CWI

Merging What's Cracked, Cracking What's Merged

Adaptive Indexing in Main-Memory Column-Stores

Stratos Idreos, Stefan Manegold, (CWI)
Harumi Kuno, Goetz Graefe (HP Labs)

PVLDB 2011, 4(9)

hp invent

monetdb

Physical design problem

Database systems perform efficiently
only after proper tuning...

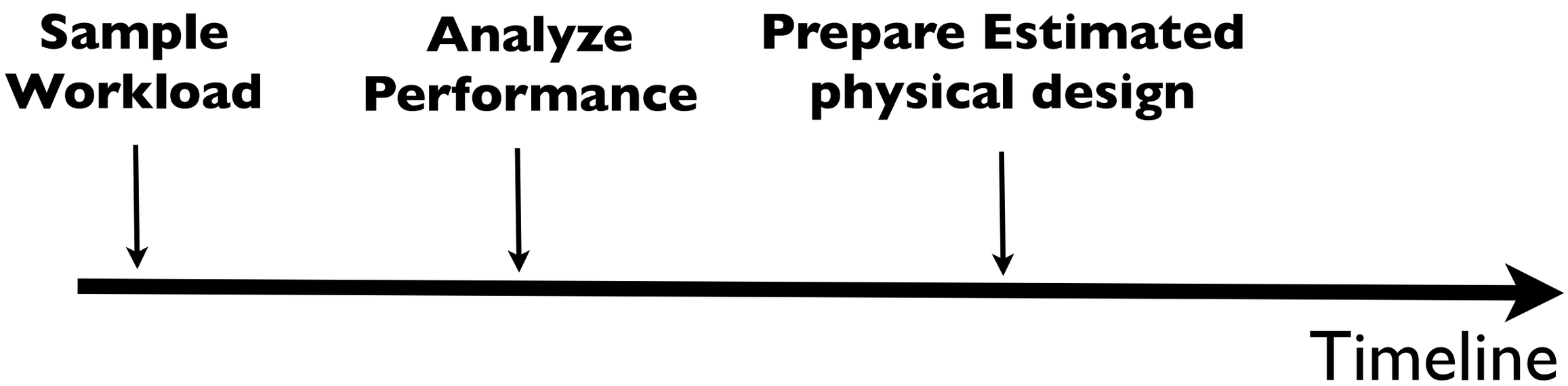
which indexes to build?
on which data parts?
and when to build them?

DBA without cracking

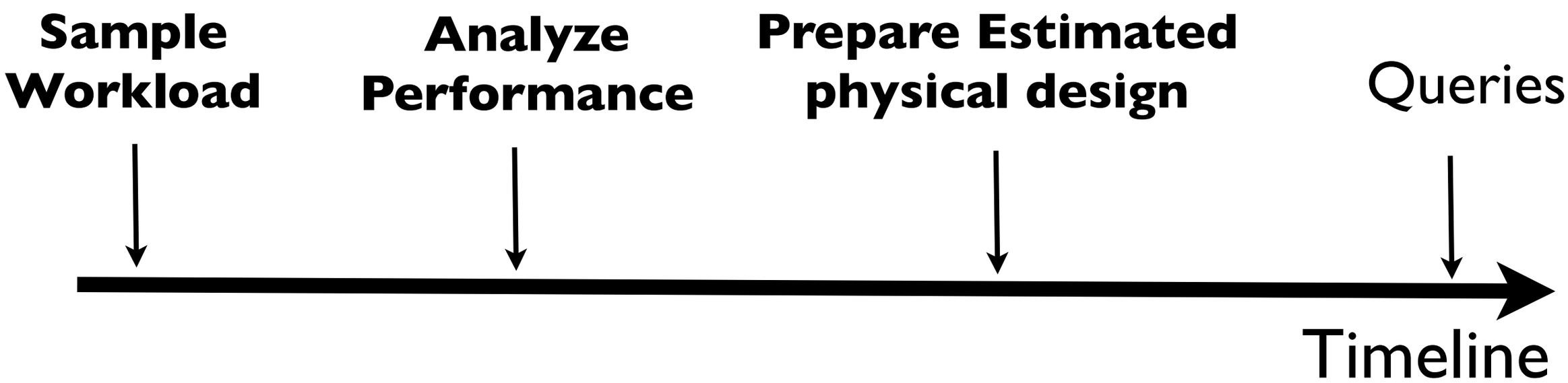
Physical Design

Physical Design

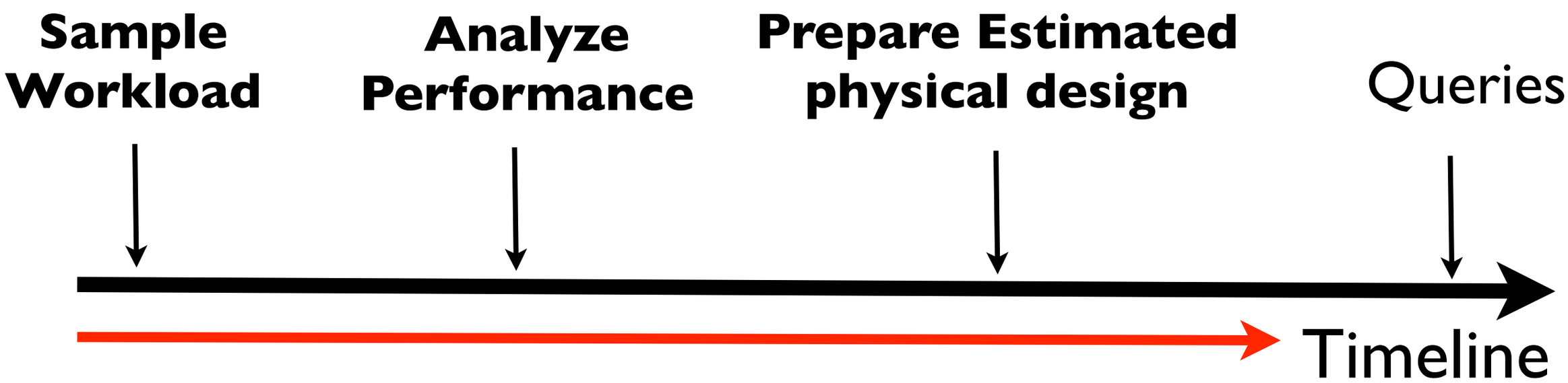
Physical Design



Physical Design

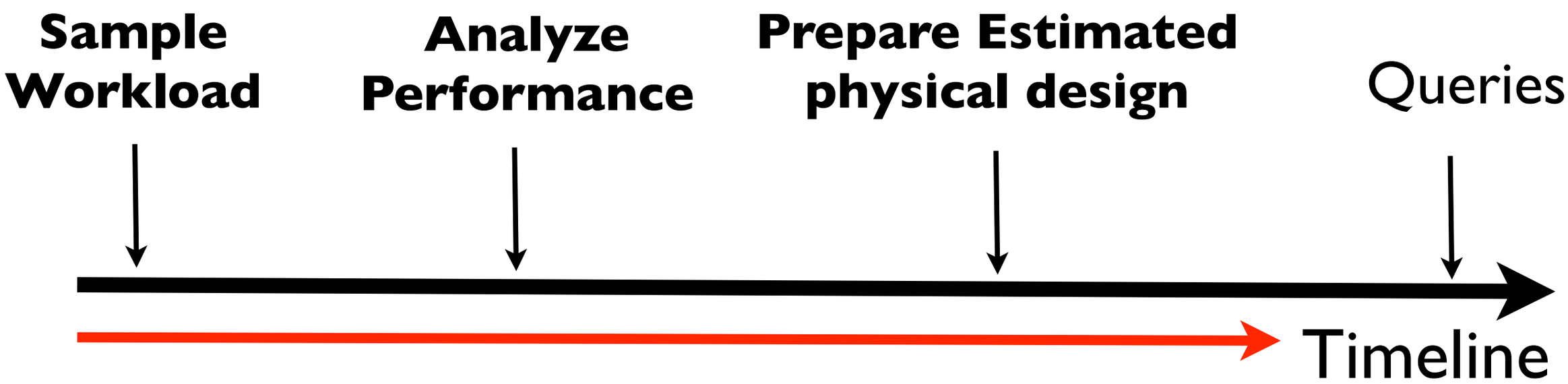


Physical Design



Complex and time consuming process

Physical Design



Complex and time consuming process

? **Dynamic Workloads**

? **Very Large Databases**

Dynamic environments

idle time **workload knowledge**

Dynamic environments

idle time **workload knowledge**

some problem cases

Dynamic environments

idle time **workload knowledge**

some problem cases

- Not enough idle time to finish proper tuning

Dynamic environments

idle time **workload knowledge**

some problem cases

- Not enough idle time to finish proper tuning
- By the time we finish tuning, the workload changes

Dynamic environments

idle time **workload knowledge**

some problem cases

- Not enough idle time to finish proper tuning
- By the time we finish tuning, the workload changes
- No index support during tuning

Dynamic environments

idle time **workload knowledge**

some problem cases

- Not enough idle time to finish proper tuning
- By the time we finish tuning, the workload changes
- No index support during tuning
- Not all data parts are equally useful

Database Cracking

For dynamic environments:

Remove all tuning, physical design steps but still get similar performance as a fully tuned system

How?

Design new auto-tuning kernels

(operators, plans, structures, etc.)

DBA with cracking

Database Cracking

no monitoring

no preparation

no external tools

no full indexes

no human involvement

CWI

Database Cracking

no monitoring

no preparation

no external tools

no full indexes

no human involvement

Continuous on-the-fly physical reorganization

hp invent

monetdb

CWI

Database Cracking

no monitoring

no preparation

no external tools

no full indexes

no human involvement

Continuous on-the-fly physical reorganization

partial, incremental, adaptive indexing

hp invent

monetdb

CWI

Database Cracking

no monitoring

no preparation

no external tools

no full indexes

no human involvement

Continuous on-the-fly physical reorganization

partial, incremental, adaptive indexing

designed for modern column-stores

hp invent

monetdb

Database Cracking

Each query is treated as an advice on how data should be stored

Cracking Example

Each query is treated as an advice on how data should be stored

Q1:
select *
from R
where R.A > 10
and R.A < 14

Column A

13
16
4
9
2
12
7
1
19
3
14
11
8
6

Cracking Example

Each query is treated as an advice on how data should be stored

Physically reorganize based on the selection predicate

Q1:
select *
from R
where R.A > 10
and R.A < 14

Column A

13
16
4
9
2
12
7
1
19
3
14
11
8
6

Cracking Example

Each query is treated as an advice on how data should be stored

Physically reorganize based on the selection predicate

Q1:
select *
from R
where R.A > 10
and R.A < 14

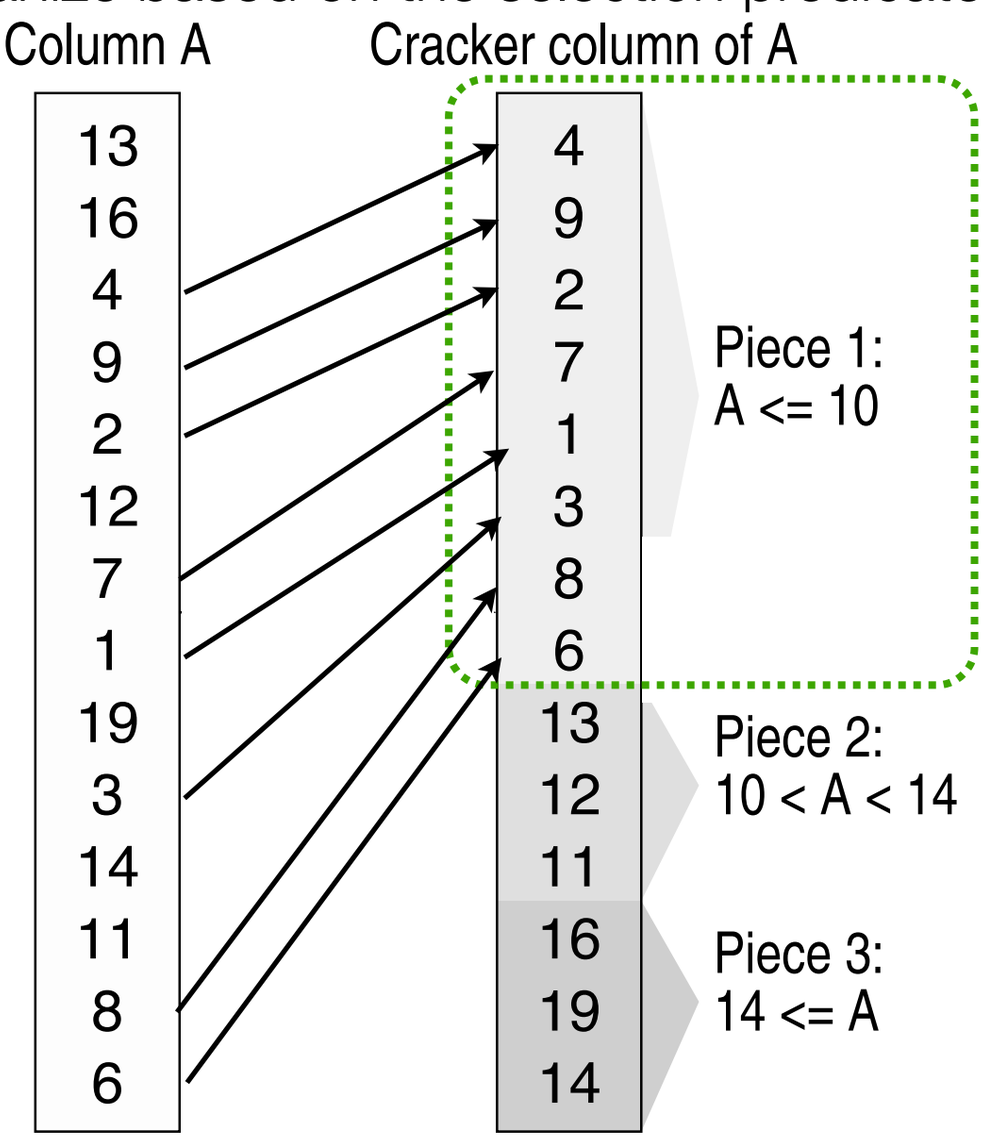
Column A	Cracker column of A	
13	4	Piece 1: A <= 10
16	9	
4	2	
9	7	
2	1	
12	3	
7	8	
1	6	
19	13	Piece 2: 10 < A < 14
3	12	
14	11	
11	16	Piece 3: 14 <= A
8	19	
6	14	

