Database Research Group

Sort-Based Parallel Loading of R-trees

Daniar Achakeev, Marc Seidemann, Markus Schmidt, Bernhard Seeger

Department of Mathematics and Computer Science

Philipps-Universität Marburg, Germany
achakeye@mathematik.uni-marburg.de

ACM SIGSPATIAL BigSpatial 2012

Agenda

- **Introduction**
- Sequential sort-based query-adaptive loading
  - Sorted-Set Partitioning
- Parallel Loading
  - MapReduce
- Results
- Conclusion

Database
Research Group

Motivation

- **R-tree**
  - Spatial and multidimensional data
- **Emerging applications**
  - Location Based Services
    - kNN, Reverse kNN, Spatial keyword search etc…
- **Tuple-by-Tuple loading is inefficient**
  - Trade-off loading time query efficiency
  - NP-hard
- **Parallelism**
  - modern hardware
  - low cost parallel architecture e.g. Hadoop

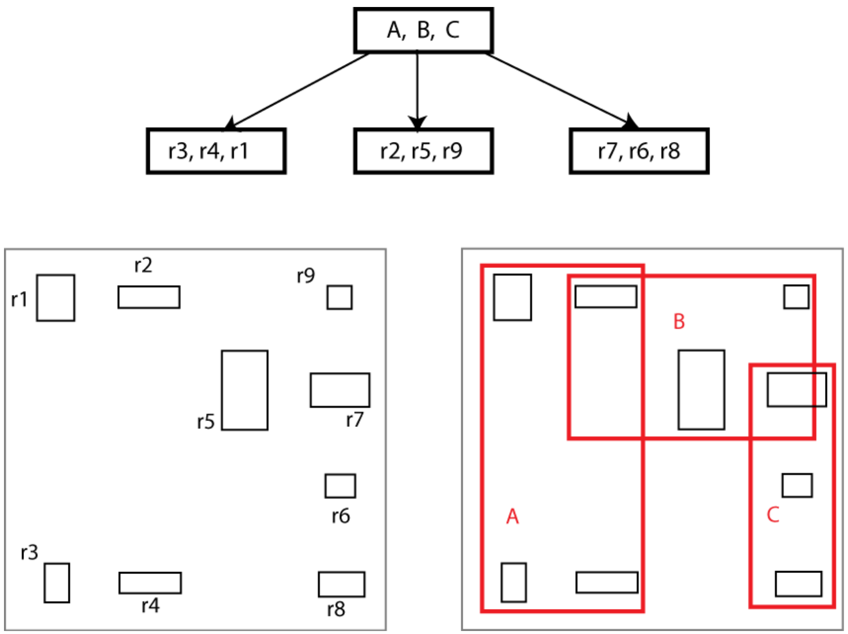
Database Research Group

Agenda

- Introduction
- **Sequential sort-based query-adaptive loading**
  - Sorted-Set Partitioning
- Parallel Loading
  - MapReduce
- Results
- Conclusion

Database Research Group

R-tree



I/O Model:

R-Tree nodes mapped to disk blocks

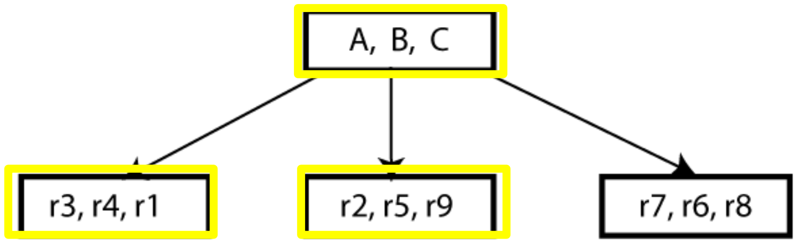
Maximal capacity: B

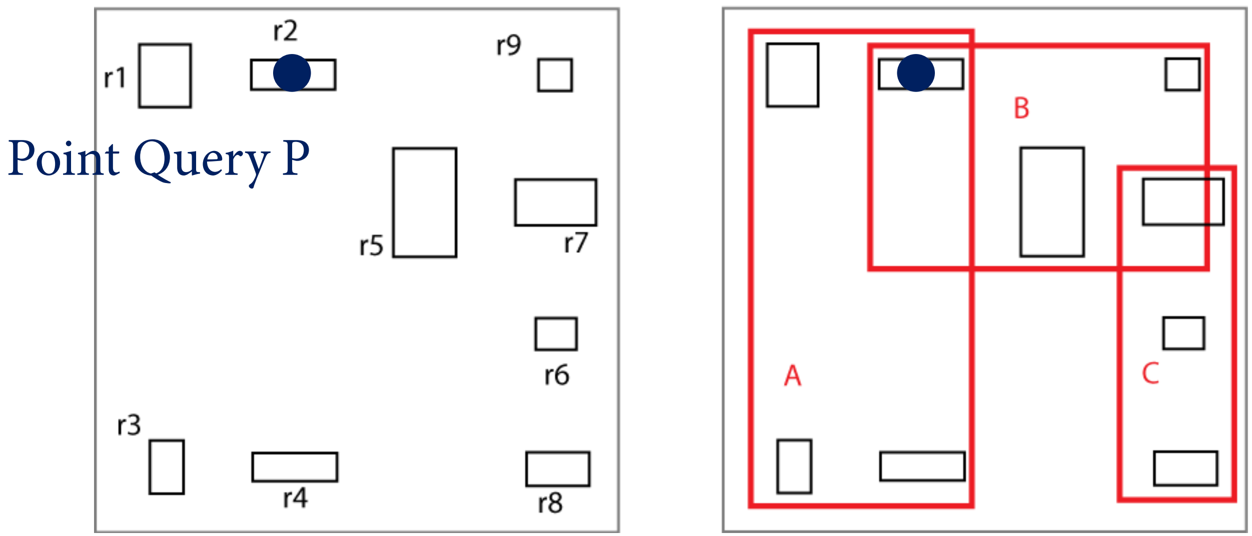
Minimal capacity: $b \leq \left\lceil \frac{B}{2} \right\rceil$

