HINT on Steroids: Batch Query Processing for Interval Data

Panagiotis Bouros

Artur Titkov

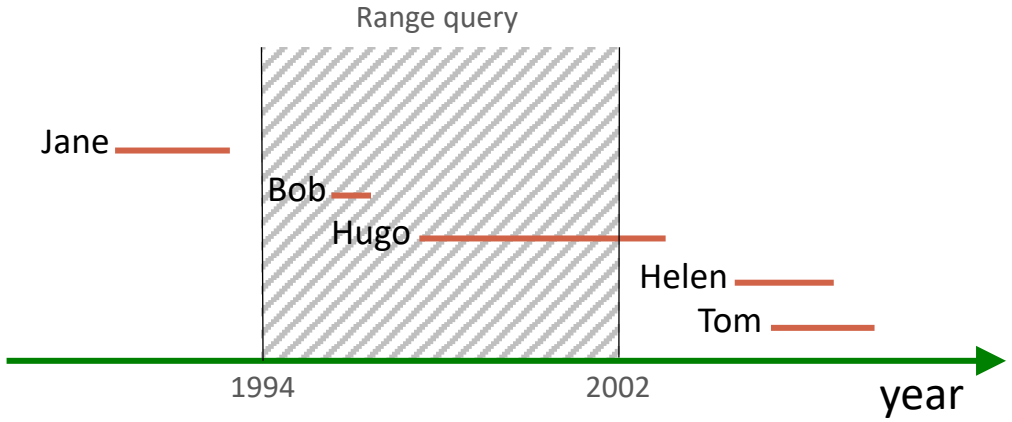
Christian Rauch

George Christodoulou

Nikos Mamoulis

Interval data

employee	start	end
Jane	1990	1993
Bob	1995	1996
Hugo	1997	2003
Helen	2005	2008
Tom	2006	2009

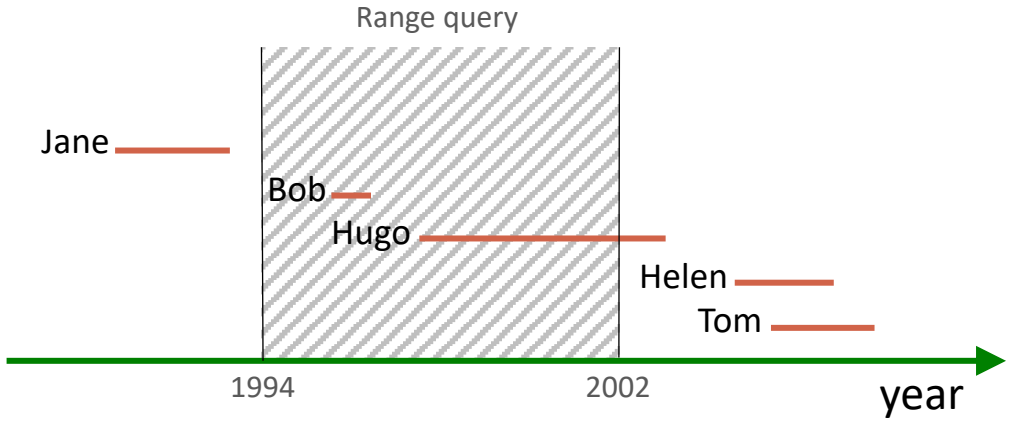


Range query

- Find all employees working at the company from 1994 to 2002

Interval data

employee	start	end
Jane	1990	1993
Bob	1995	1996
Hugo	1997	2003
Helen	2005	2008
Tom	2006	2009



Range query

- Find all employees working at the company from 1994 to 2002

Need for batch processing

Setting

- Query-heavy workflows
- Thousands or millions of incoming queries per second
- Modern OLTP systems and cloud services
  - Big IT companies, Amazon, Google etc
  - Amazon S3 receives 1M requests per second

Solution

- Process the queries in batches
  - Share and save resources
  - Reduce the overall time

Need for batch processing

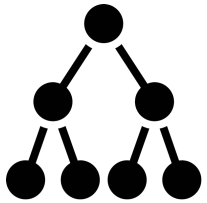
Setting

- Query-heavy workflows
- Thousands or millions of incoming queries per second
- Modern OLTP systems and cloud services
  - Big IT companies, Amazon, Google etc
  - Amazon S3 receives 1M requests per second

Solution

- Process the queries in batches
  - Share and save resources
  - Reduce the overall time

Background

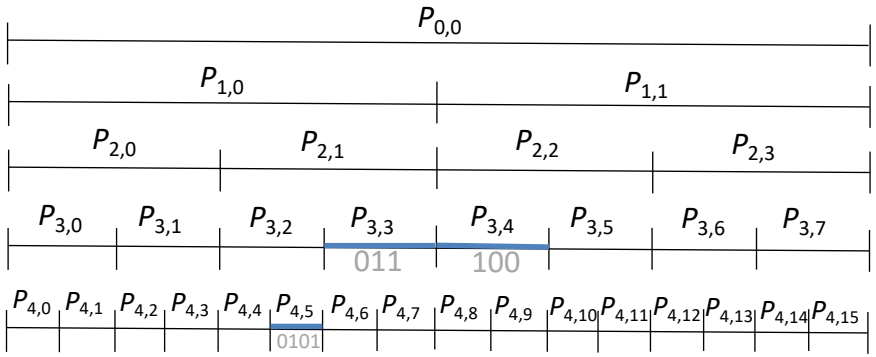


Indexing intervals

[G. Christodoulou, P. Bouros, N. Mamoulis, *HINT: A Hierarchical Index for Intervals in Main Memory*, ACM SIGMOD Conference 2022]
[G. Christodoulou, P. Bouros, N. Mamoulis, *HINT: a hierarchical interval index for Allen relationships*, VLDB Journal 33(1), 2024]