Cracking Example

Each query is treated as an advice on how data should be stored

Physically reorganize based on the selection predicate

Q1:
select *
from R
where R.A > 10
and R.A < 14



Cracking Example

Each query is treated as an advice on how data should be stored

Physically reorganize based on the selection predicate

Q1:
select *
from R
where R.A > 10
and R.A < 14

Column A
13
16
4
9
2
12
7
1
19
3
14
11
8
6

Cracker column of A

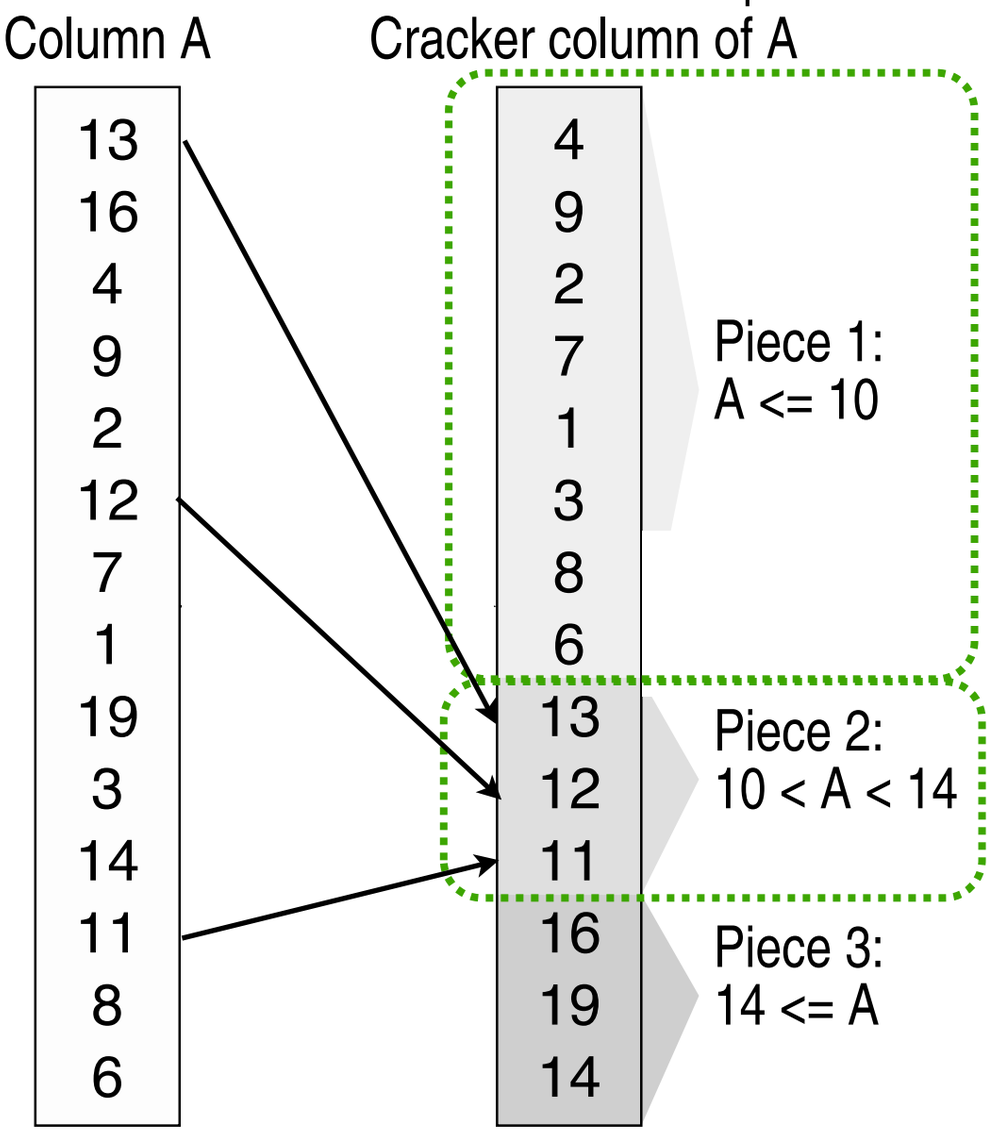
4	Piece 1: A <= 10
9	
2	
7	
1	
3	
8	
6	
13	Piece 2: 10 < A < 14
12	
11	
16	Piece 3: 14 <= A
19	
14	