Minimal Bounding Rectangle MBR

Database Research Group

R-tree Query Types

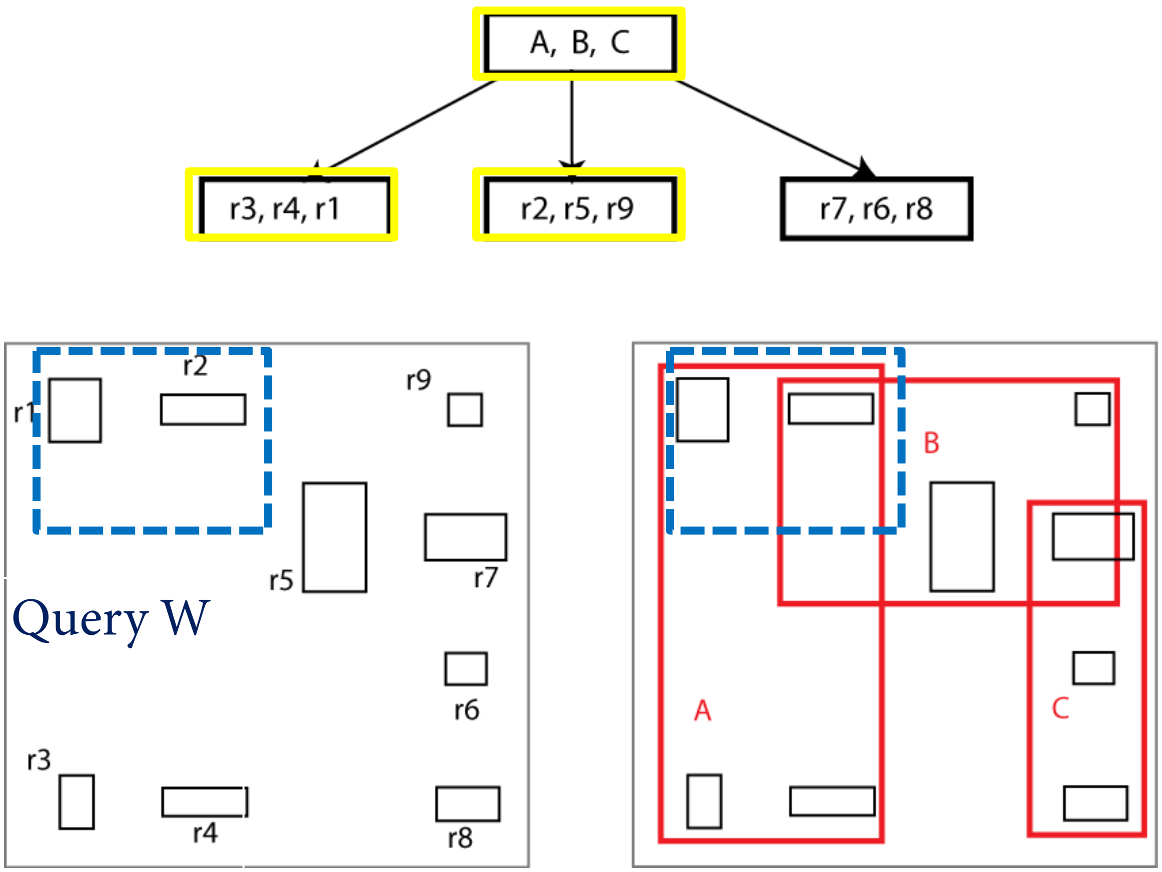




Goal: minimize node accesses!

Database Research Group

R-tree Query Types

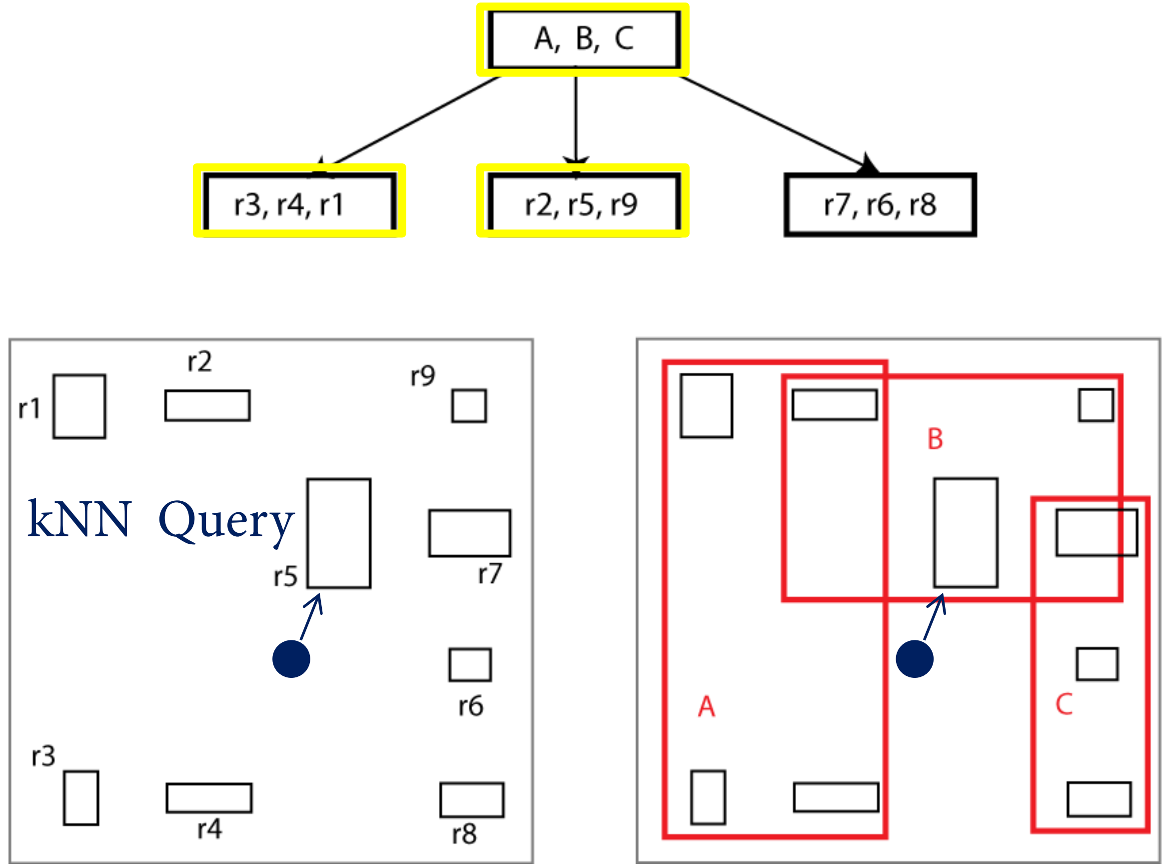


Window Query W

Goal: minimize node accesses!

Database Research Group

R-tree Query Types

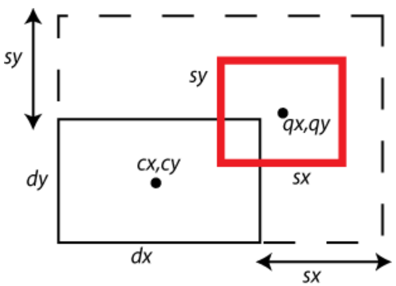


Goal: minimize node accesses!

Database Research Group

Cost Model

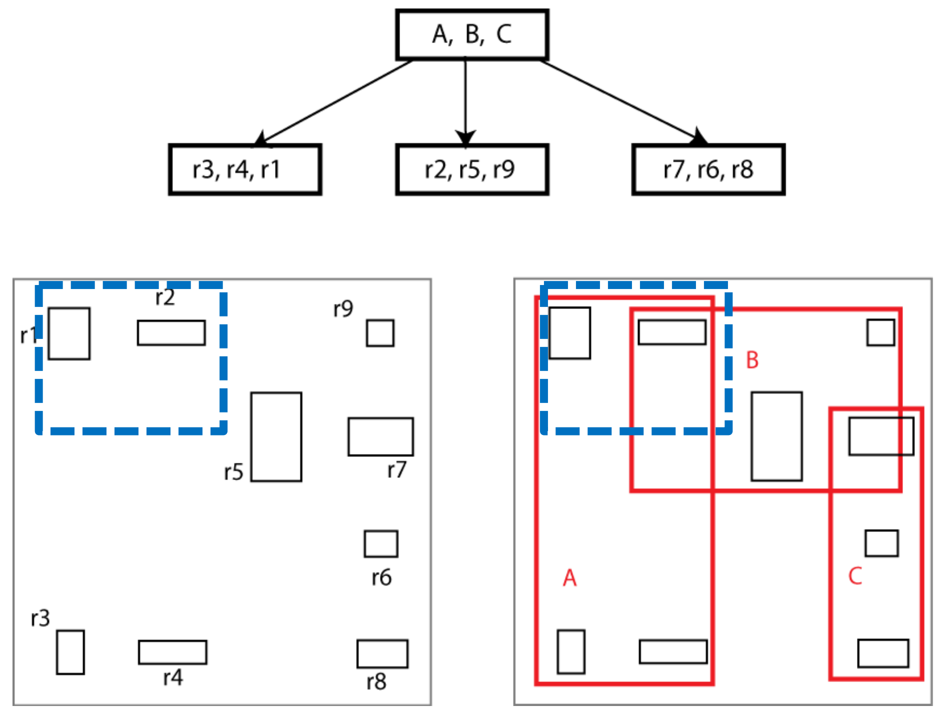
Minimize sum of areas of node MBRs ⇔ Minimize node accesses !