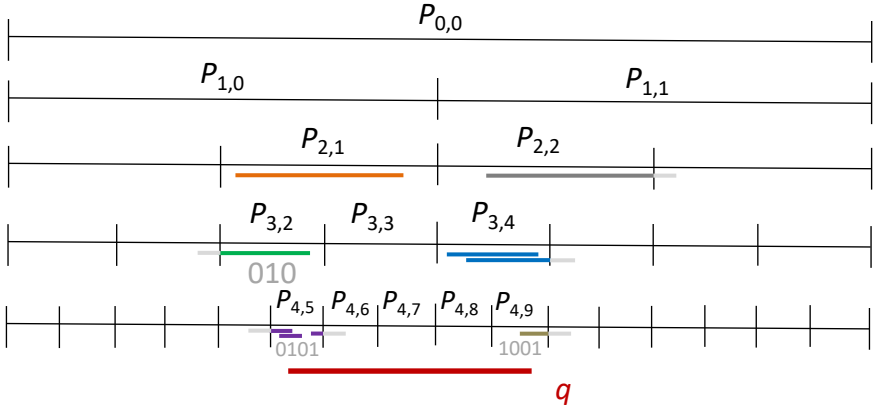
HINT: a hierarchical index for intervals

- Hierarchy of 1D grids
- Long intervals on high levels
- Intervals in at most 2 partitions per level



Query processing

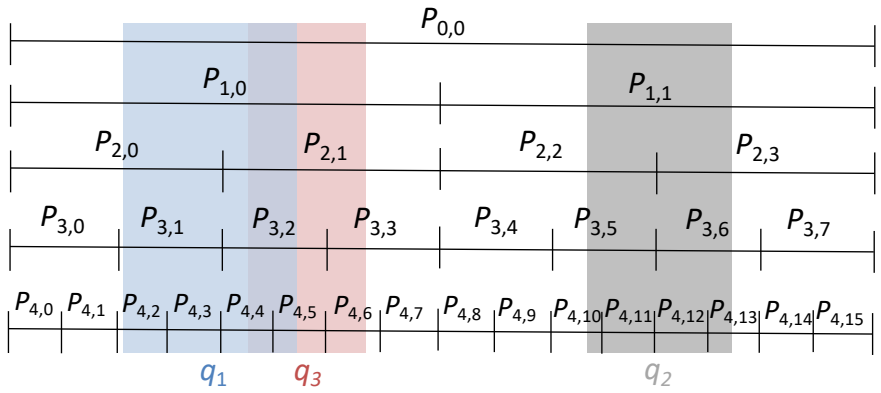
- Duplicate avoidance
- Bottom-up query processing
- Optimizations
  - Subdivisions P^{Oin}, P^{Oaft}, P^{Rin}, P^{Raft}
  - Space decomposition
  - Reduce cache misses
  - Deal with skewness & sparsity
- Allen algebra relationships



Batch processing challenges

Locality

- Avoid jumps in memory to access relevant partitions
- Horizontal jumps
  - Queries cover different parts of the index
  - Example, q_1, q_3 versus q_2
- Vertical jumps
  - Bottom-up traversal
  - Climb hierarchy for q_1, then for q_2, and lastly for q_3



Save resources

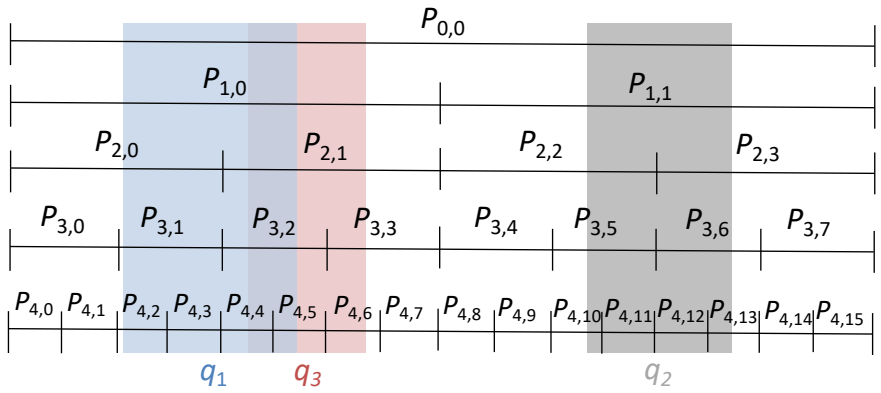
- Share computations among queries

Strategies

Query-based

Idea

- Evaluate queries in serial fashion
- Every query independently probs HINT, bottom-up



Pros

✓ Simple and straightforward

Cons

✗ Cache agnostic

✗ Horizontal and vertical jumps

Query-based

Idea

- Evaluate queries in serial fashion
- Every query independently probs HINT, bottom-up