Cracking Example

Each query is treated as an advice on how data should be stored

Physically reorganize based on the selection predicate

Q1:
select *
from R
where R.A > 10
and R.A < 14

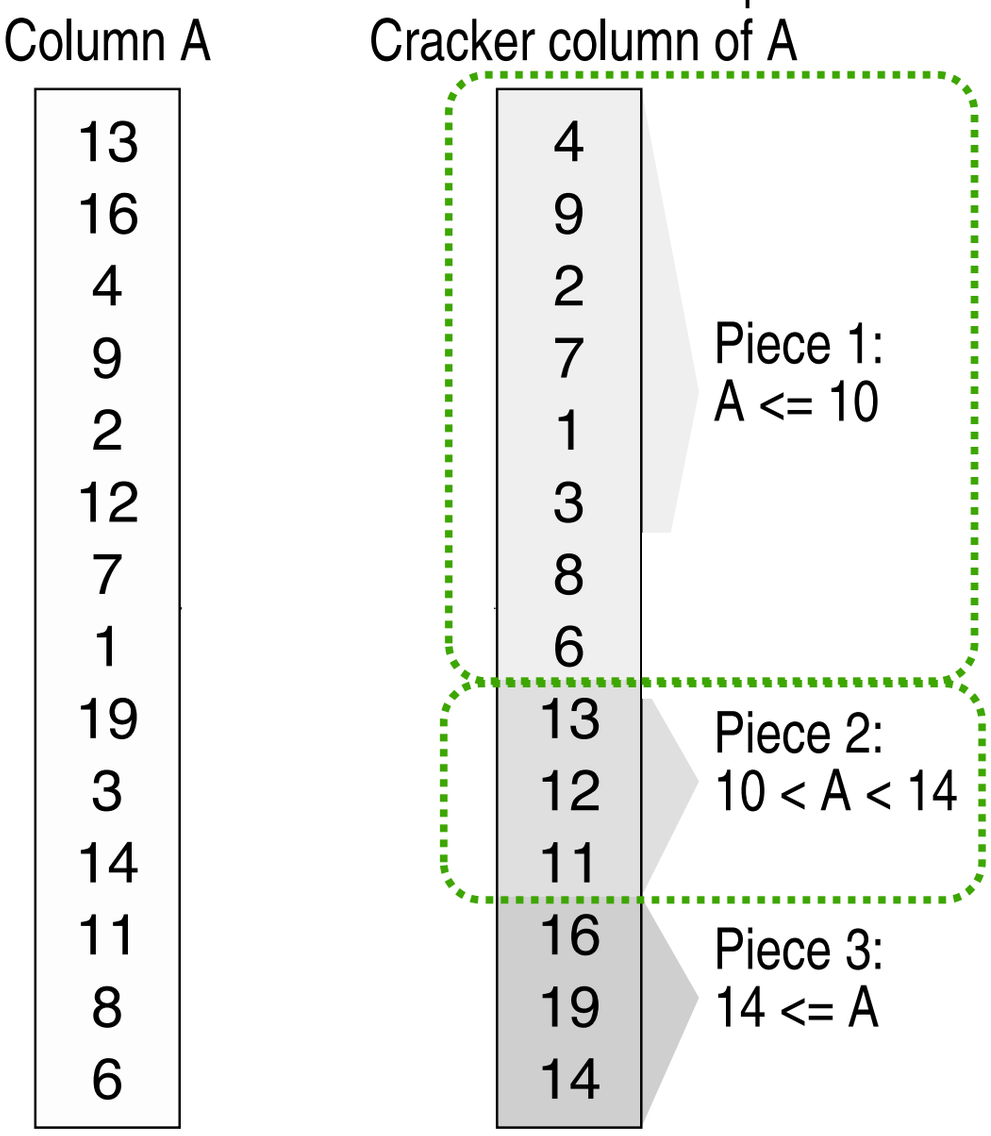


Cracking Example

Each query is treated as an advice on how data should be stored

Physically reorganize based on the selection predicate

Q1:
select *
from R
where R.A > 10
and R.A < 14

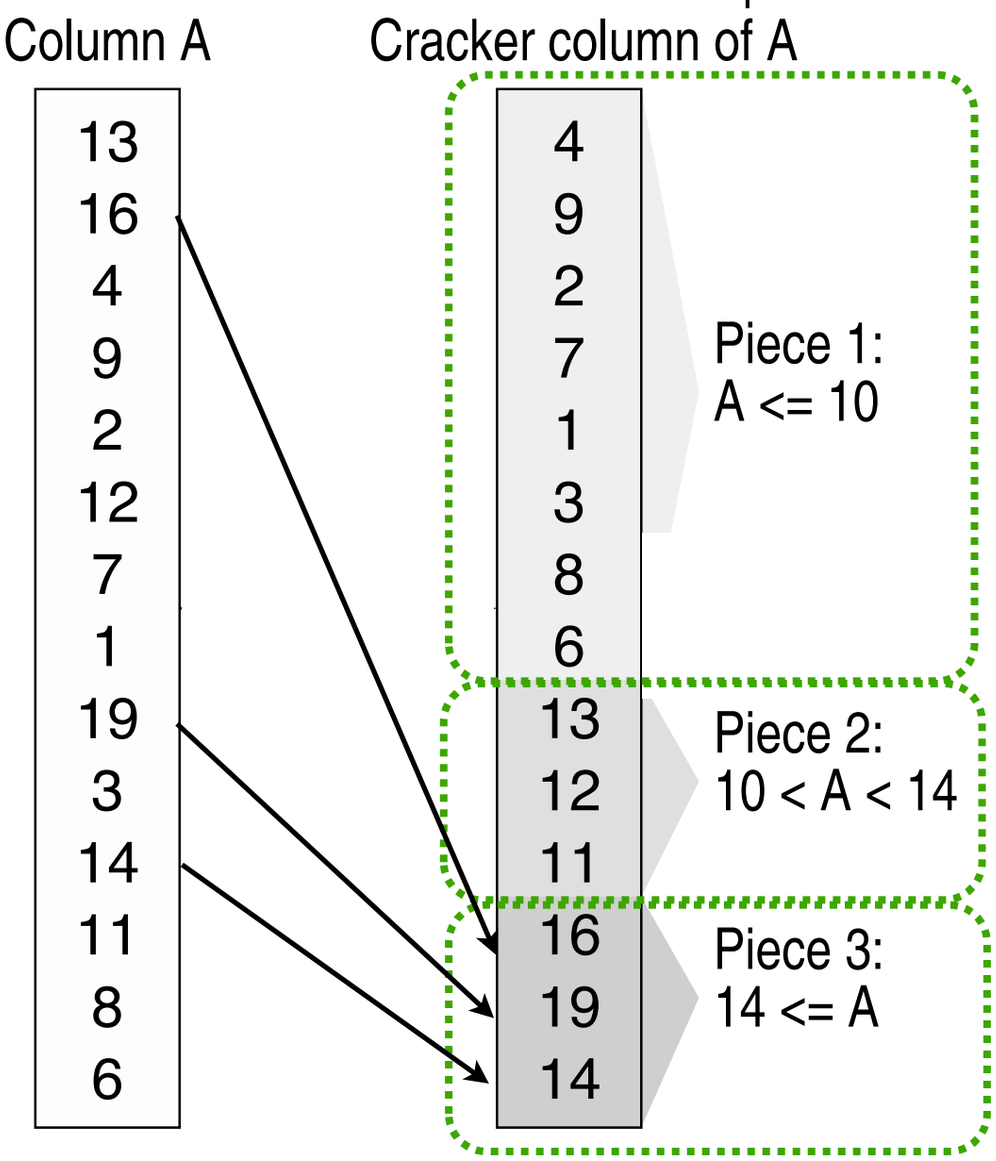


Cracking Example

Each query is treated as an advice on how data should be stored

Physically reorganize based on the selection predicate

Q1:
select *
from R
where R.A > 10
and R.A < 14

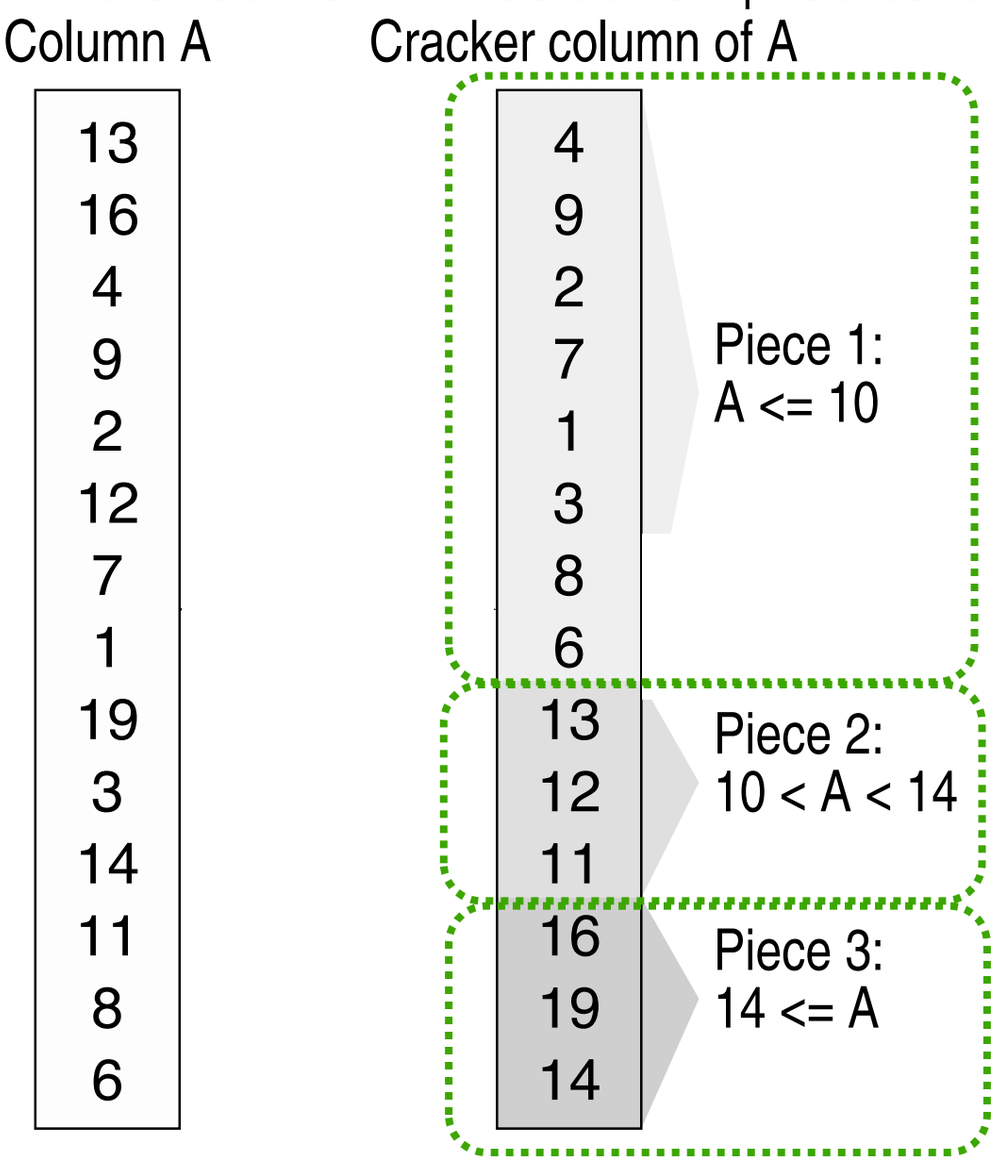


Cracking Example

Each query is treated as an advice on how data should be stored

Physically reorganize based on the selection predicate

Q1:
select *
from R
where R.A > 10
and R.A < 14