The average number of rectangles intersecting the query window

$$\sum_{i=1}^{N} (dx_i + sx) \cdot (dy_i + sy)$$

Query profile QP is given by (*sx,sy*)



I.Kamel and C. Faloutsos. On packing r-tress, In CIKM 1993

B.-U. Pagel et. al. Towards an analysis of range query performance in spatial data structures. In PODS 1993

Y. Theodoridis and T. Sellis A model for the prediction of r-tree performance. In PODS 1996

Objective

Rectangles

Queries

Minimal bounding rectangles (MBR) of R-tree leaf level

Database Research Group

Sort-based Query-Adaptive Loading [1]

- **NP-Hardness of optimal partitioning**
- **Conceptual Easy Heuristic Algorithm**
  - Sorting according Space Filling Curve
  - Dynamic Programming
  - Adaptive SFC
- **Excellent I/O performance**
  - I/O Complexity is bounded by external sort $O(\frac{N}{B} \cdot log_{\frac{M}{B}} \frac{N}{B})$
- **Experiments**
  - Non trivial test framework
  - Average better query performance
  - Robustness for different query and data distribution
- **Parallel Version**

[1] D. Achakeev, B Seeeger and P. Widmayer, Sort-Based Qeury Adaptive Loading of R-trees, in CIKM 2012

Database Research Group

Sort-based Query-Adaptive Loading

Bottom-Up and Level-by-Level

1. **Determination of Sort Order and Sorting:** For a given QP determine a sort order that minimizes cost.

- Quadratic queries (aspect ratio 1:1); Hilbert or Z-Curve
- Otherwise asymmetric Z-Curve

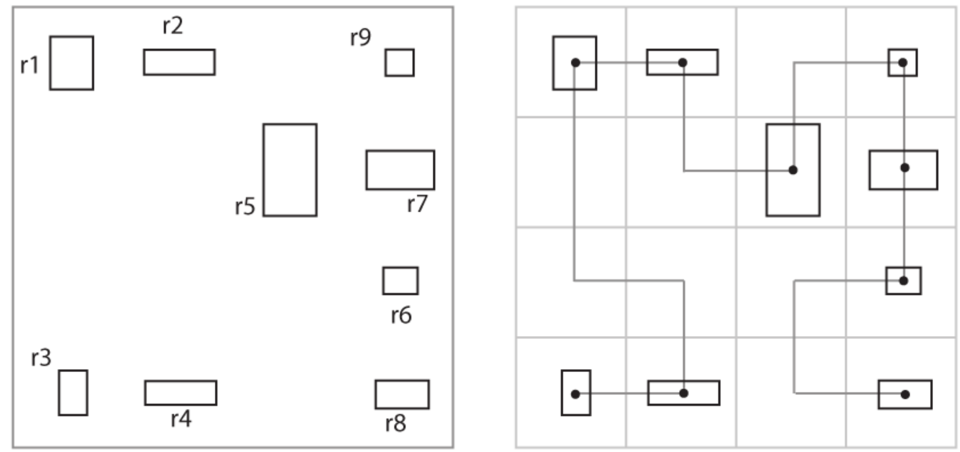
2. **Sorted set partitioning.** Partition the sorted sequence of rectangles into subsequences of size between minimal page capacity ***b*** and maximal page capacity ***B***

3. **Recursive Step:** Generation of index entries and recursion

Database Research Group

Sorted-Set-Partitioning

Example: b=2, B=3



The problem of optimal partitioning is *NP-hard*!

Idea:

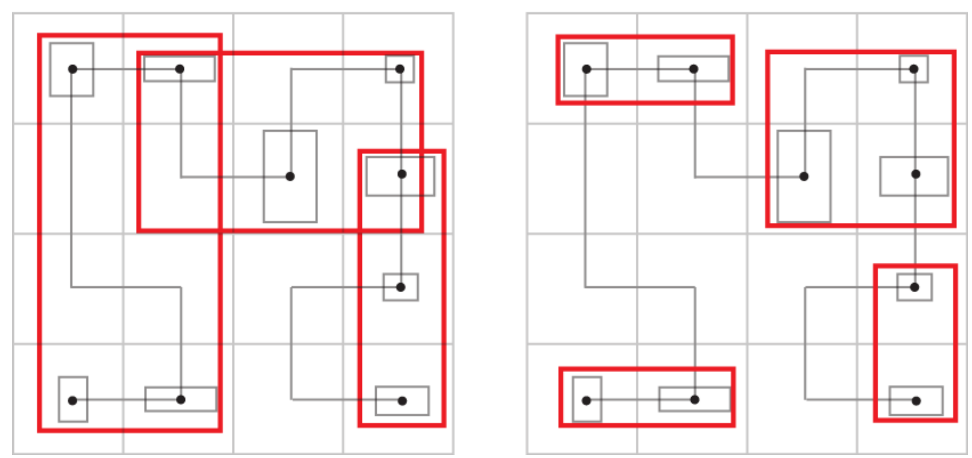
1. Space Filling Curves

2. Dynamic Programming

Hilbert: $(r3_1, r4_2, r1_3, r2_4, r5_5, r9_6, r7_7, r6_8, r8_9)$

Example:
Max page capacity B=3
Min page capacity b=2

Cost function:
Sum of MBR areas



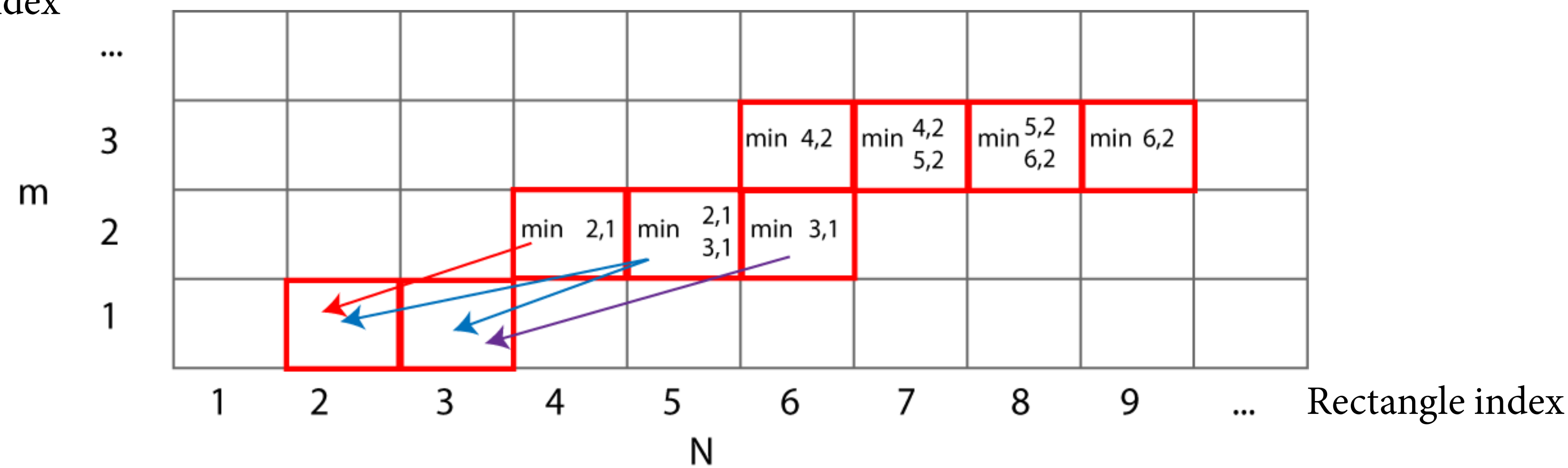
Standard approach **Our approach**

Storage-Bounded Partitioning

Dynamic Programming (DP)