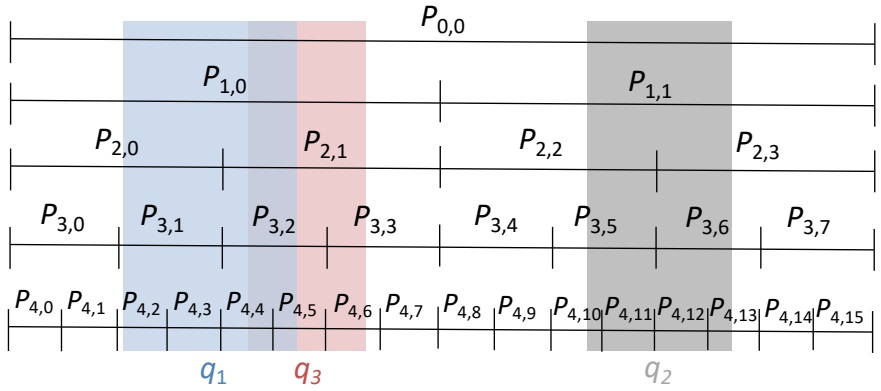
Pros

✓ Simple and straightforward

Cons

✗ Cache agnostic

✗ Horizontal and vertical jumps



Strategy	Accessed partitions
Query-based	$P_{4,2} \rightarrow P_{4,3} \rightarrow P_{4,4} \rightarrow P_{4,5} \rightarrow P_{3,1} \rightarrow P_{3,2} \rightarrow P_{2,0} \rightarrow P_{2,1} \rightarrow P_{1,0} \rightarrow P_{0,0} \rightarrow$ $P_{4,10} \rightarrow P_{4,11} \rightarrow P_{4,12} \rightarrow P_{4,13} \rightarrow P_{3,5} \rightarrow P_{3,6} \rightarrow P_{2,2} \rightarrow P_{2,3} \rightarrow P_{1,1} \rightarrow P_{0,0} \rightarrow$ $P_{4,4} \rightarrow P_{4,5} \rightarrow P_{4,6} \rightarrow P_{3,2} \rightarrow P_{3,3} \rightarrow P_{2,1} \rightarrow P_{1,0} \rightarrow P_{0,0}$

Query-based

Idea

- Evaluate queries in serial fashion
- Every query independently probs HINT, bottom-up