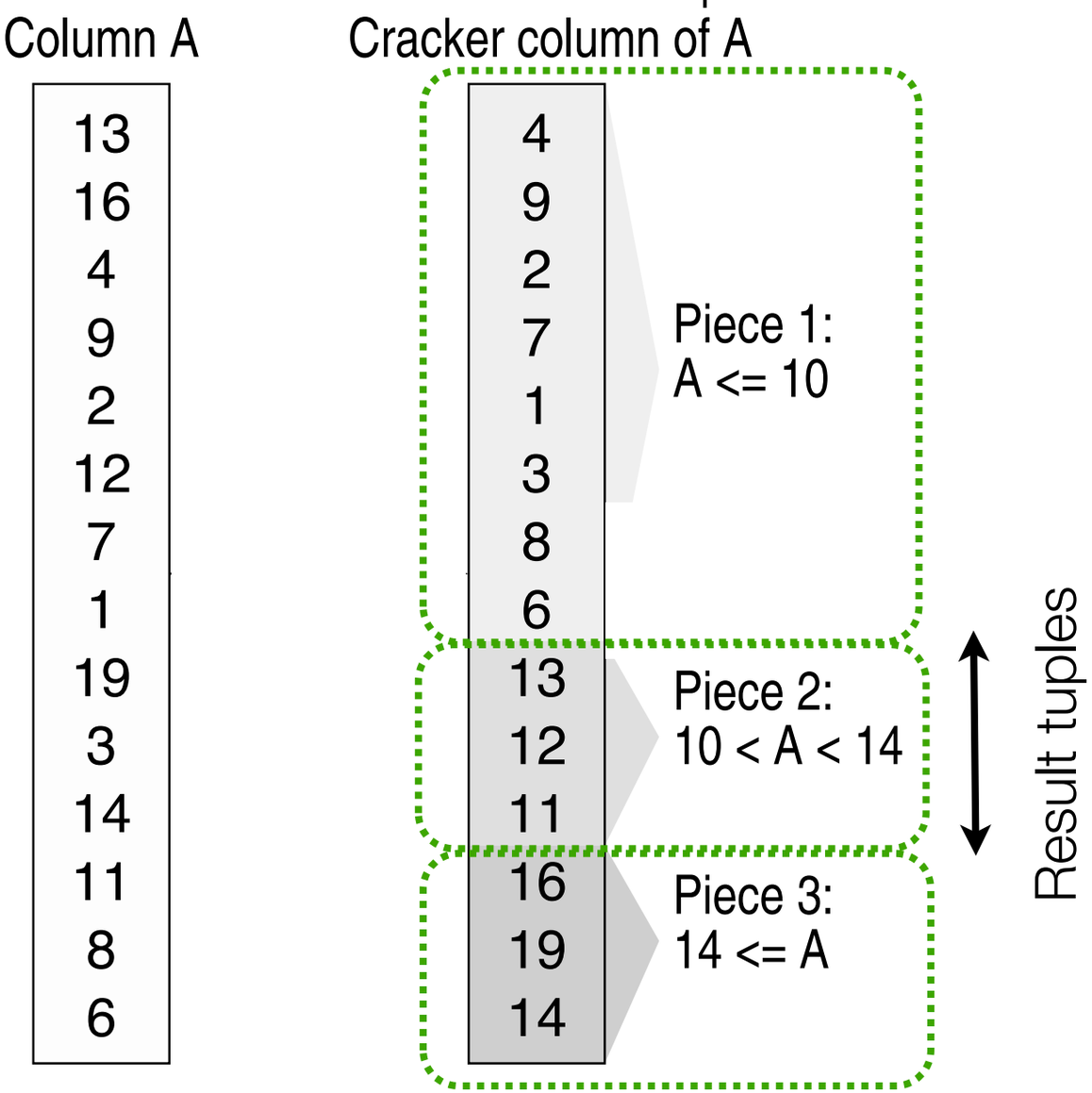


Cracking Example

Each query is treated as an advice on how data should be stored

Physically reorganize based on the selection predicate

Q1:
select *
from R
where R.A > 10
and R.A < 14

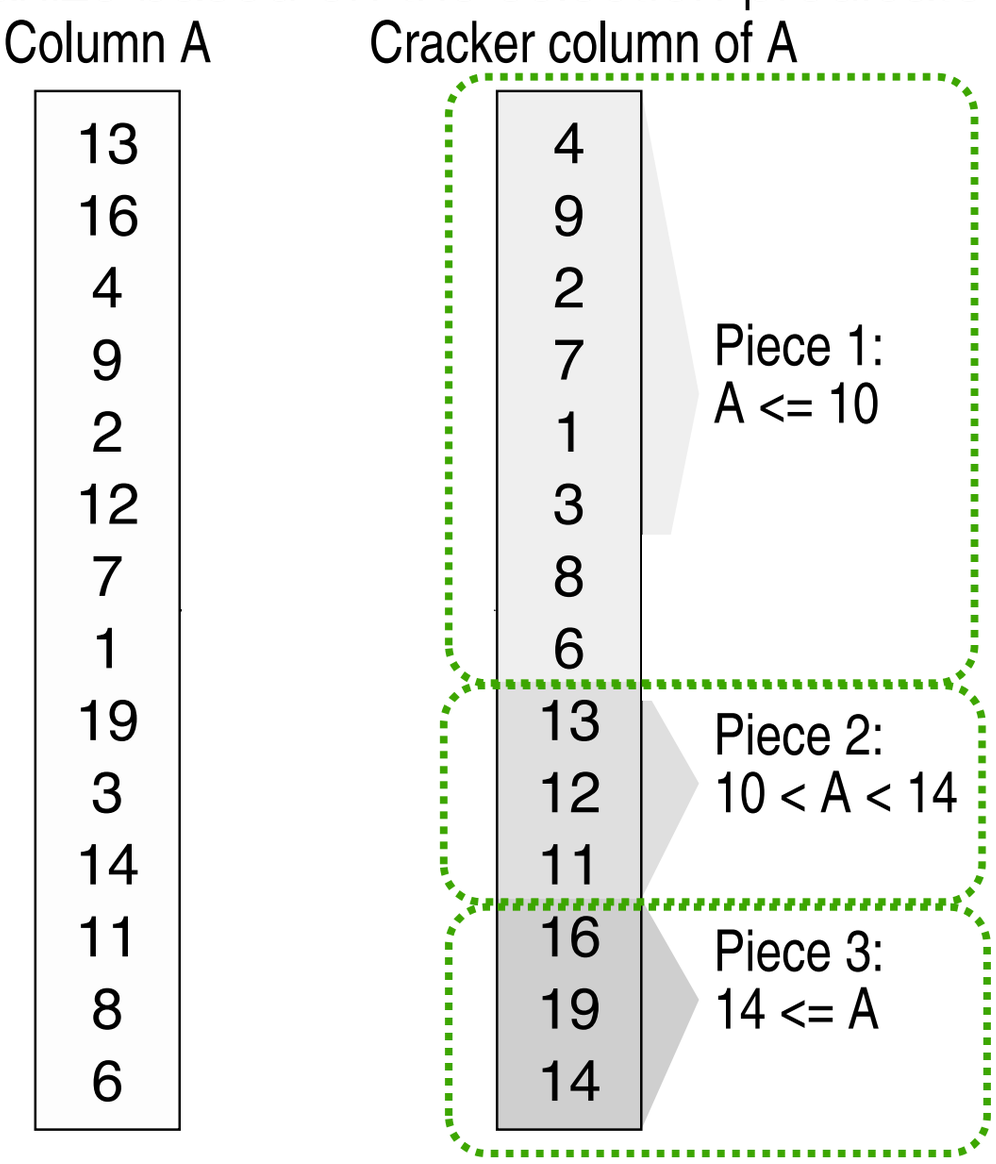


Cracking Example

Each query is treated as an advice on how data should be stored

Physically reorganize based on the selection predicate

Q1:
select *
from R
where R.A > 10
and R.A < 14



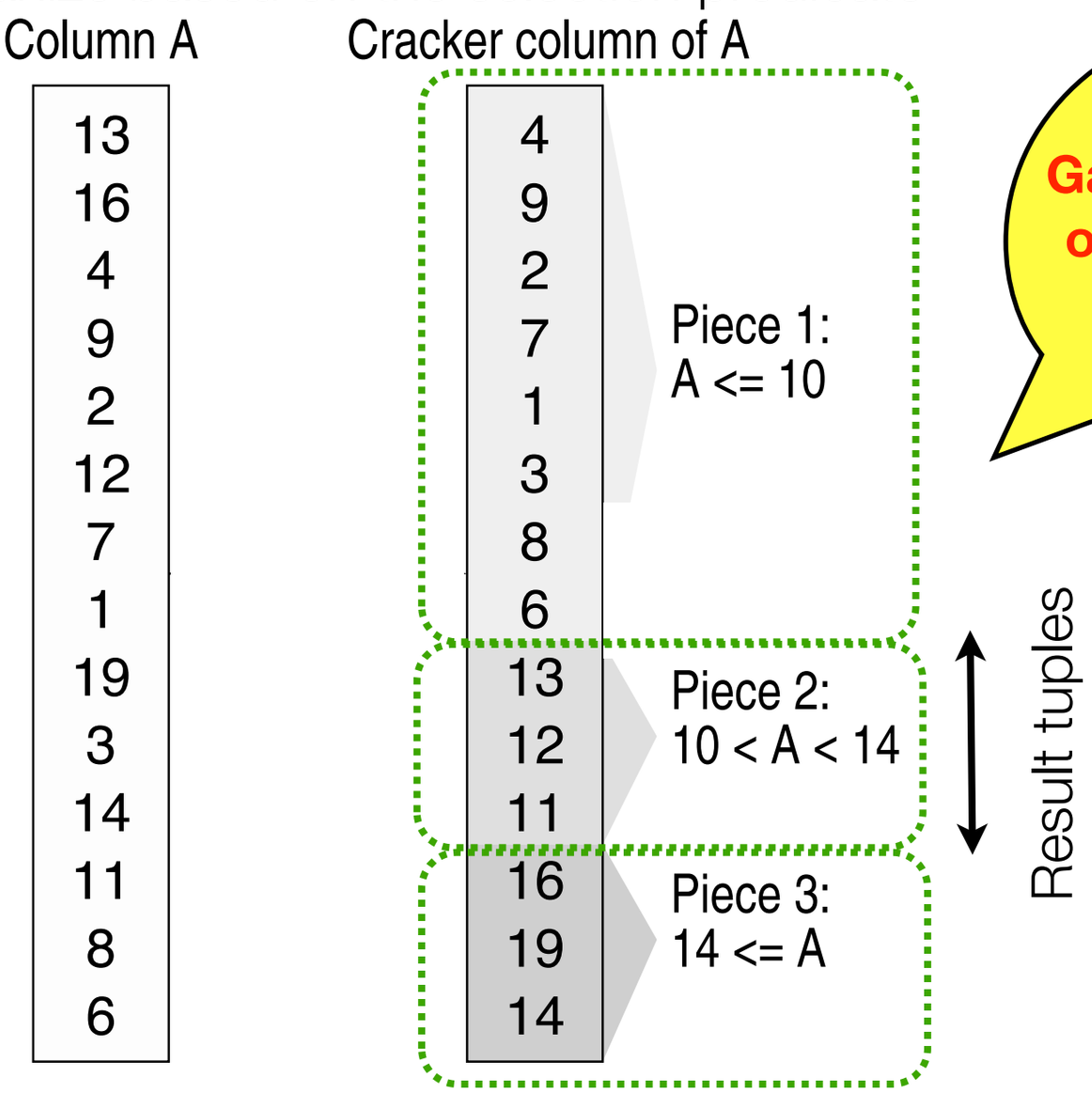
Gain knowledge on how data is organized

Cracking Example

Each query is treated as an advice on how data should be stored

Physically reorganize based on the selection predicate

Q1:
select *
from R
where R.A > 10
and R.A < 14



Gain knowledge on how data is organized

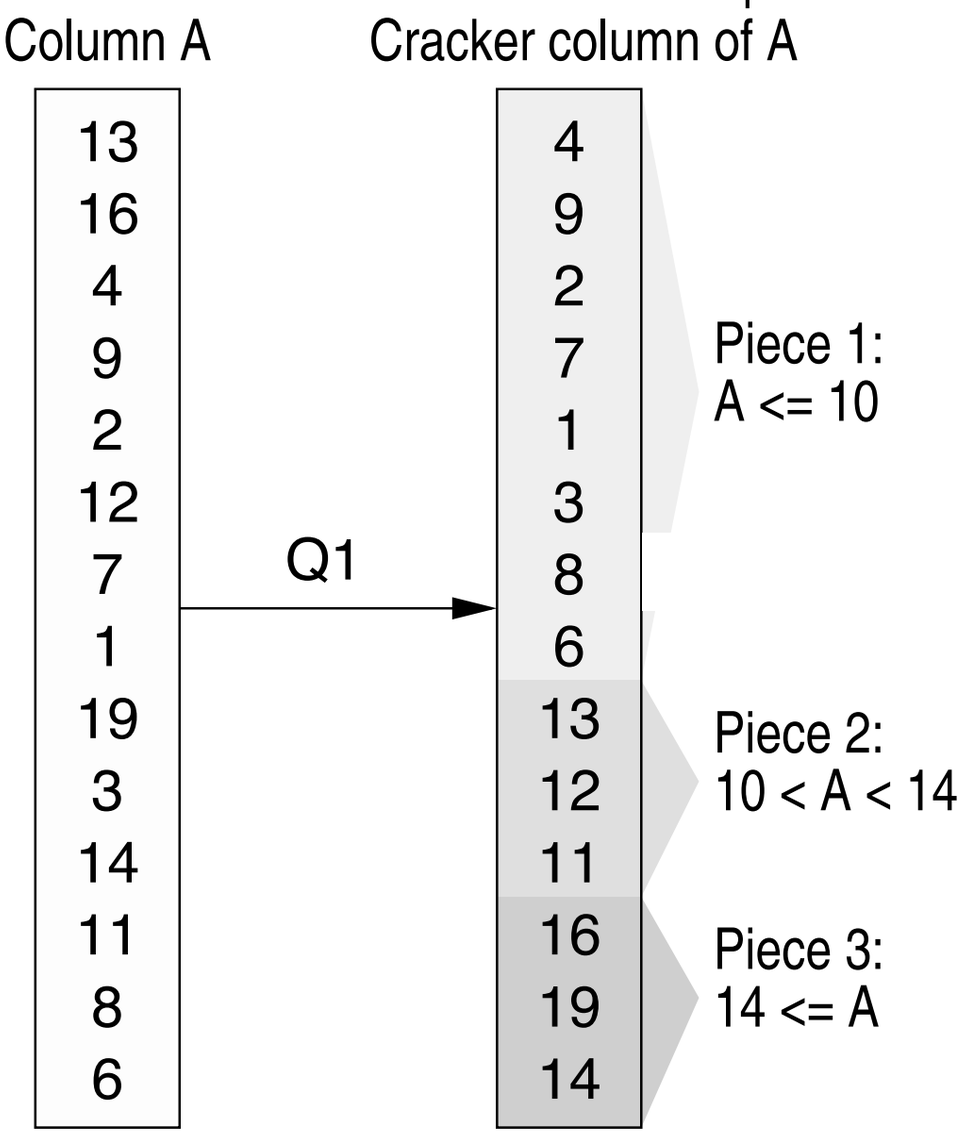
Dynamically/on-the-fly within the select-operator