Hilbert: $\{r3_1, r4_2, r1_3, r2_4, r5_5, r9_6, r7_7, r6_8, r8_9\}$ b=2,B=3

Page index



C[5][2]=min {C[2][1] + area_{QP}(MBR({3,4,5})), C[3][1] + area_{QP} (MBR({4,5}))}

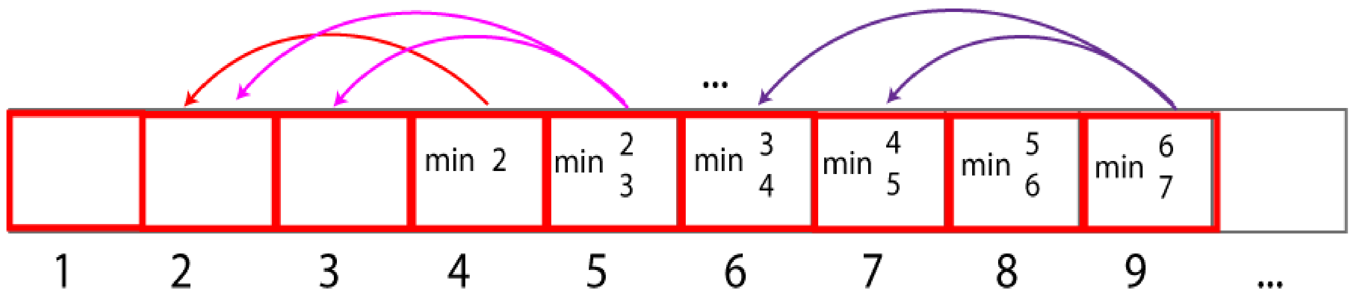
- V-Optimal Histograms
- $N/B \leq m \leq N/b$
- Quadratic time $O(N^2 \cdot B)$ and space $O(N^2)$

$$opt^*(i,k) = \min_{b \leq j \leq B} \{opt^*(i-j, k-1) + Area_{QP}(MBR(p_{i-j+1,i}))\}$$

Database Research Group

Query-Optimal Partitioning GOPT

Hilbert: ($r3_1$,$r4_2$,$r1_3$,$r2_4$,$r5_5$,$r9_6$,$r7_7$,$r6_8$,$r8_9$)



b=2,B=3

N

Sorted Rectangles

…C[6]=min {C[3] + area_{QP} (MBR({4,5,6})), C[4] + area_{QP} (MBR({5,6}))}

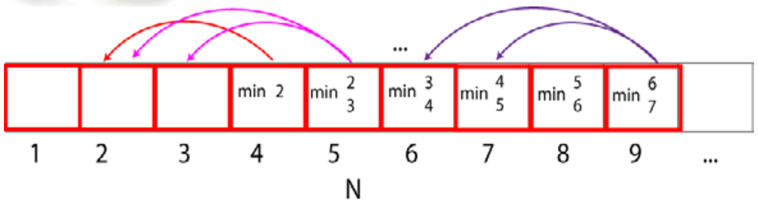
- Linear time $O(N \cdot B)$ and linear space $O(N)$
- Number of output partitions $\boldsymbol{m}$ is bounded by $N/B \leq m \leq N/b$

$$gopt^*(i) = \min_{b \leq j \leq B} \{gopt^*(i-j) + Area_{QP}(MBR(p_{i-j+1,i}))\}$$

- Generalized methods for all levels are also investigated

Database Research Group

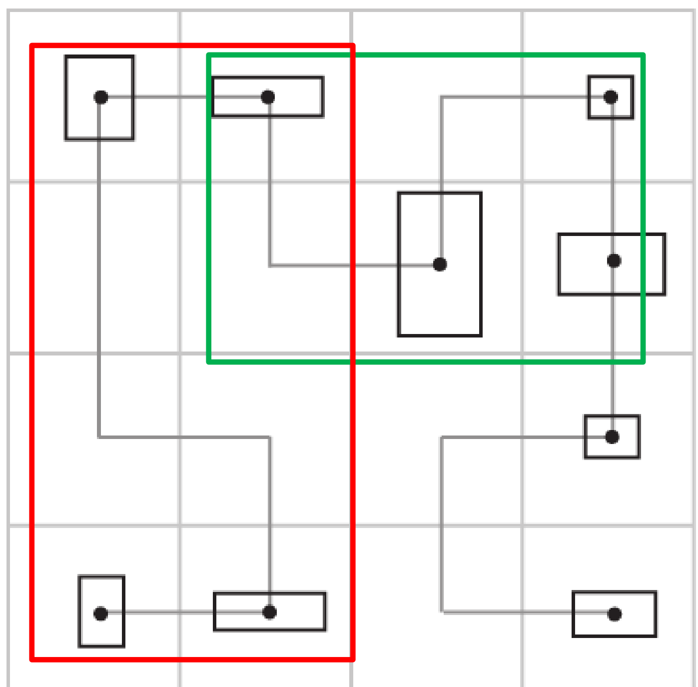
GOPT Example …



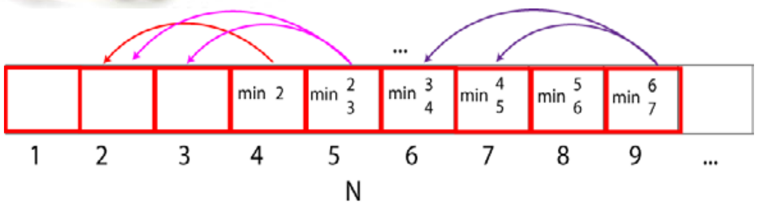
b=2,B=3

Hilbert: $(r3_1, r4_2, r1_3, r2_4, r5_5, r9_6, r7_7, r6_8, r8_9)$

$\ldots C[6] = \min\{\mathbf{C[3]} + area_{QP}(\mathbf{MBR(\{4,5,6\})}), C[4] + area_{QP}(MBR(\{5,6\}))\}$



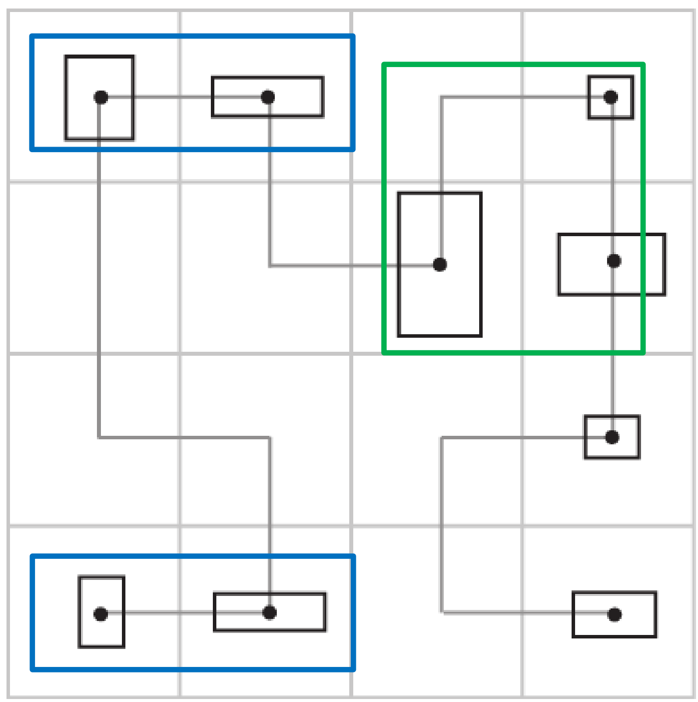
GOPT Example …



b=2,B=3

Hilbert: ($r3_1, r4_2, r1_3, r2_4, r5_5, r9_6, r7_7, r6_8, r8_9$)