Pros

✓ Simple and straightforward

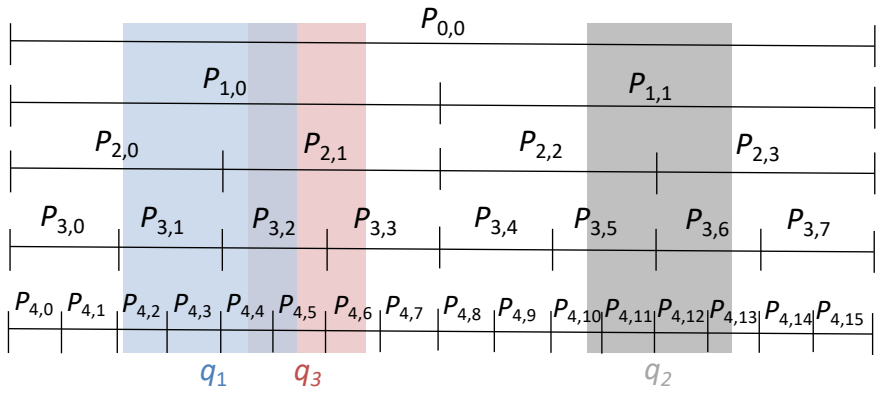
Cons

✗ Cache agnostic

✗ Horizontal and vertical jumps

Improvement

- Consume queries by their *start*

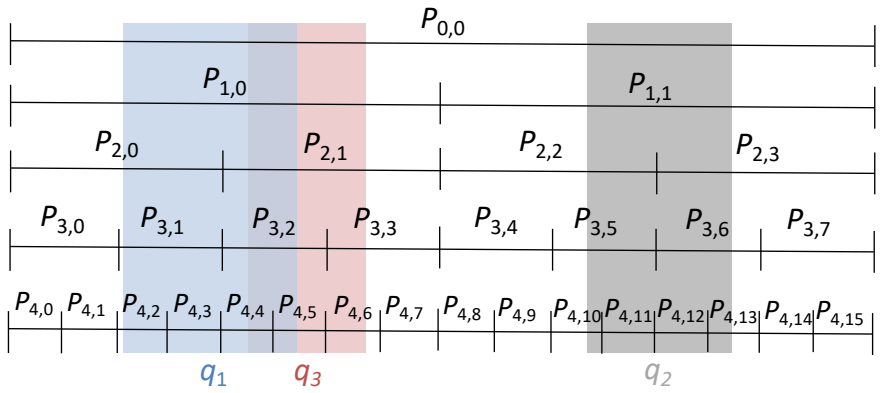


Strategy	Accessed partitions
Query-based	$P_{4,2} \rightarrow P_{4,3} \rightarrow P_{4,4} \rightarrow P_{4,5} \rightarrow P_{3,1} \rightarrow P_{3,2} \rightarrow P_{2,0} \rightarrow P_{2,1} \rightarrow P_{1,0} \rightarrow P_{0,0} \rightarrow$ $P_{4,10} \rightarrow P_{4,11} \rightarrow P_{4,12} \rightarrow P_{4,13} \rightarrow P_{3,5} \rightarrow P_{3,6} \rightarrow P_{2,2} \rightarrow P_{2,3} \rightarrow P_{1,1} \rightarrow P_{0,0} \rightarrow$ $P_{4,4} \rightarrow P_{4,5} \rightarrow P_{4,6} \rightarrow P_{3,2} \rightarrow P_{3,3} \rightarrow P_{2,1} \rightarrow P_{1,0} \rightarrow P_{0,0}$
Query-based with sorting	$P_{4,2} \rightarrow P_{4,3} \rightarrow P_{4,4} \rightarrow P_{4,5} \rightarrow P_{3,1} \rightarrow P_{3,2} \rightarrow P_{2,0} \rightarrow P_{2,1} \rightarrow P_{1,0} \rightarrow P_{0,0} \rightarrow$ $P_{4,4} \rightarrow P_{4,5} \rightarrow P_{4,6} \rightarrow P_{3,2} \rightarrow P_{3,3} \rightarrow P_{2,1} \rightarrow P_{1,0} \rightarrow P_{0,0} \rightarrow$ $P_{4,10} \rightarrow P_{4,11} \rightarrow P_{4,12} \rightarrow P_{4,13} \rightarrow P_{3,5} \rightarrow P_{3,6} \rightarrow P_{2,2} \rightarrow P_{2,3} \rightarrow P_{1,1} \rightarrow P_{0,0}$

Level-based

Idea

- Consume queries by their *start*
- Operate in a per-level fashion
  - Access all relevant partitions on a level
  - Move to the next



Pros

✓ Cache-aware

✓ Vertical jumps

Cons

✗ Horizontal jumps

Level-based

Idea

- Consume queries by their *start*
- Operate in a per-level fashion
  - Access all relevant partitions on a level
  - Move to the next

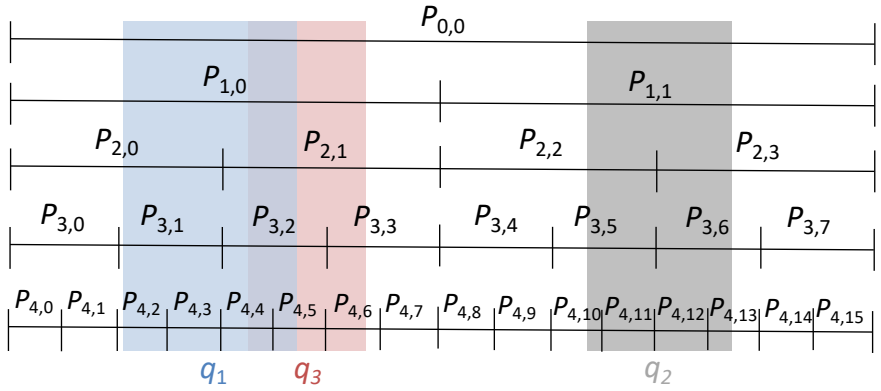
Pros

✓ Cache-aware

✓ Vertical jumps

Cons

✗ Horizontal jumps



Strategy	Accessed partitions
Query-based	$P_{4,2} \rightarrow P_{4,3} \rightarrow P_{4,4} \rightarrow P_{4,5} \rightarrow P_{3,1} \rightarrow P_{3,2} \rightarrow P_{2,0} \rightarrow P_{2,1} \rightarrow P_{1,0} \rightarrow P_{0,0} \rightarrow$ $P_{4,10} \rightarrow P_{4,11} \rightarrow P_{4,12} \rightarrow P_{4,13} \rightarrow P_{3,5} \rightarrow P_{3,6} \rightarrow P_{2,2} \rightarrow P_{2,3} \rightarrow P_{1,1} \rightarrow P_{0,0} \rightarrow$ $P_{4,4} \rightarrow P_{4,5} \rightarrow P_{4,6} \rightarrow P_{3,2} \rightarrow P_{3,3} \rightarrow P_{2,1} \rightarrow P_{1,0} \rightarrow P_{0,0}$
Query-based with sorting	$P_{4,2} \rightarrow P_{4,3} \rightarrow P_{4,4} \rightarrow P_{4,5} \rightarrow P_{3,1} \rightarrow P_{3,2} \rightarrow P_{2,0} \rightarrow P_{2,1} \rightarrow P_{1,0} \rightarrow P_{0,0} \rightarrow$ $P_{4,4} \rightarrow P_{4,5} \rightarrow P_{4,6} \rightarrow P_{3,2} \rightarrow P_{3,3} \rightarrow P_{2,1} \rightarrow P_{1,0} \rightarrow P_{0,0} \rightarrow$ $P_{4,10} \rightarrow P_{4,11} \rightarrow P_{4,12} \rightarrow P_{4,13} \rightarrow P_{3,5} \rightarrow P_{3,6} \rightarrow P_{2,2} \rightarrow P_{2,3} \rightarrow P_{1,1} \rightarrow P_{0,0}$