Cracking Example

Each query is treated as an advice on how data should be stored

Physically reorganize based on the selection predicate

Q1:
select *
from R
where R.A > 10
and R.A < 14



Dynamically/on-the-fly within the select-operator

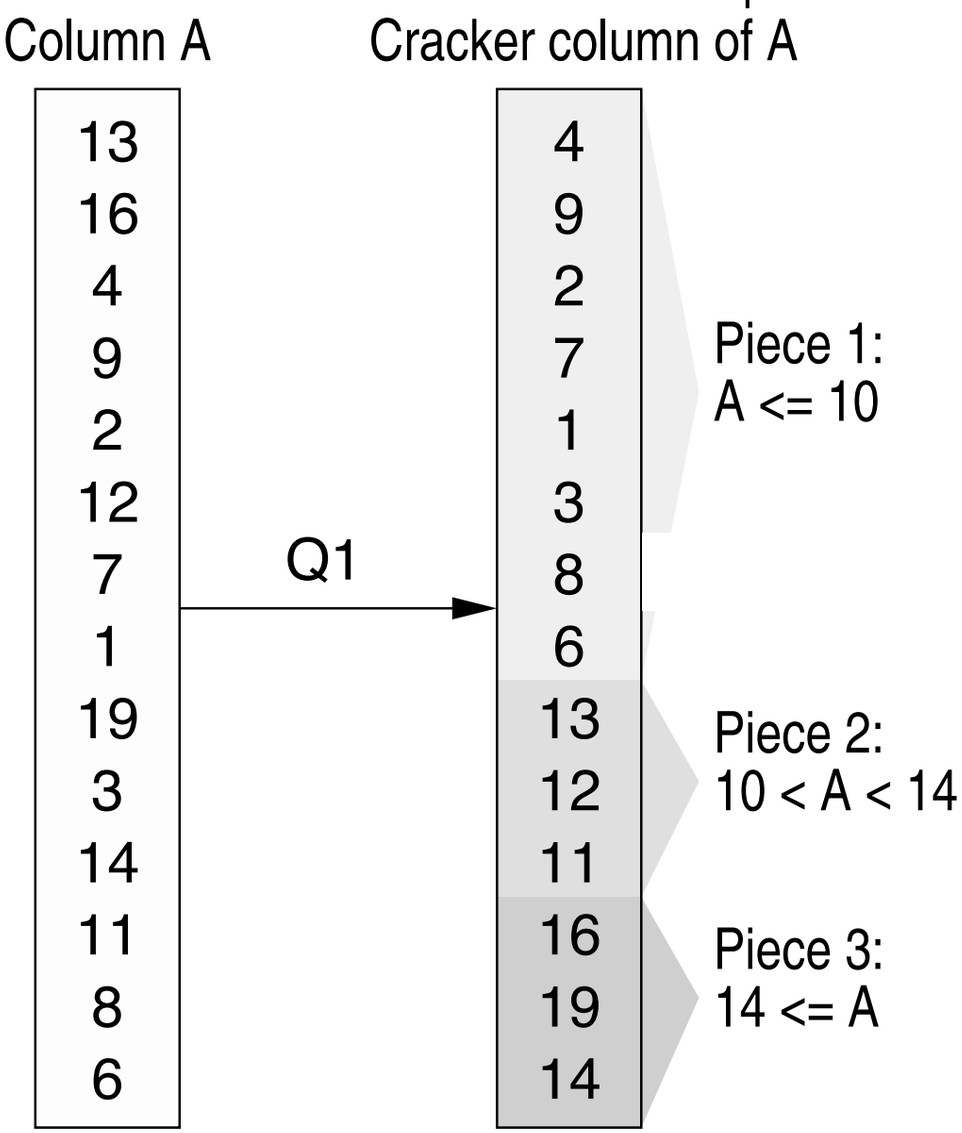
Cracking Example

Each query is treated as an advice on how data should be stored

Physically reorganize based on the selection predicate

Q1:
select *
from R
where R.A > 10
and R.A < 14

Q2:
select *
from R
where R.A > 7
and R.A <= 16



Dynamically/on-the-fly within the select-operator

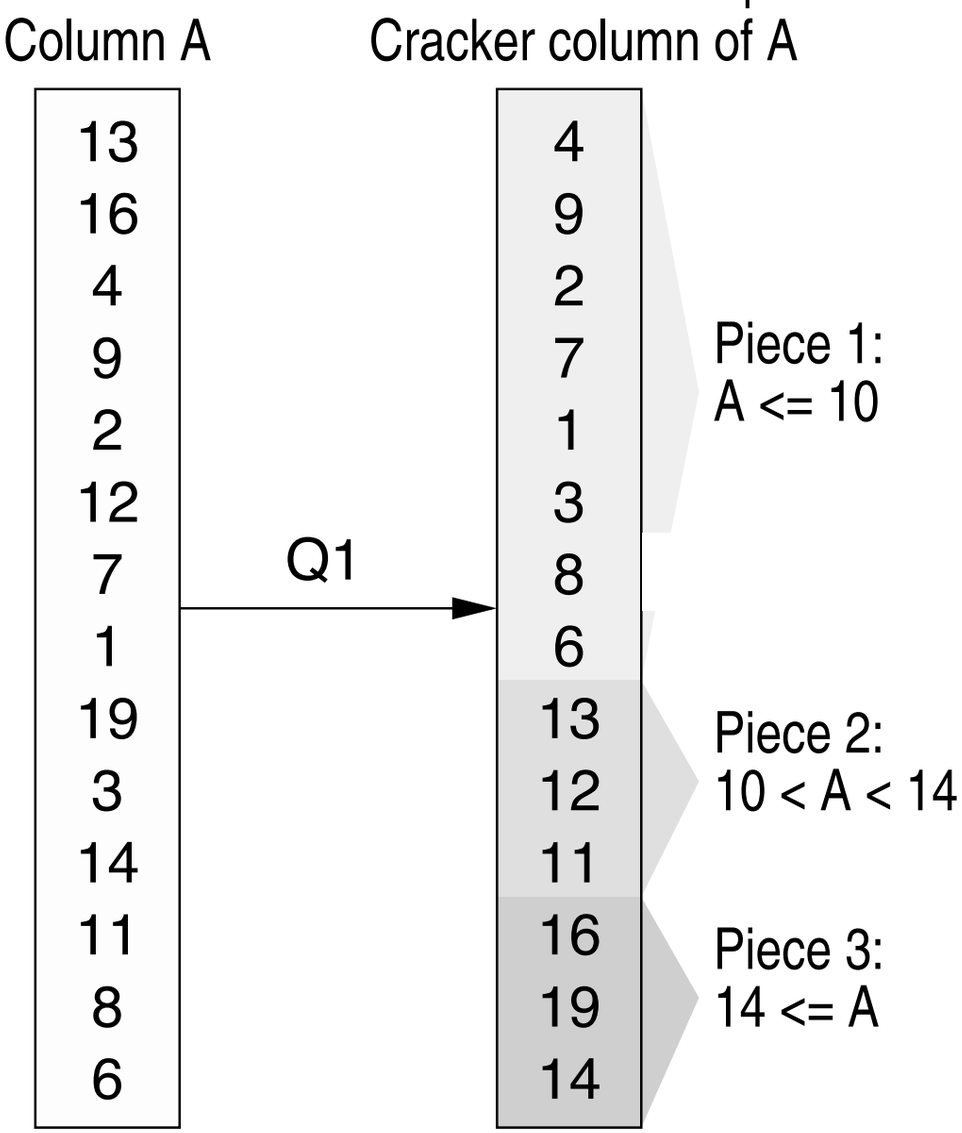
Cracking Example

Each query is treated as an advice on how data should be stored

Physically reorganize based on the selection predicate

Q1:
select *
from R
where R.A > 10
and R.A < 14

Q2:
select *
from R
where R.A > 7
and R.A <= 16



Dynamically/on-the-fly within the select-operator

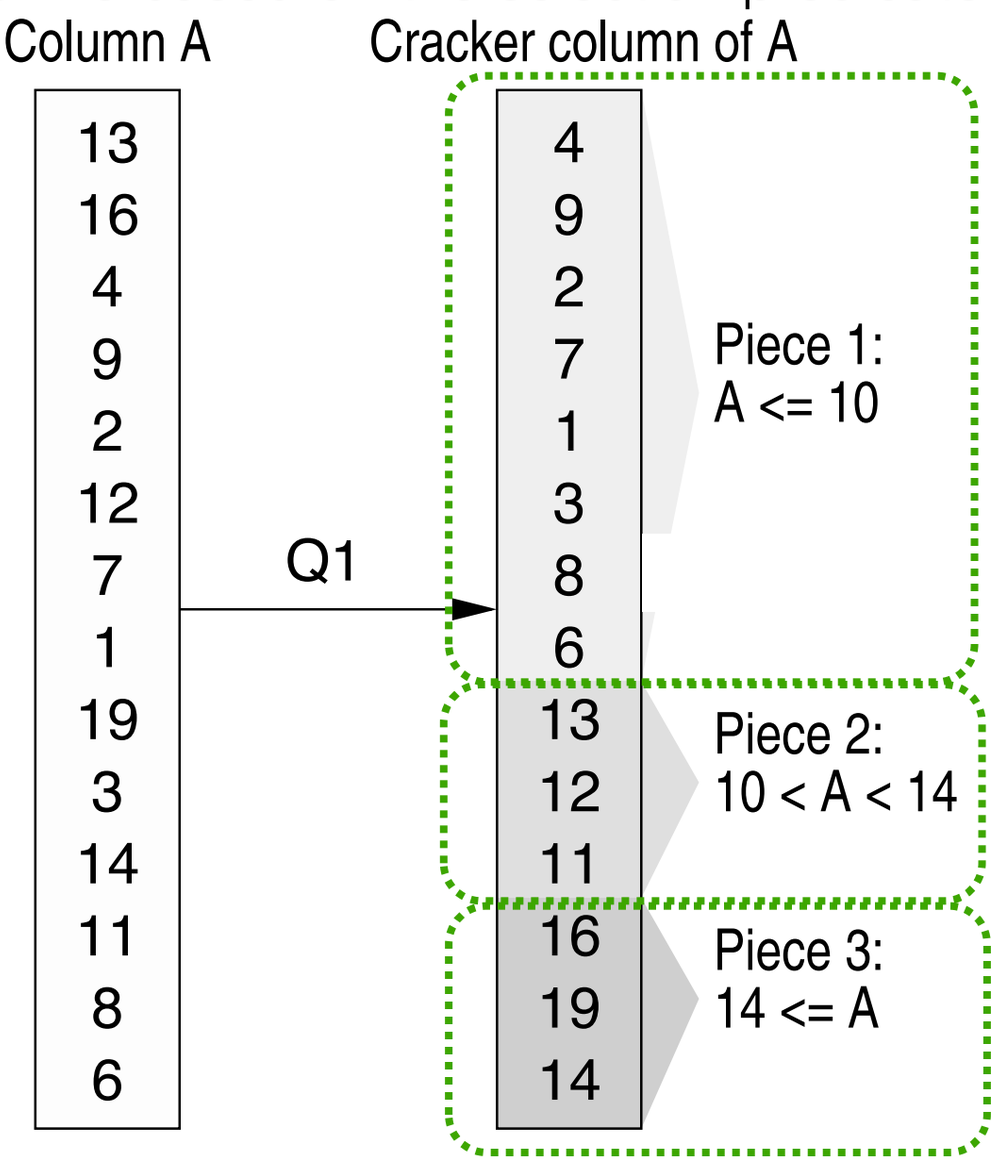
Cracking Example

Each query is treated as an advice on how data should be stored

Physically reorganize based on the selection predicate

Q1:
select *
from R
where R.A > 10
and R.A < 14

Q2:
select *
from R
where R.A > 7
and R.A <= 16



Dynamically/on-the-fly within the select-operator

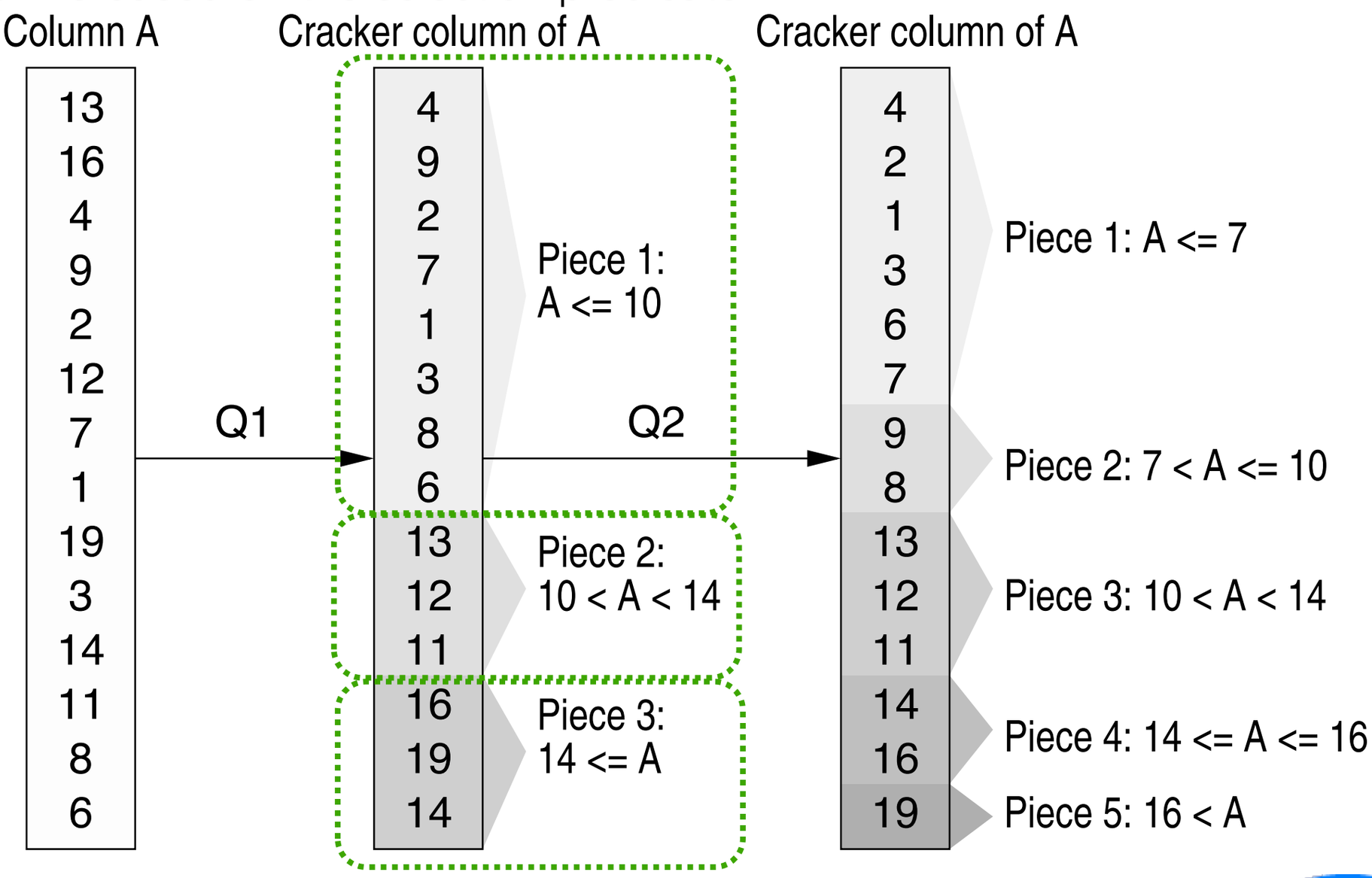