…$C[6]=\min\{C[3] + area_{QP}(MBR(\{4,5,6\})),\ \mathbf{C[4]} + area_{QP}(\mathbf{MBR(\{5,6\})})\}$

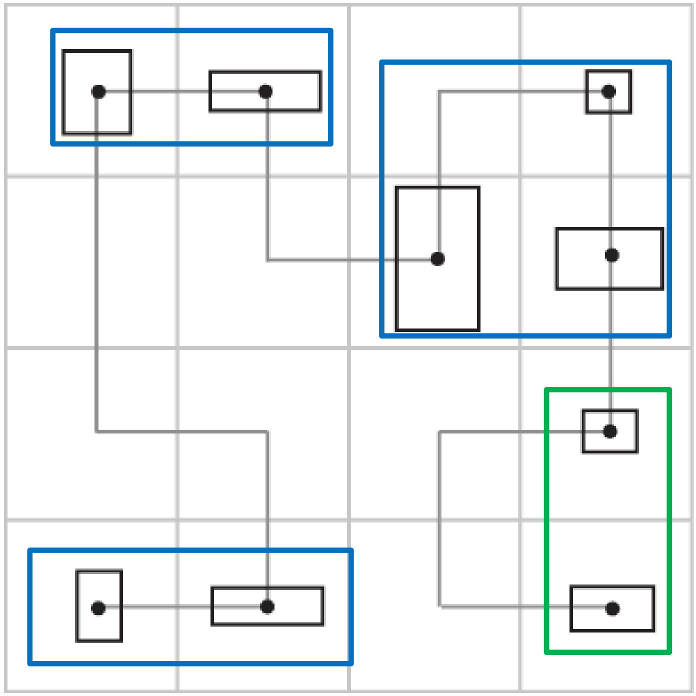


Database Research Group

GOPT Example …

Hilbert: ($r3_1$,$r4_2$,$r1_3$,$r2_4$,$r5_5$,$r9_6$,$r7_7$,$r6_8$,$r8_9$)

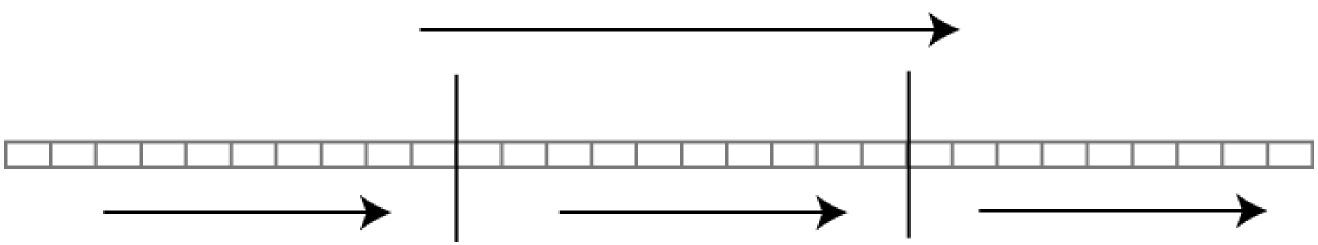
End Result



Database Research Group

Practical Considerations

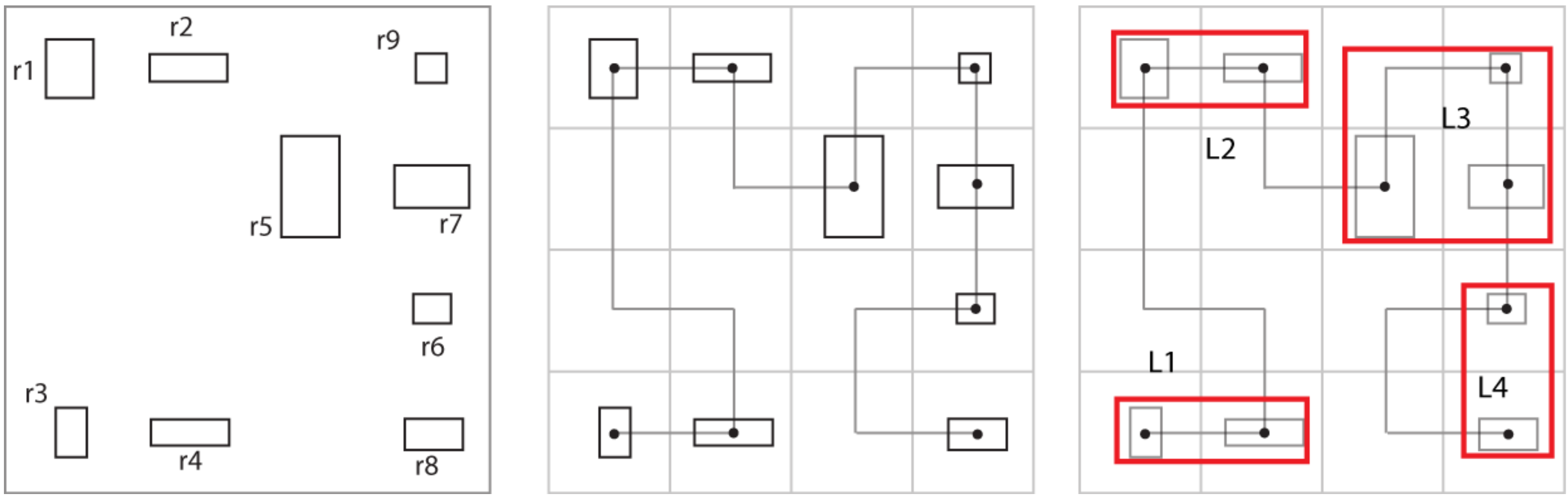
- **Reduce CPU and memory costs**
  - Use main memory efficiently
  - Simple heuristic: chunking



$\geq B^2$ $\geq B^2$ $\geq B^2$

Practical Considerations

- **Sort data only once for leaf level**
  - Use the sorting order of the produced output