Level-based

Idea

- Consume queries by their *start*
- Operate in a per-level fashion
  - Access all relevant partitions on a level
  - Move to the next

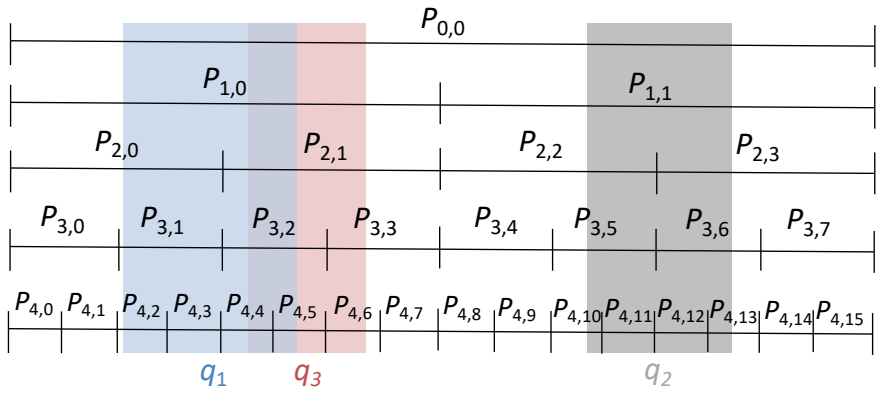
Pros

✓ Cache-aware

✓ Vertical jumps

Cons

✗ Horizontal jumps

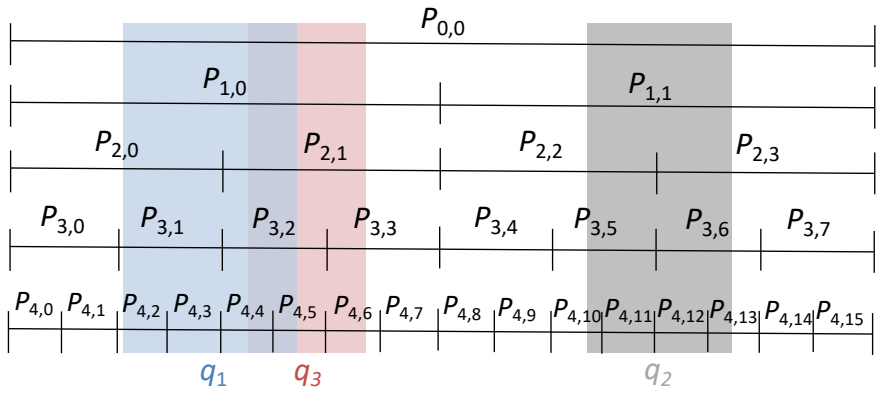


Strategy	Accessed partitions
Query-based	$P_{4,2} \rightarrow P_{4,3} \rightarrow P_{4,4} \rightarrow P_{4,5} \rightarrow P_{3,1} \rightarrow P_{3,2} \rightarrow P_{2,0} \rightarrow P_{2,1} \rightarrow P_{1,0} \rightarrow P_{0,0} \rightarrow$ $P_{4,10} \rightarrow P_{4,11} \rightarrow P_{4,12} \rightarrow P_{4,13} \rightarrow P_{3,5} \rightarrow P_{3,6} \rightarrow P_{2,2} \rightarrow P_{2,3} \rightarrow P_{1,1} \rightarrow P_{0,0} \rightarrow$ $P_{4,4} \rightarrow P_{4,5} \rightarrow P_{4,6} \rightarrow P_{3,2} \rightarrow P_{3,3} \rightarrow P_{2,1} \rightarrow P_{1,0} \rightarrow P_{0,0}$
Query-based with sorting	$P_{4,2} \rightarrow P_{4,3} \rightarrow P_{4,4} \rightarrow P_{4,5} \rightarrow P_{3,1} \rightarrow P_{3,2} \rightarrow P_{2,0} \rightarrow P_{2,1} \rightarrow P_{1,0} \rightarrow P_{0,0} \rightarrow$ $P_{4,4} \rightarrow P_{4,5} \rightarrow P_{4,6} \rightarrow P_{3,2} \rightarrow P_{3,3} \rightarrow P_{2,1} \rightarrow P_{1,0} \rightarrow P_{0,0} \rightarrow$ $P_{4,10} \rightarrow P_{4,11} \rightarrow P_{4,12} \rightarrow P_{4,13} \rightarrow P_{3,5} \rightarrow P_{3,6} \rightarrow P_{2,2} \rightarrow P_{2,3} \rightarrow P_{1,1} \rightarrow P_{0,0}$
Level-based with sorting	$P_{4,2} \rightarrow P_{4,3} \rightarrow P_{4,4} \rightarrow P_{4,5} \rightarrow P_{4,4} \rightarrow P_{4,5} \rightarrow P_{4,6} \rightarrow P_{4,10} \rightarrow P_{4,11} \rightarrow P_{4,12} \rightarrow P_{4,13} \rightarrow$ $P_{3,1} \rightarrow P_{3,2} \rightarrow P_{3,2} \rightarrow P_{3,3} \rightarrow P_{3,5} \rightarrow P_{3,6} \rightarrow$ $P_{2,0} \rightarrow P_{2,1} \rightarrow P_{2,1} \rightarrow P_{2,2} \rightarrow P_{2,3} \rightarrow$ $P_{1,0} \rightarrow P_{1,0} \rightarrow P_{1,1} \rightarrow$ $P_{0,0} \rightarrow P_{0,0} \rightarrow P_{0,0}$

