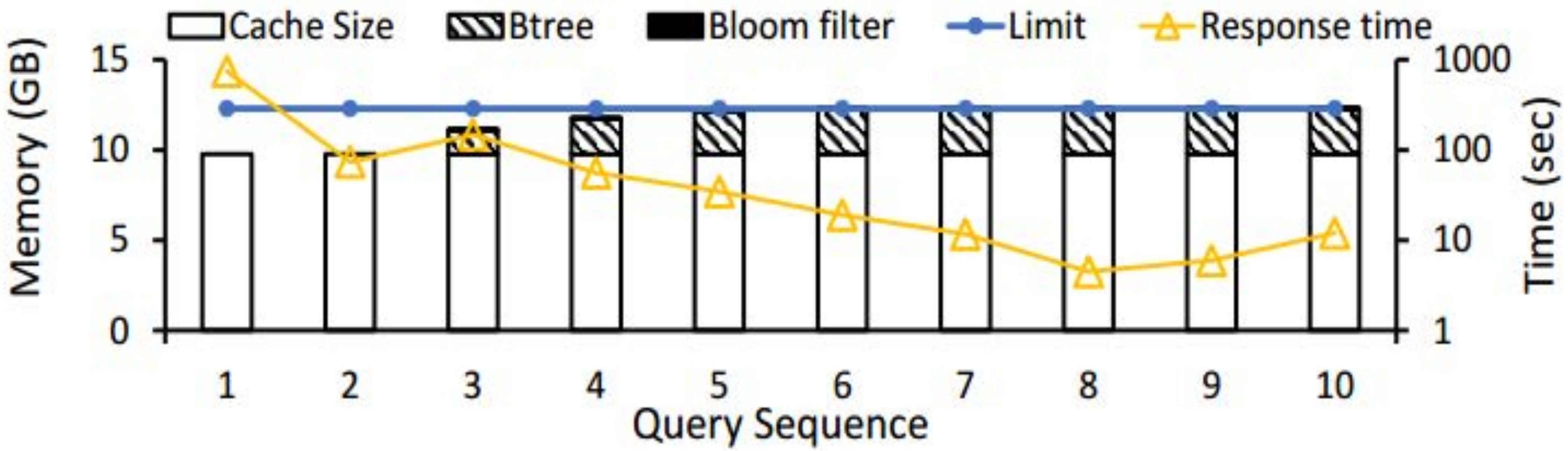
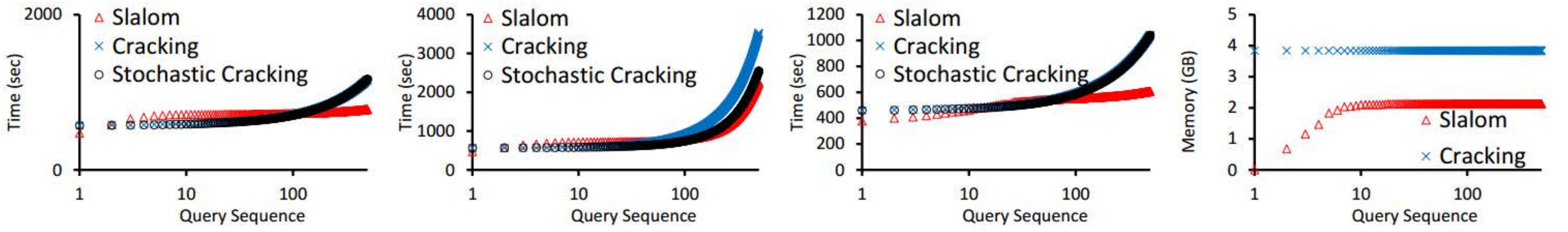


Figure 11: Slalom memory allocation (12 GB memory budget).

This experiment shows that Slalom can operate under limited memory budget gracefully managing the available resources to improve query execution performance.

Experiments

- Figure 12 shows the comparison between Slalom, Cracking and Stochastic Cracking.



(a) Random/Uniform data **(b)** Zoom In Alt./Uniform data **(c)** Random/Clustered data **(d)** Memory on Random/Uniform data

Figure 12: Cracking techniques converge more efficiently but Slalom takes advantage of data distribution.

Slalom & Cracking can be used in tandem

Conclusion

Pro

- Speed-up in situ query processing
- Online logical partitioning algorithm
- Low-overhead online fine-grained index selection
- Performance comparable to in-memory DBMS

Con

- Only compares the memory consumption on random workload over uniform data (Figure 12, adaptivity efficiency)
- The DBMS X with index performs better than Slalom during the most queries in the last experiment.