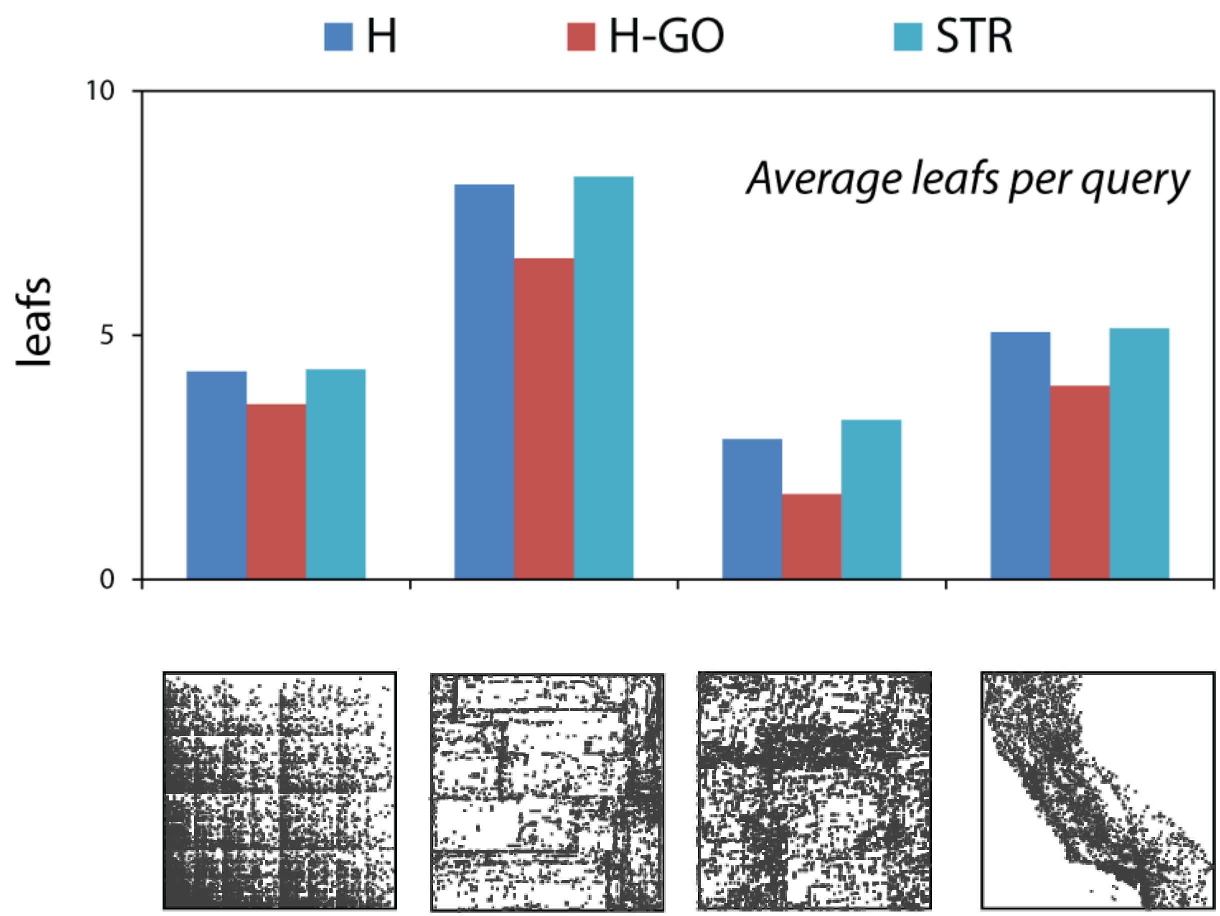
Sorted data: $\{r3_1, r4_2, r1_3, r2_4, r5_5, r9_6, r7_7, r6_8, r8_9\}$

Partitioning output: $L1 = \{r3_1, r4_2\}$, $L2 = \{r1_3, r2_4\}$, $L3 = \{r5_5, r9_6, r7_7\}$, $L4 = \{r6_8, r8_9\}$

Index Level: {L1,L2,L3,L4}

Database Research Group

Results



H: standard sort-based bulk loading, Hilbert-Order

H-GO: *our approach GOPT, Hilbert-Order*

STR: STR loading

4 KB Pages;

Queries follow data distribution.

The query size is defined by the number of results.

Database Research Group

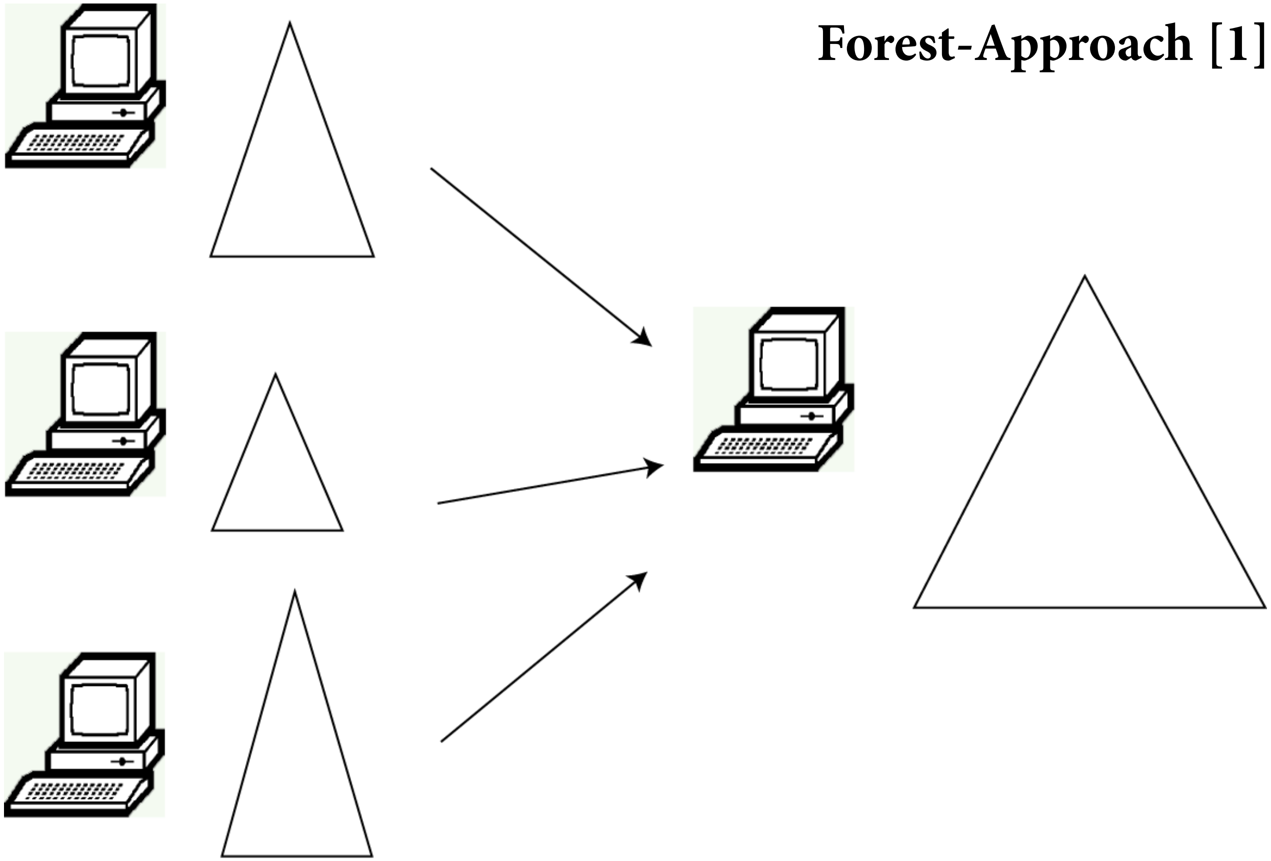
Agenda

- Introduction
- Sequential Loading
  - Sorted-Set Partitioning
- **Parallel Loading**
  - MapReduce
- Results
- Conclusion

Database Research Group

Introduction

Forest-Approach [1][2]