Cracking Example

Each query is treated as an advice on how data should be stored

Physically reorganize based on the selection predicate

Q1:
select *
from R
where R.A > 10
and R.A < 14

Q2:
select *
from R
where R.A > 7
and R.A <= 16



Dynamically/on-the-fly within the select-operator

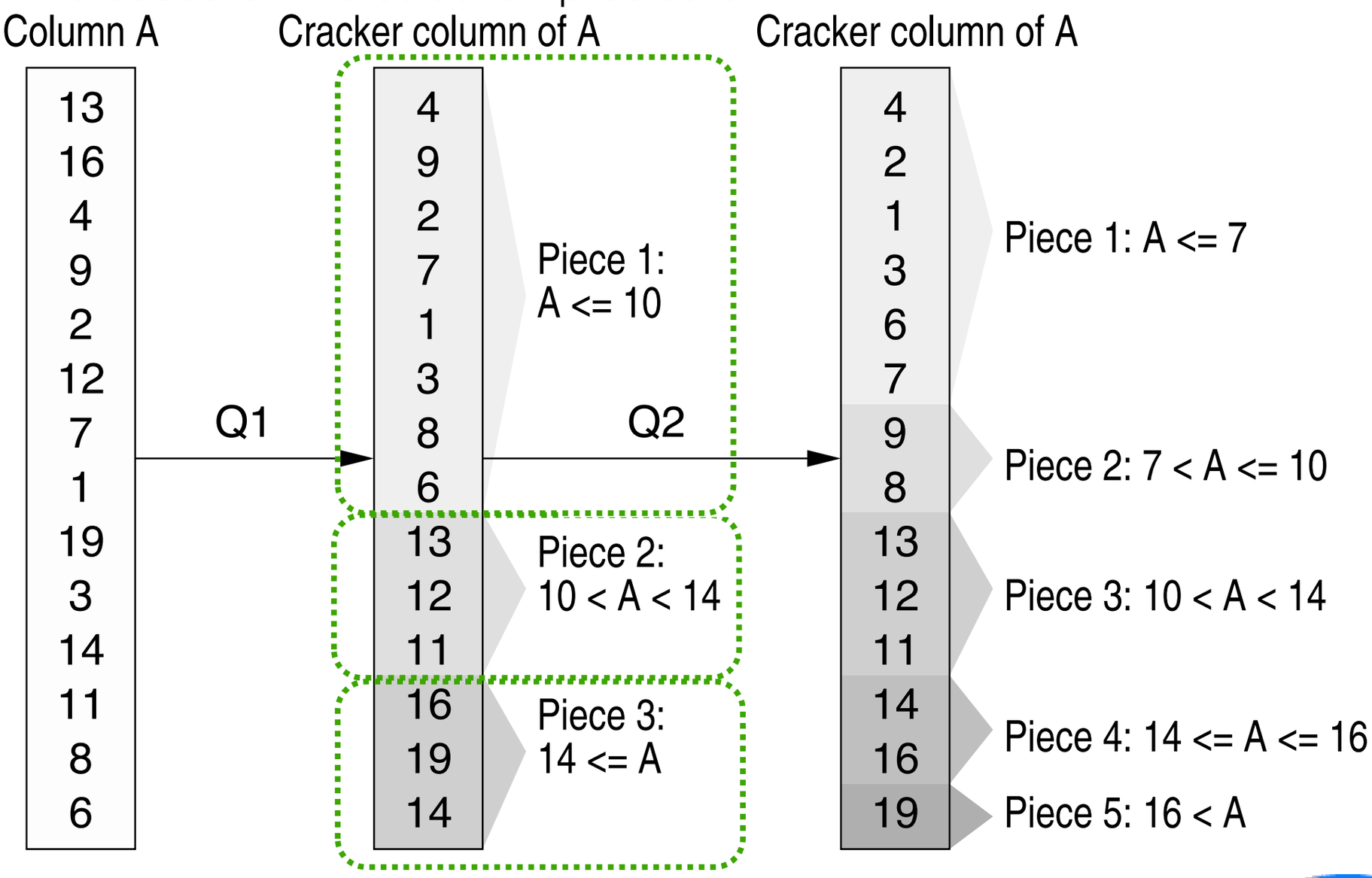
Cracking Example

Each query is treated as an advice on how data should be stored

Physically reorganize based on the selection predicate

Q1:
select *
from R
where R.A > 10
and R.A < 14

Q2:
select *
from R
where R.A > 7
and R.A <= 16



Dynamically/on-the-fly within the select-operator

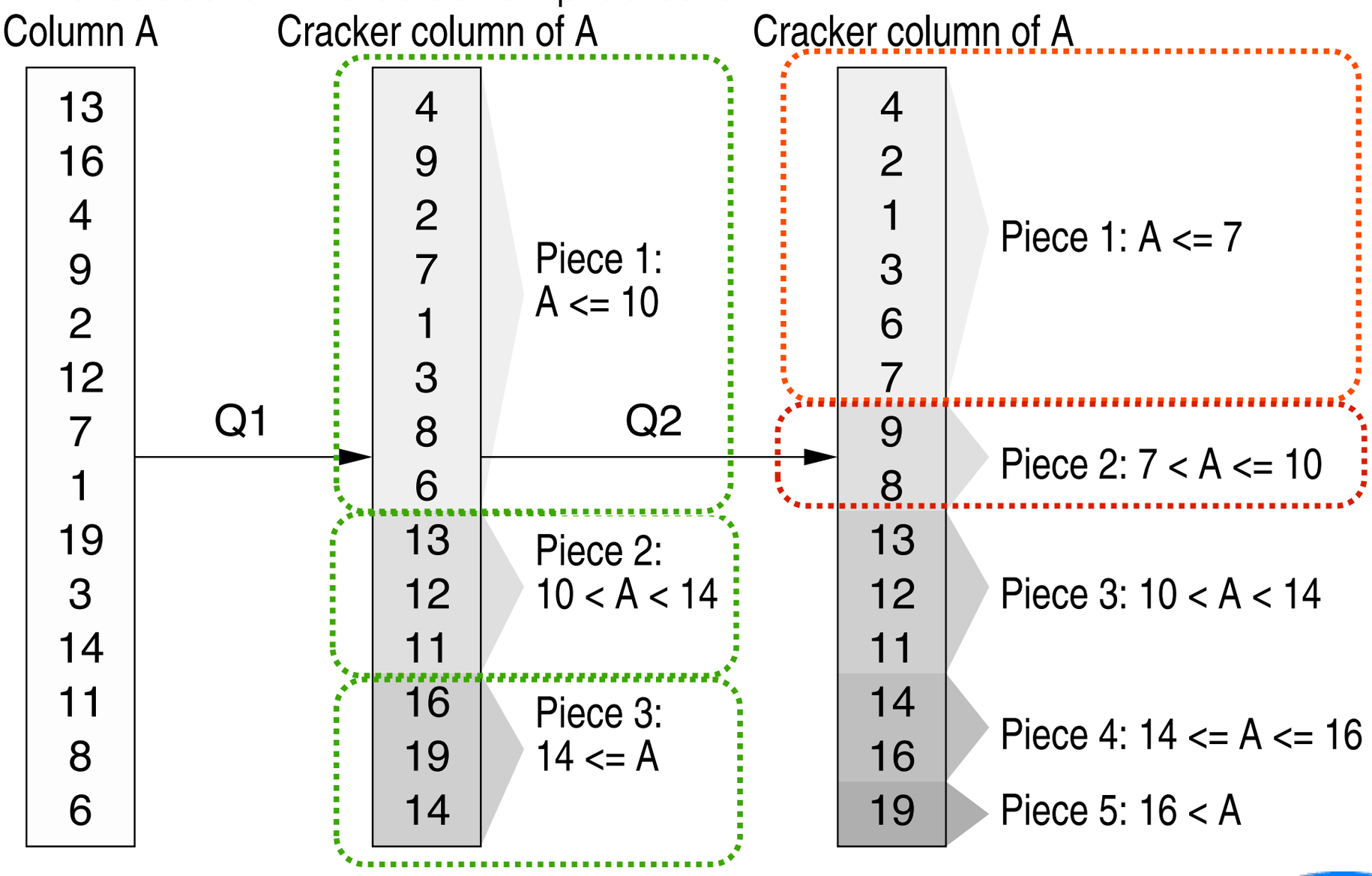
Cracking Example

Each query is treated as an advice on how data should be stored

Physically reorganize based on the selection predicate

Q1:
select *
from R
where R.A > 10
and R.A < 14

Q2:
select *
from R
where R.A > 7
and R.A <= 16



Dynamically/on-the-fly within the select-operator

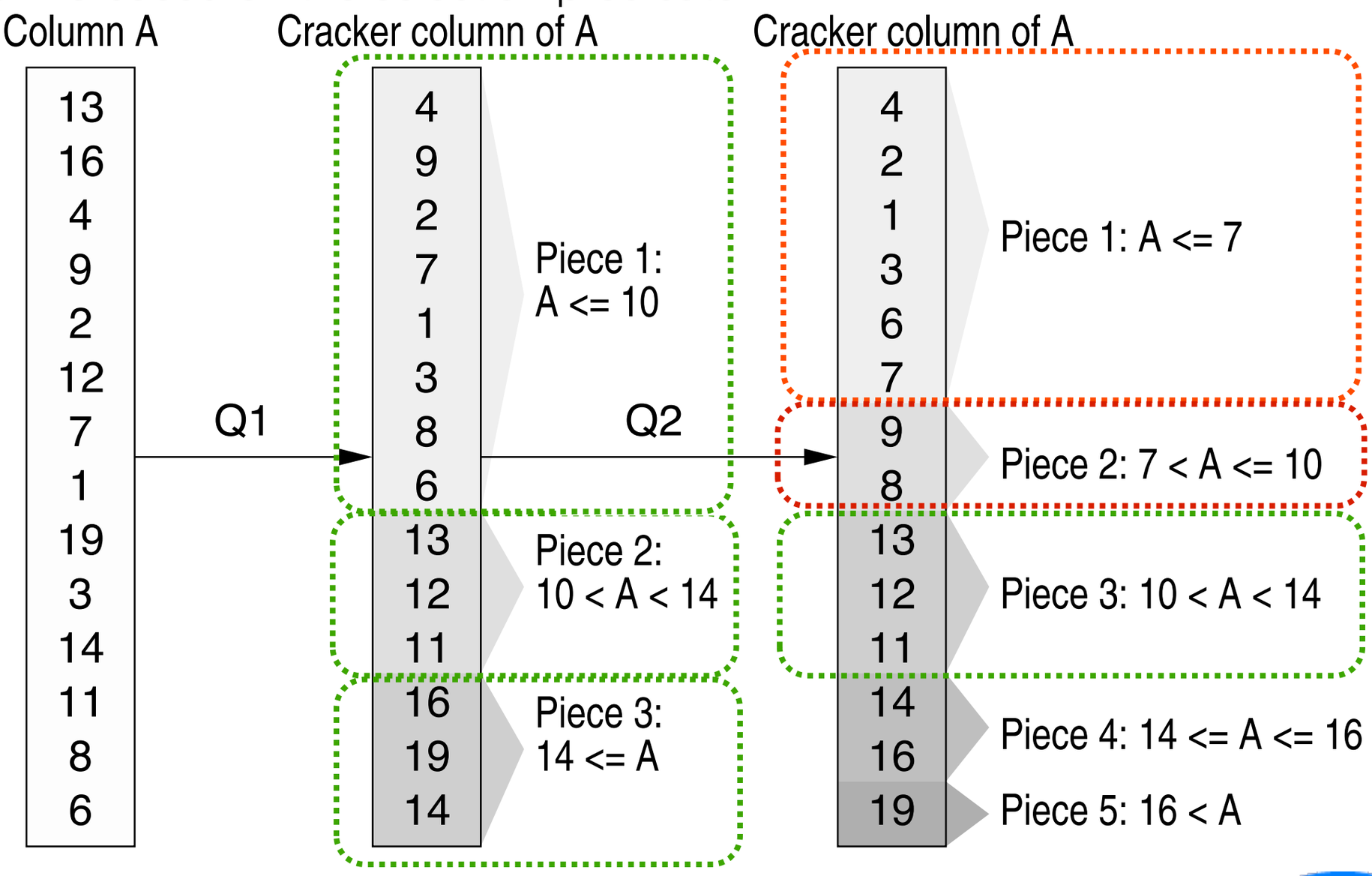
Cracking Example

Each query is treated as an advice on how data should be stored

Physically reorganize based on the selection predicate

Q1:
select *
from R
where R.A > 10
and R.A < 14

Q2:
select *
from R
where R.A > 7
and R.A <= 16



Dynamically/on-the-fly within the select-operator