Partition-based

Idea

- Consume queries by their *start*
- Operate in a per-level fashion
  - Access all relevant partitions on a level
  - Move to the next
- Evaluate all queries for a partition before moving to next



Pros

✓ Cache-aware

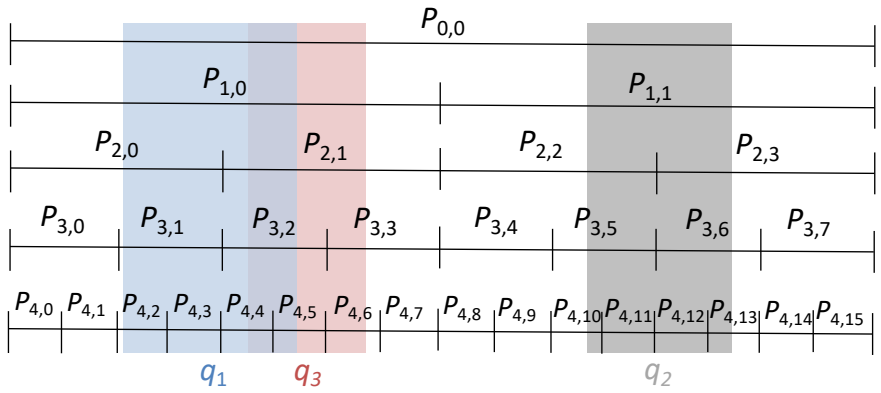
✓ Horizontal jumps

✓ Vertical jumps

Partition-based

Idea

- Consume queries by their *start*
- Operate in a per-level fashion
  - Access all relevant partitions on a level
  - Move to the next
- Evaluate all queries for a partition before moving to next



Pros

✓ Cache-aware

✓ Horizontal jumps

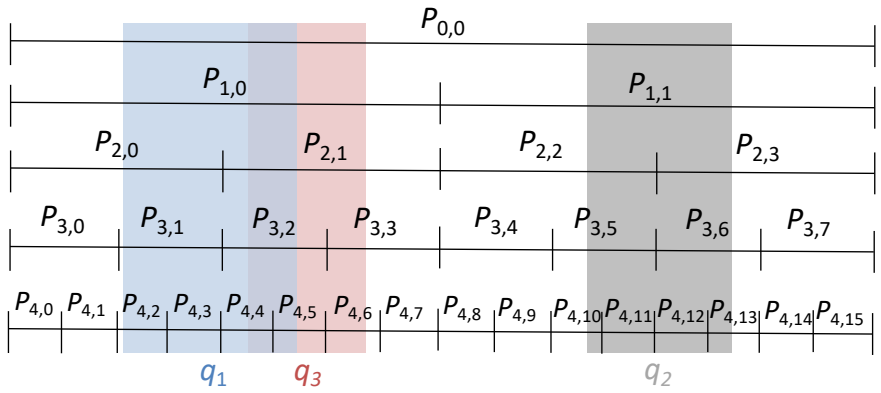
✓ Vertical jumps

Strategy	Accessed partitions
Query-based	$P_{4,2} \rightarrow P_{4,3} \rightarrow P_{4,4} \rightarrow P_{4,5} \rightarrow P_{3,1} \rightarrow P_{3,2} \rightarrow P_{2,0} \rightarrow P_{2,1} \rightarrow P_{1,0} \rightarrow P_{0,0} \rightarrow$ $P_{4,10} \rightarrow P_{4,11} \rightarrow P_{4,12} \rightarrow P_{4,13} \rightarrow P_{3,5} \rightarrow P_{3,6} \rightarrow P_{2,2} \rightarrow P_{2,3} \rightarrow P_{1,1} \rightarrow P_{0,0} \rightarrow$ $P_{4,4} \rightarrow P_{4,5} \rightarrow P_{4,6} \rightarrow P_{3,2} \rightarrow P_{3,3} \rightarrow P_{2,1} \rightarrow P_{1,0} \rightarrow P_{0,0}$
Query-based with sorting	$P_{4,2} \rightarrow P_{4,3} \rightarrow P_{4,4} \rightarrow P_{4,5} \rightarrow P_{3,1} \rightarrow P_{3,2} \rightarrow P_{2,0} \rightarrow P_{2,1} \rightarrow P_{1,0} \rightarrow P_{0,0} \rightarrow$ $P_{4,4} \rightarrow P_{4,5} \rightarrow P_{4,6} \rightarrow P_{3,2} \rightarrow P_{3,3} \rightarrow P_{2,1} \rightarrow P_{1,0} \rightarrow P_{0,0} \rightarrow$ $P_{4,10} \rightarrow P_{4,11} \rightarrow P_{4,12} \rightarrow P_{4,13} \rightarrow P_{3,5} \rightarrow P_{3,6} \rightarrow P_{2,2} \rightarrow P_{2,3} \rightarrow P_{1,1} \rightarrow P_{0,0}$
Level-based with sorting	$P_{4,2} \rightarrow P_{4,3} \rightarrow P_{4,4} \rightarrow P_{4,5} \rightarrow P_{4,4} \rightarrow P_{4,5} \rightarrow P_{4,6} \rightarrow P_{4,10} \rightarrow P_{4,11} \rightarrow P_{4,12} \rightarrow P_{4,13} \rightarrow$ $P_{3,1} \rightarrow P_{3,2} \rightarrow P_{3,2} \rightarrow P_{3,3} \rightarrow P_{3,5} \rightarrow P_{3,6} \rightarrow$ $P_{2,0} \rightarrow P_{2,1} \rightarrow P_{2,1} \rightarrow P_{2,2} \rightarrow P_{2,3} \rightarrow$ $P_{1,0} \rightarrow P_{1,0} \rightarrow P_{1,1} \rightarrow$ $P_{0,0} \rightarrow P_{0,0} \rightarrow P_{0,0}$