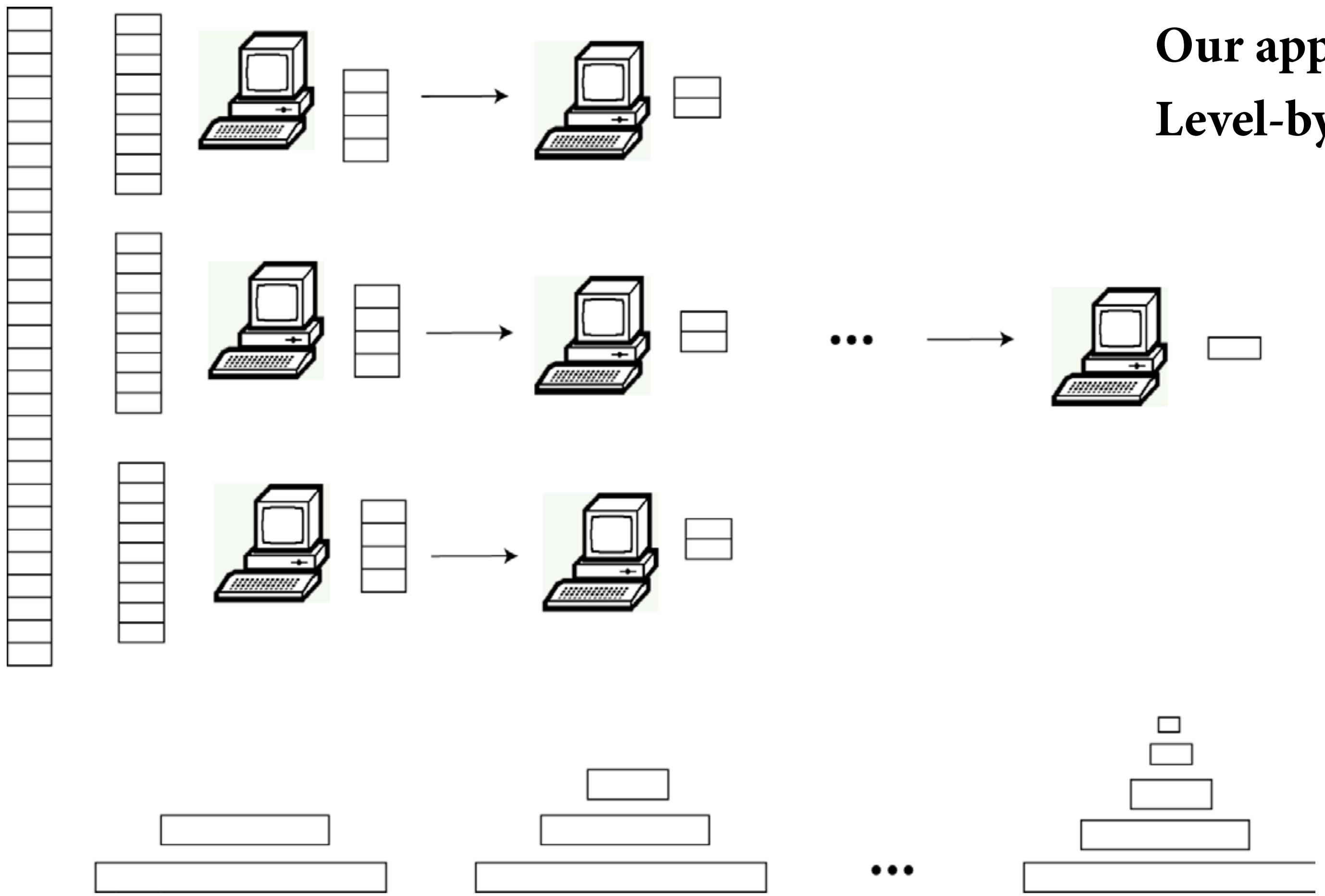


[1] A. Cary, Z. Sun, V. Hristidis, and N. Rishe. Experiences on processing spatial data with mapreduce, in SSDBM 2009

[2] A. Papadopoulos, and Y. Manolopoulos. Parallel bulk-loading of spatial data. In Parallel Comput. 2003

Database Research Group

Introduction



Our approach

Level-by-level [1]

[1] Daniar Achakeev, Marc Seidemann, Markus Schmidt, Bernhard Seeger: Sort-based Parallel Loading of R-trees, ACM SIGSPATIAL BigSpatial-2012;

Parallel Level-By-Level Loading

1. Computation of initial split vector V
2. Parallel sort using SFC
3. Data distribution over machines using V
4. Computation of optimal partitioning GOPT
5. Computation of split vector for the next level
6. Recursion: Step 3 using output of step 4

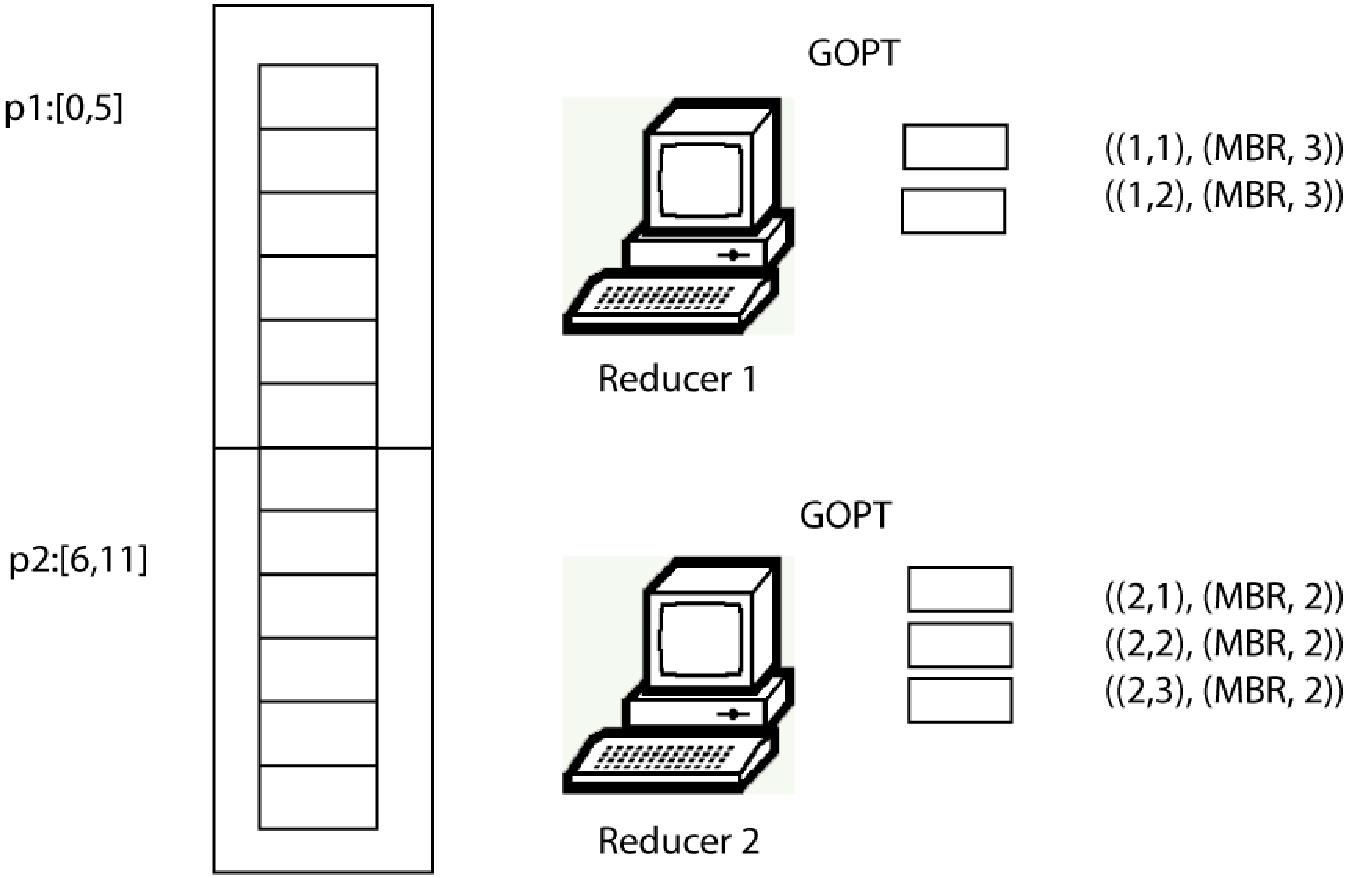
Database Research Group

MapReduce

- **Leaf node generation:**
  - Mapper
    [(null, MBRdata), …] -> [(SFC-Key, Data),…]
  - Split vector V [p_1, … , p_m] computed using parallel random sampling
  - Partitioner distributes data using V
  - Reducer runs gopt for its sorted key interval
    (SFC-Key, [MBRdata]) -> [((reducerRank, MBRRank), info), …]
  - Reducer Output:
    - Sorted input data
    - Leaf node MBR
      - Key (reducerRank, localRank)