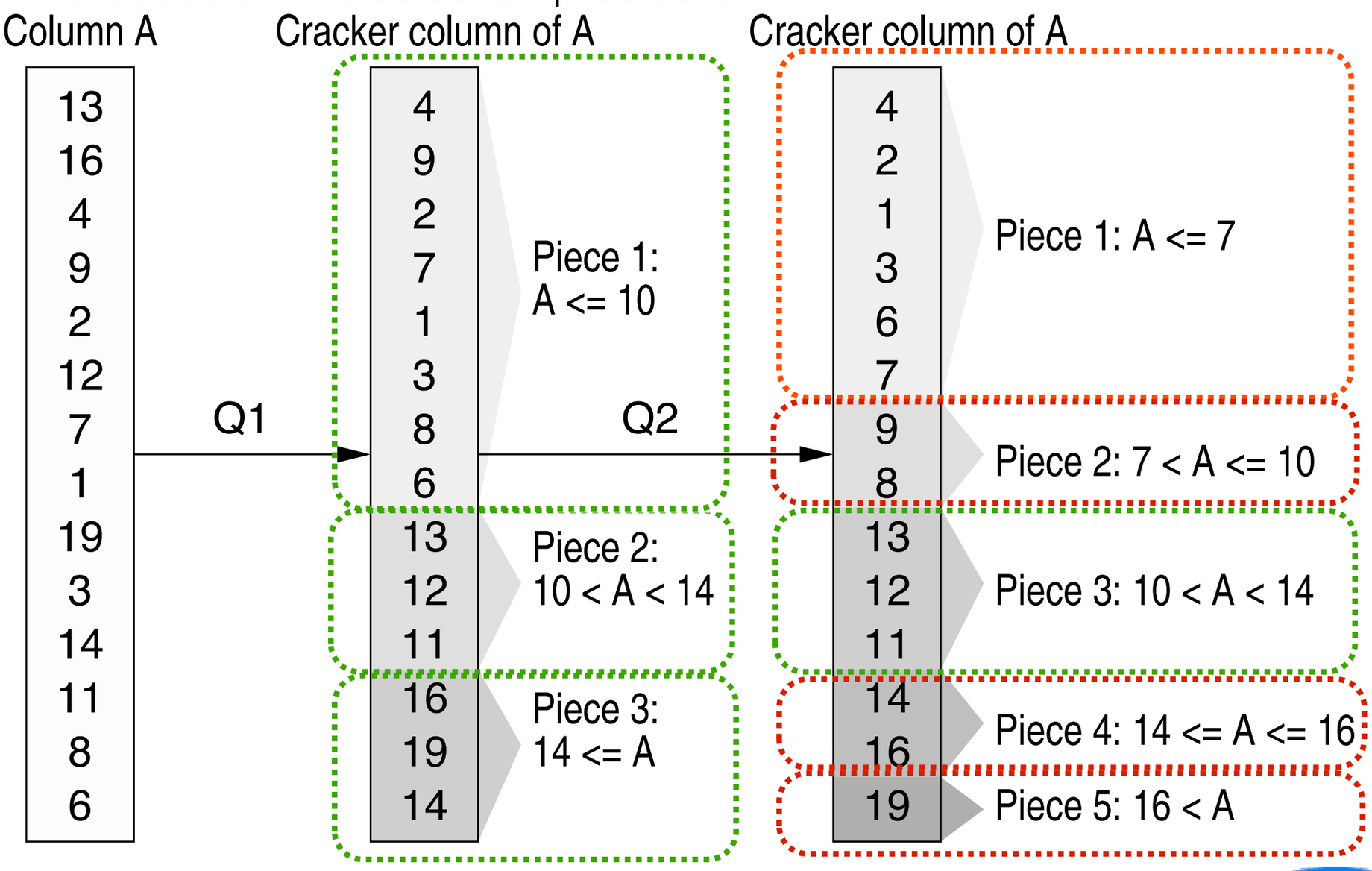
Cracking Example

Each query is treated as an advice on how data should be stored

Physically reorganize based on the selection predicate

Q1:
select *
from R
where R.A > 10
and R.A < 14

Q2:
select *
from R
where R.A > 7
and R.A <= 16



Dynamically/on-the-fly within the select-operator

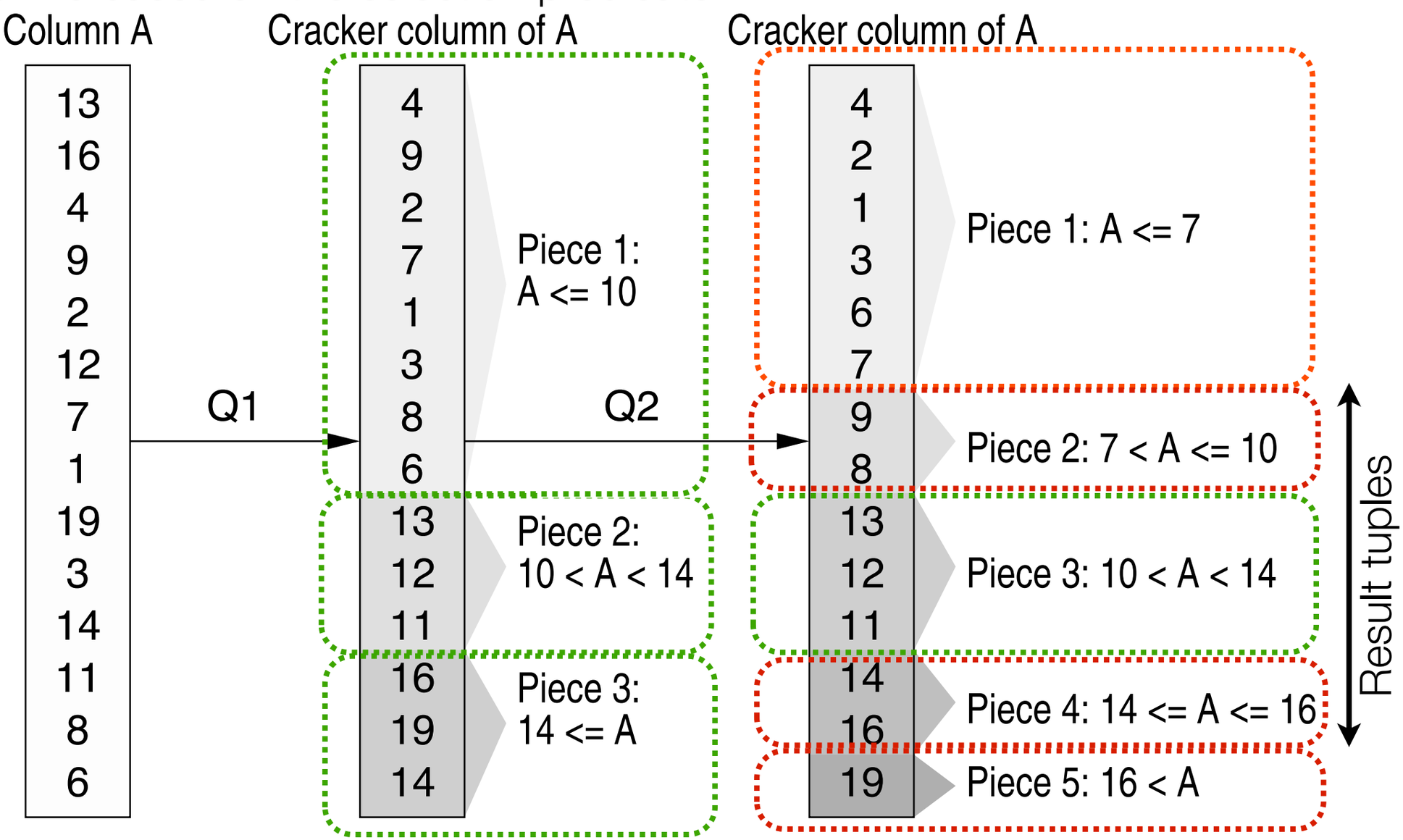
Cracking Example

Each query is treated as an advice on how data should be stored

Physically reorganize based on the selection predicate

Q1:
select *
from R
where R.A > 10
and R.A < 14

Q2:
select *
from R
where R.A > 7
and R.A <= 16



Dynamically/on-the-fly within the select-operator

Cracking Example

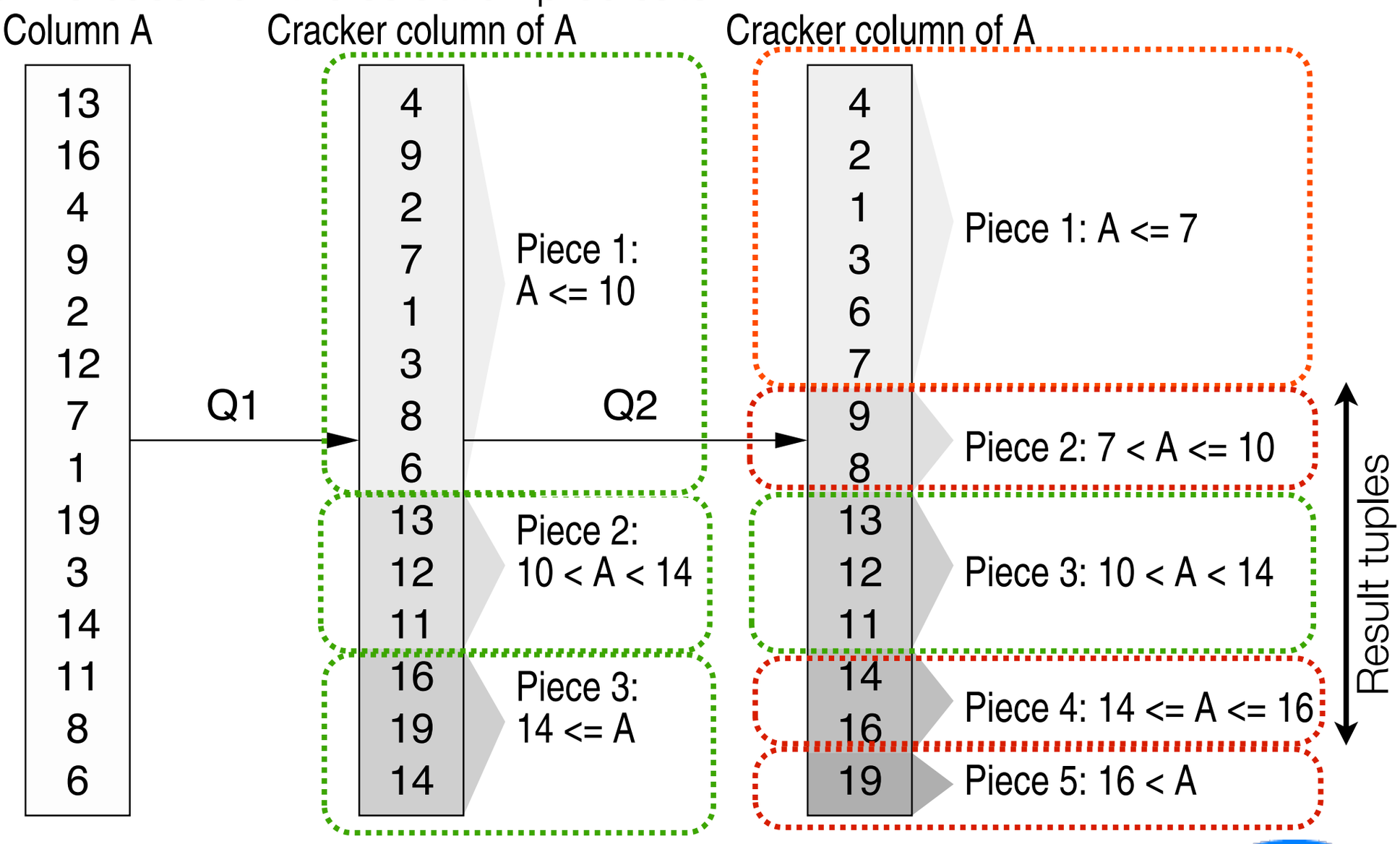
The more we crack, the more we learn

Each query is treated as an advice on how data shou

Physically reorganize based on the selection predicate

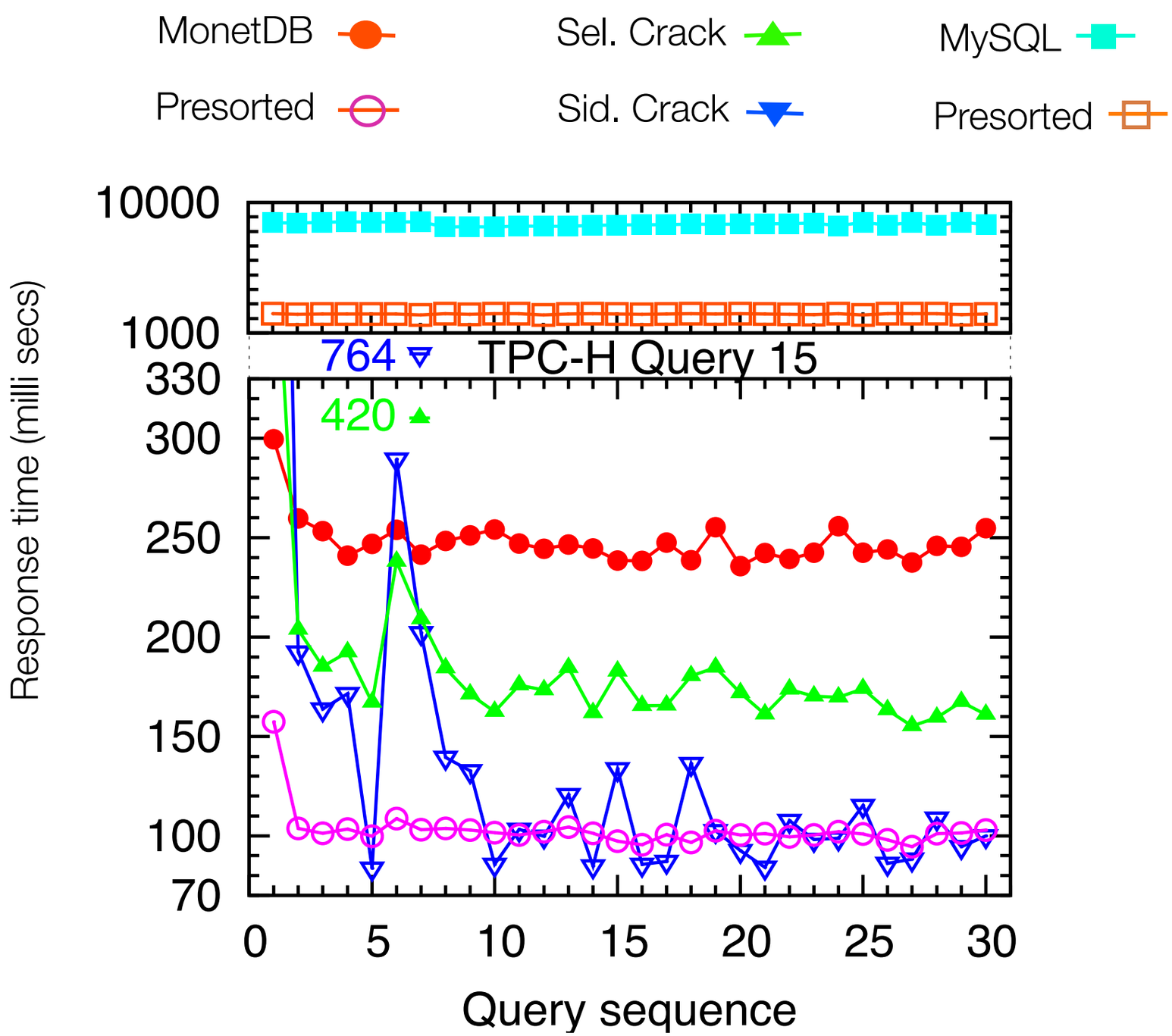
Q1:
select *
from R
where R.A > 10
and R.A < 14

Q2:
select *
from R
where R.A > 7
and R.A <= 16

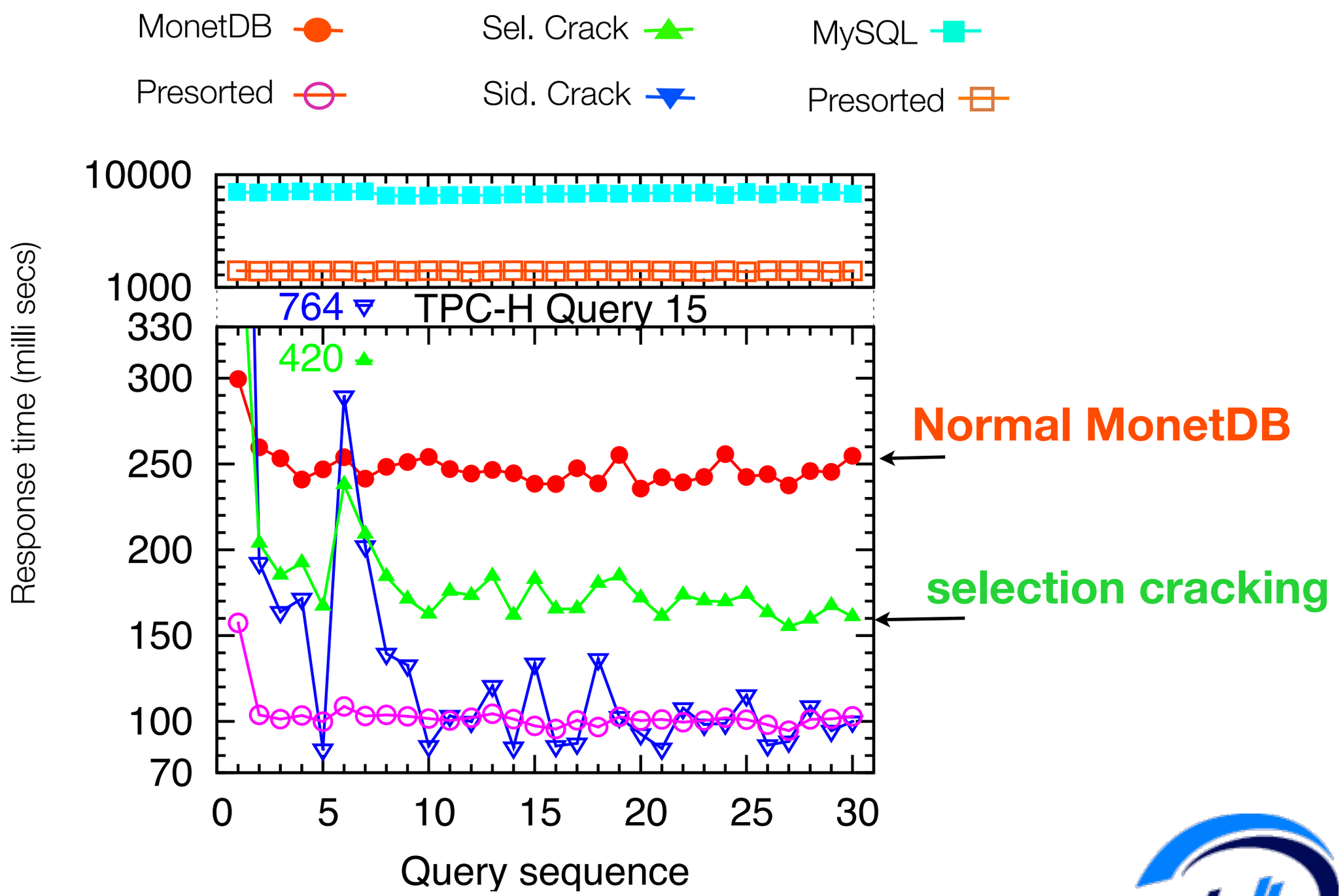


Dynamically/on-the-fly within the select-operator

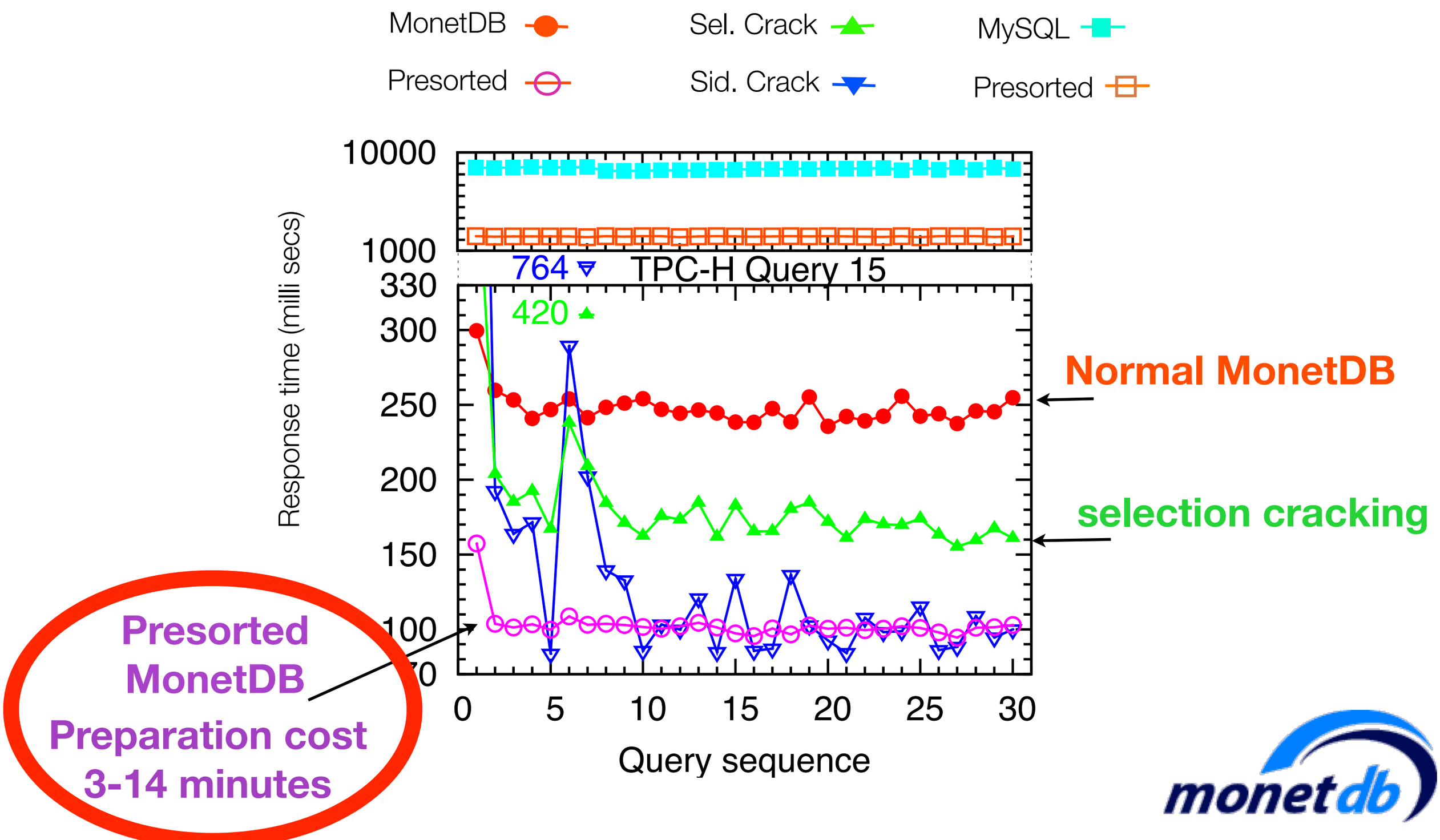
Self-organizing behavior (TPC-H)



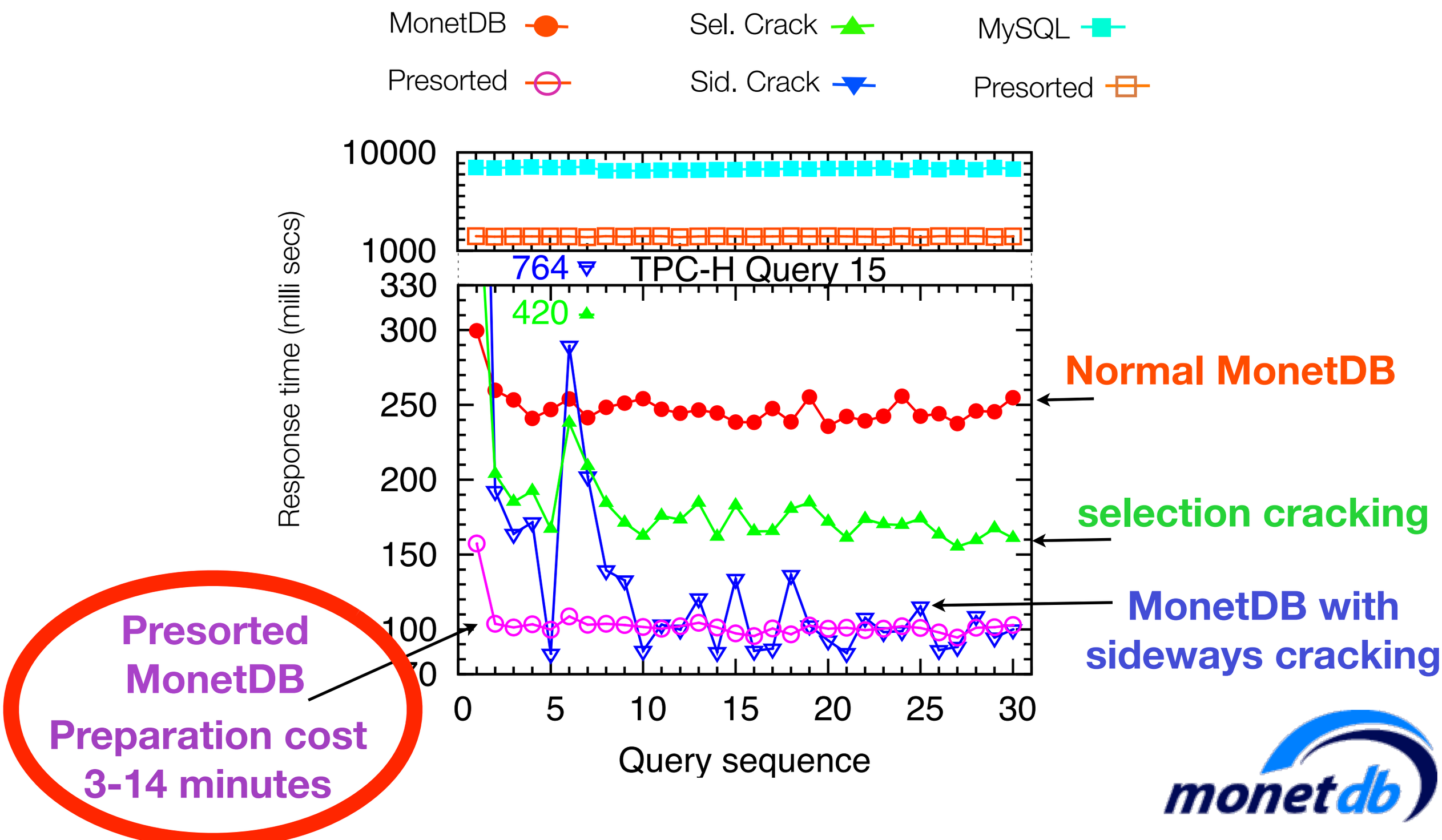
Self-organizing behavior (TPC-H)