Partition-based

Idea

- Consume queries by their *start*
- Operate in a per-level fashion
  - Access all relevant partitions on a level
  - Move to the next
- Evaluate all queries for a partition before moving to next



Pros

✓ Cache-aware

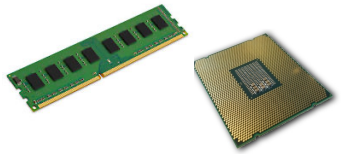
✓ Horizontal jumps

✓ Vertical jumps

Strategy	Accessed partitions
Query-based	$P_{4,2} \rightarrow P_{4,3} \rightarrow P_{4,4} \rightarrow P_{4,5} \rightarrow P_{3,1} \rightarrow P_{3,2} \rightarrow P_{2,0} \rightarrow P_{2,1} \rightarrow P_{1,0} \rightarrow P_{0,0} \rightarrow$ $P_{4,10} \rightarrow P_{4,11} \rightarrow P_{4,12} \rightarrow P_{4,13} \rightarrow P_{3,5} \rightarrow P_{3,6} \rightarrow P_{2,2} \rightarrow P_{2,3} \rightarrow P_{1,1} \rightarrow P_{0,0} \rightarrow$ $P_{4,4} \rightarrow P_{4,5} \rightarrow P_{4,6} \rightarrow P_{3,2} \rightarrow P_{3,3} \rightarrow P_{2,1} \rightarrow P_{1,0} \rightarrow P_{0,0}$
Query-based with sorting	$P_{4,2} \rightarrow P_{4,3} \rightarrow P_{4,4} \rightarrow P_{4,5} \rightarrow P_{3,1} \rightarrow P_{3,2} \rightarrow P_{2,0} \rightarrow P_{2,1} \rightarrow P_{1,0} \rightarrow P_{0,0} \rightarrow$ $P_{4,4} \rightarrow P_{4,5} \rightarrow P_{4,6} \rightarrow P_{3,2} \rightarrow P_{3,3} \rightarrow P_{2,1} \rightarrow P_{1,0} \rightarrow P_{0,0} \rightarrow$ $P_{4,10} \rightarrow P_{4,11} \rightarrow P_{4,12} \rightarrow P_{4,13} \rightarrow P_{3,5} \rightarrow P_{3,6} \rightarrow P_{2,2} \rightarrow P_{2,3} \rightarrow P_{1,1} \rightarrow P_{0,0}$
Level-based with sorting	$P_{4,2} \rightarrow P_{4,3} \rightarrow P_{4,4} \rightarrow P_{4,5} \rightarrow P_{4,4} \rightarrow P_{4,5} \rightarrow P_{4,6} \rightarrow P_{4,10} \rightarrow P_{4,11} \rightarrow P_{4,12} \rightarrow P_{4,13} \rightarrow$ $P_{3,1} \rightarrow P_{3,2} \rightarrow P_{3,2} \rightarrow P_{3,3} \rightarrow P_{3,5} \rightarrow P_{3,6} \rightarrow$ $P_{2,0} \rightarrow P_{2,1} \rightarrow P_{2,1} \rightarrow P_{2,2} \rightarrow P_{2,3} \rightarrow$ $P_{1,0} \rightarrow P_{1,0} \rightarrow P_{1,1} \rightarrow$ $P_{0,0} \rightarrow P_{0,0} \rightarrow P_{0,0}$
Partition-based with sorting	$P_{4,2} \rightarrow P_{4,3} \rightarrow P_{4,4} \rightarrow P_{4,4} \rightarrow P_{4,5} \rightarrow P_{4,5} \rightarrow P_{4,6} \rightarrow P_{4,10} \rightarrow P_{4,11} \rightarrow P_{4,12} \rightarrow P_{4,13} \rightarrow$ $P_{3,1} \rightarrow P_{3,2} \rightarrow P_{3,2} \rightarrow P_{3,3} \rightarrow P_{3,5} \rightarrow P_{3,6} \rightarrow$ $P_{2,0} \rightarrow P_{2,1} \rightarrow P_{2,1} \rightarrow P_{2,2} \rightarrow P_{2,3} \rightarrow$ $P_{1,0} \rightarrow P_{1,0} \rightarrow P_{1,1} \rightarrow$ $P_{0,0} \rightarrow P_{0,0} \rightarrow P_{0,0}$