Database Research Group

MapReduce


p1:[0,5]
p2:[6,11]
GOPT
Reducer 1
((1,1), (MBR, 3))
((1,2), (MBR, 3))
GOPT
Reducer 2
((2,1), (MBR, 2))
((2,2), (MBR, 2))
((2,3), (MBR, 2))

MapReduce

- **Index node generation**
  - Mapper Identity
  - Partitioner: lexicographical order (reducerRank, MBRRank)
  - Reducer runs GOPT on ((reducerRank, MBRRank), info) objects
- **Final R-tree**
  - level files in parallel
  - level file sequentially

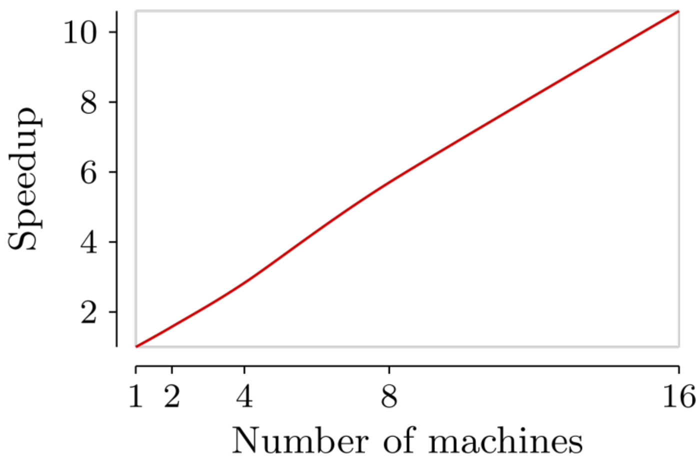
Database Research Group

Results

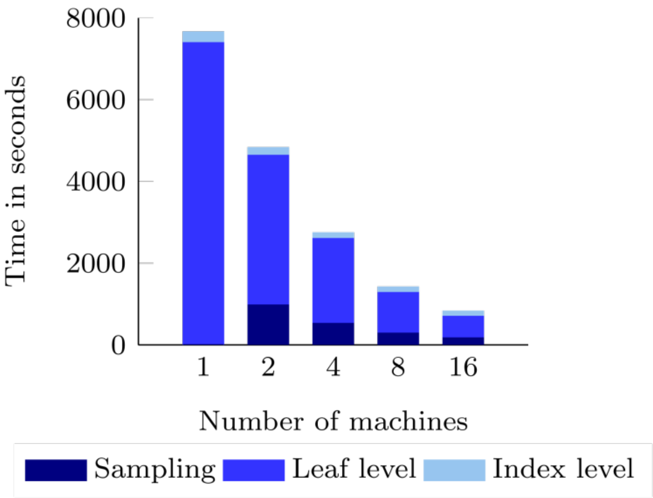
- **Settings:**
  - Java, Hadoop 0.20.205.0, XXL-Java-Library
  - Amazon: medium machines
  - Data set TIGER USA Streets 72M MBR approx. 3.6 GB
  - Extended TIGER USA approx. 13 GB
  - Machines (1),2,4,8,16 (+ 1 Jobtracker)
  - Random Sampling 3%

Database Research Group

Results

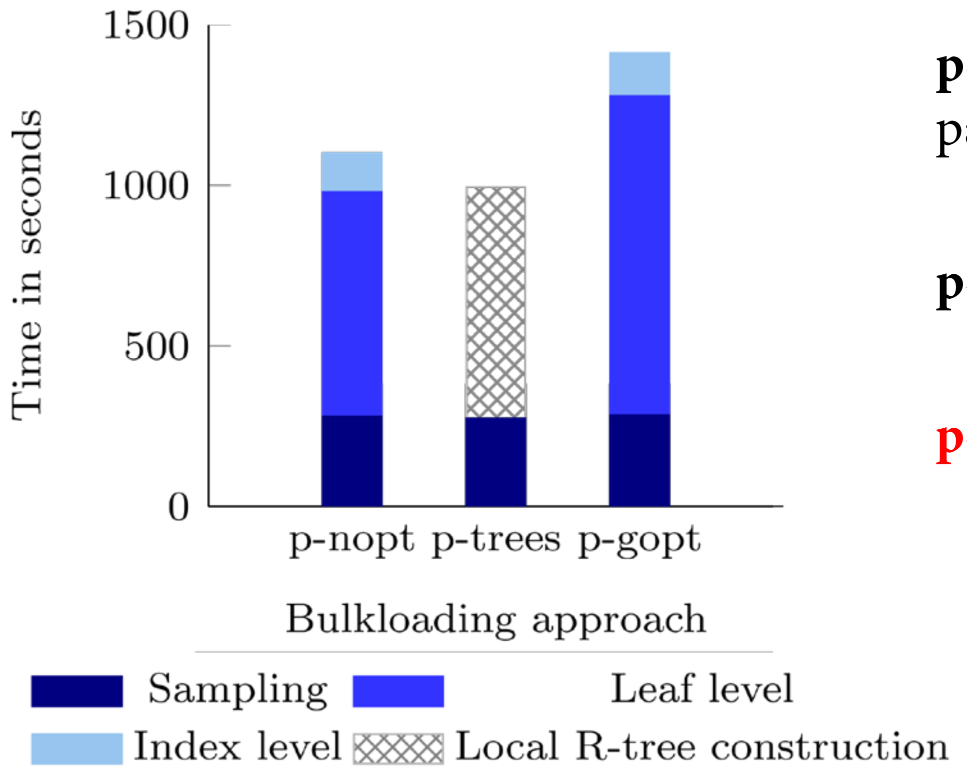


Speedup
10
8
6
4
2
1
2
4
8
16
Number of machines



Time in seconds
8000
6000
4000
2000
0
1
2
4
8
16
Number of machines
Sampling
Leaf level
Index level

Results



p-nopt: level-by-level, fixed-size partitioning

p-trees: forest approach [1]

p-gopt: out approach

[1] A. Cary, Z. Sun, V. Hristidis, and N. Rishe. Experiences on processing spatial data with mapreduce, in SSDBM 2009

Database Research Group

Conclusions & Next

- **Novel parallel level-by-level approach**
  - Excellent I/O performance
  - Almost linear speedup
  - Robust query performance
  - Conceptual Simplicity
- **Efficient partitioning for load balancing**
  - Minimize overlap/MBR Area
- **Loading algorithms for parallel R-trees**
  - Balancing query performance over set of machines

Database Research Group

Thank You!

Q&A

Database Research Group