Experiments

Setup



Hardware

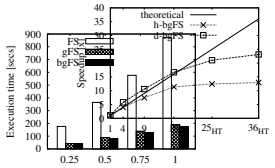
- Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20 GHz with 384 GBs of RAM running CentOS Linux

Software

- All strategies implemented in C++, compiled with -O3, -mavx and -march=native flags
- HINT variant with subdivisions, sorting, skewness & sparsity and cache misses optimizations

Datasets

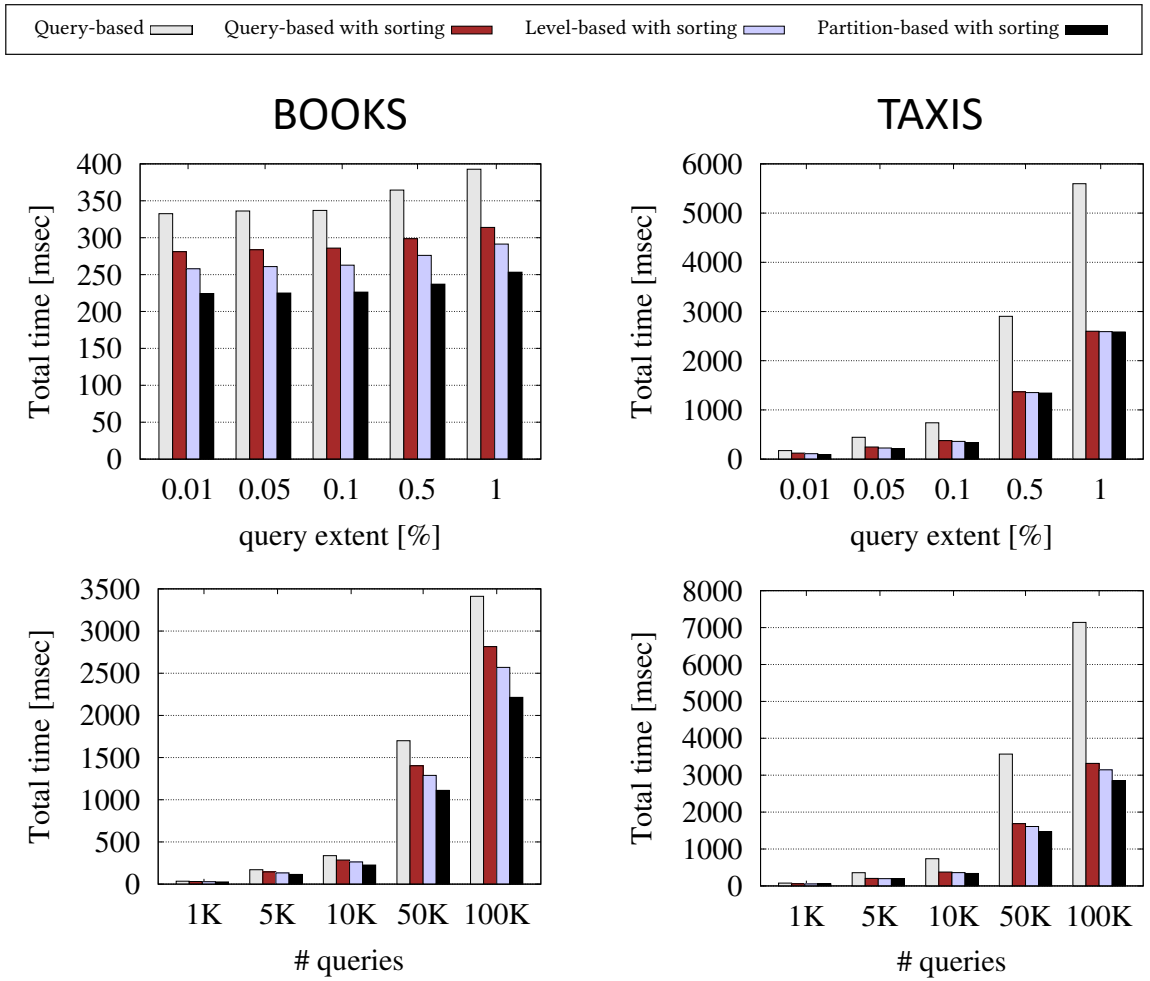
- BOOKS: periods of time book lent in Aarhus city libraries in 2013
- WEBKIT: periods of time file unchanged in git repository from 2001 to 2016
- TAXIS: period of taxi trips in New York City in 2013
- GREEND: power usage from households in Austria and Italy from 2010 to 2014
- Synthetic: interval duration follows exponential distribution, uniformly distributed *starts*



Experiments

- Total execution time of query batch
- Vary batch size (# queries) and query extent

Real datasets



1) Sorting queries reduces horizontal jumps
   - Query-based Vs Query-based with sorting

2) Level-based strategies outperform query-based
   - More pronounced in BOOKS, WEBKIT
   - Longer intervals

3) Partition-based the champion

To sum up…

Conclusions

- Studied batch processing for selection queries on intervals
- Proposed two processing strategies on top of state-of-the-art HINT
  - Operate on a per-level basis
  - Improve locality by eliminating jumps on HINT

Future work

- Investigate how to share and save computations
- Parellel batch processing

Thank you!

Questions ?

To download the source code and the datasets used, visit
https://github.com/pbour/batch_hint