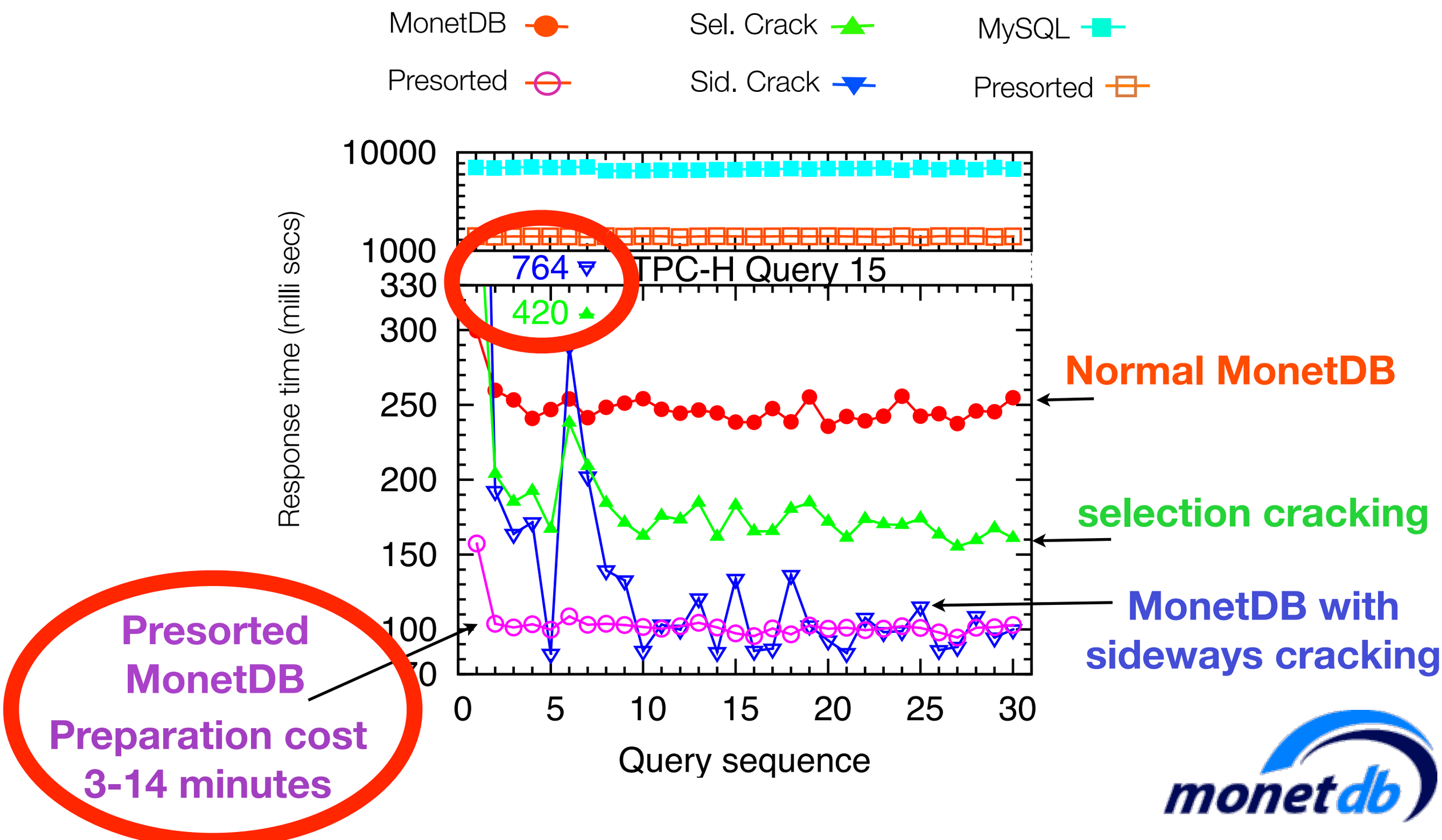
Self-organizing behavior (TPC-H)



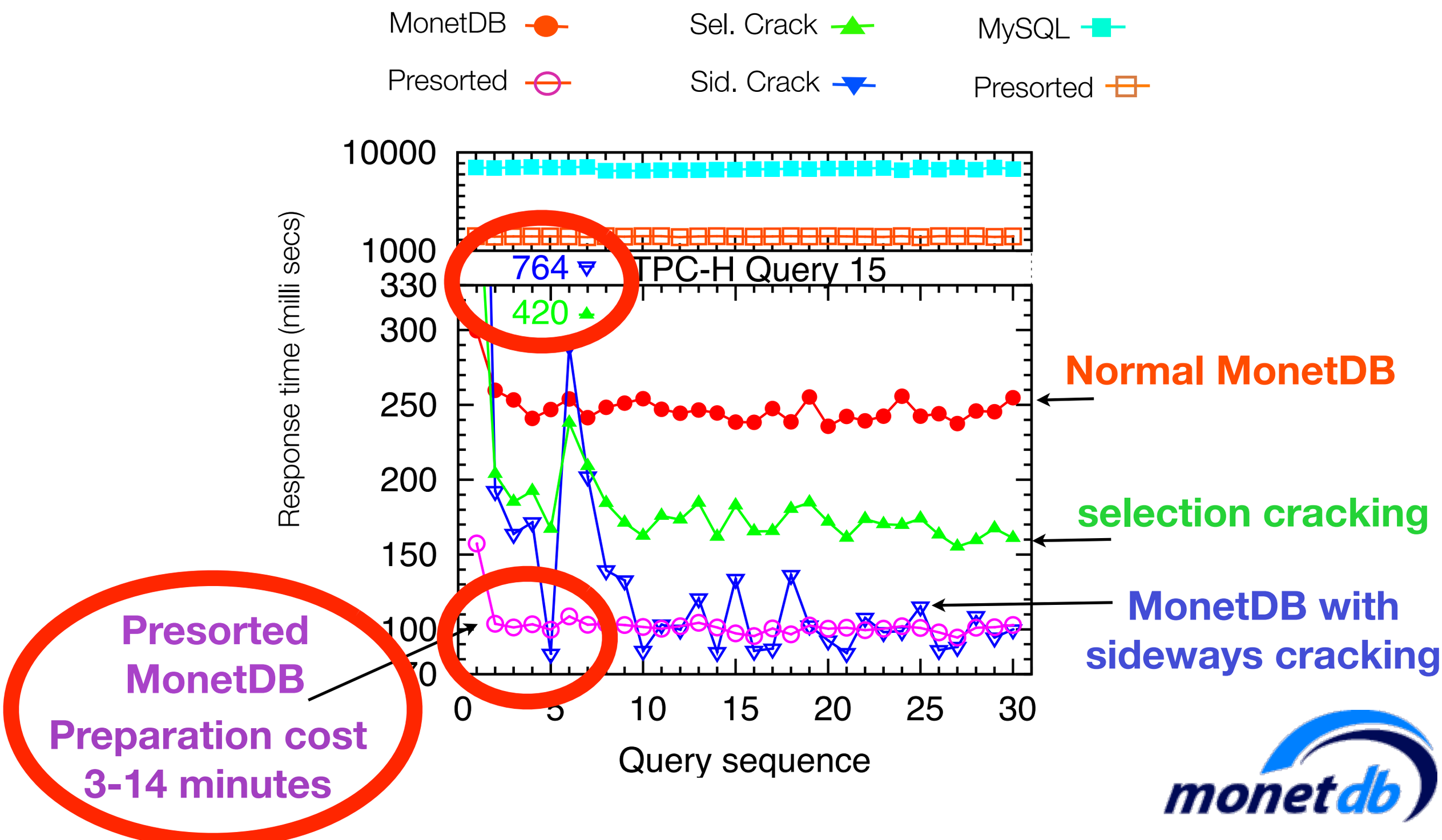
Self-organizing behavior (TPC-H)



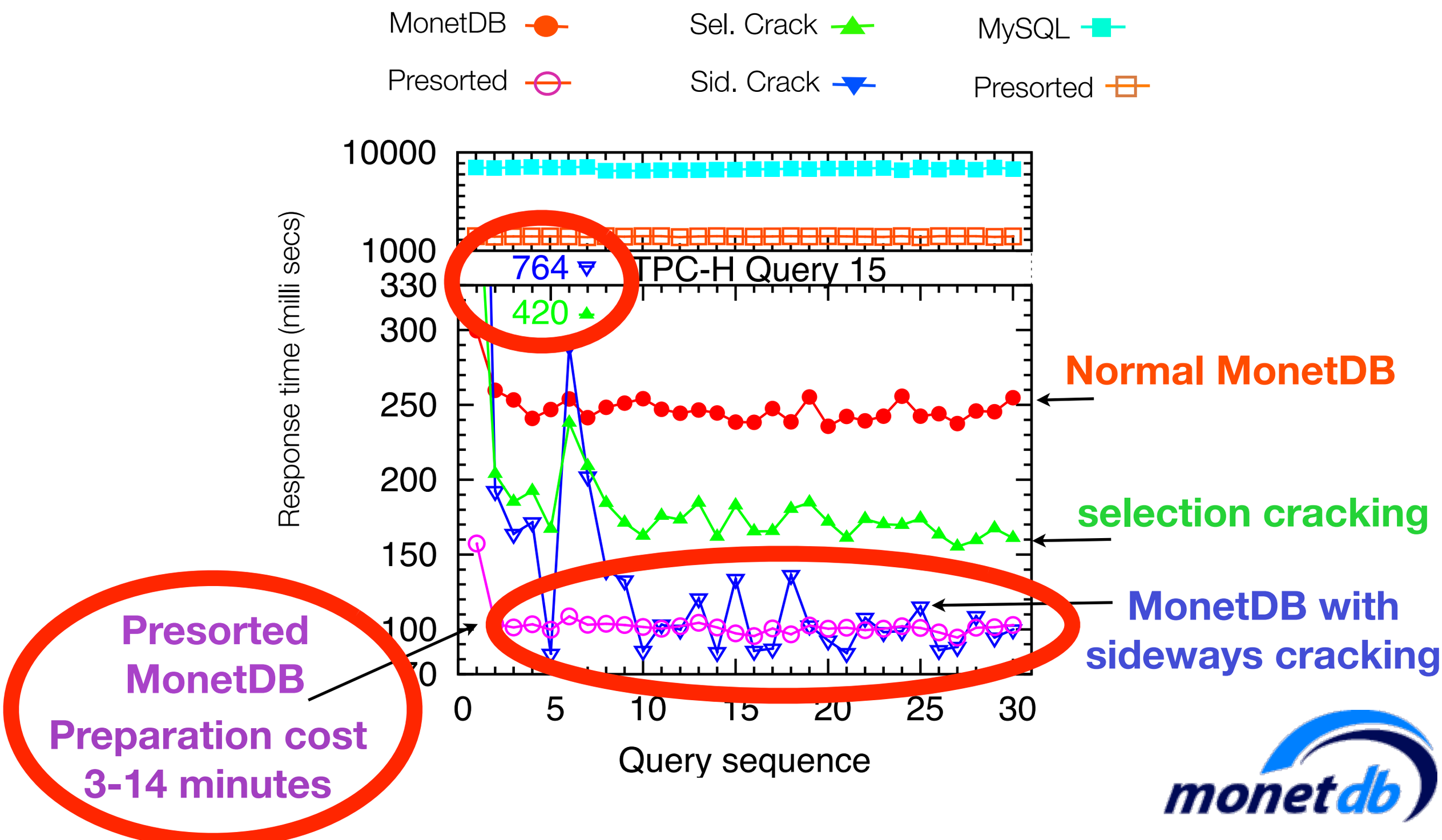
Self-organizing behavior (TPC-H)



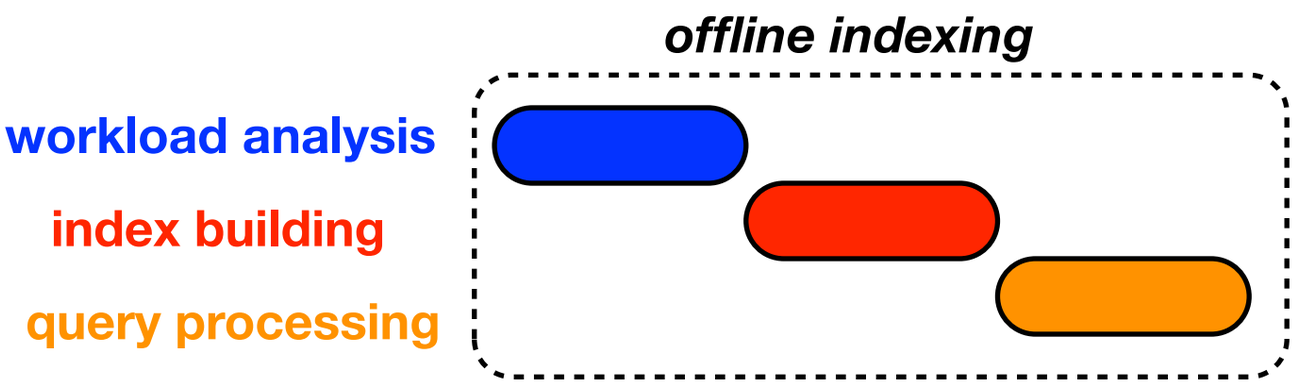
Self-organizing behavior (TPC-H)



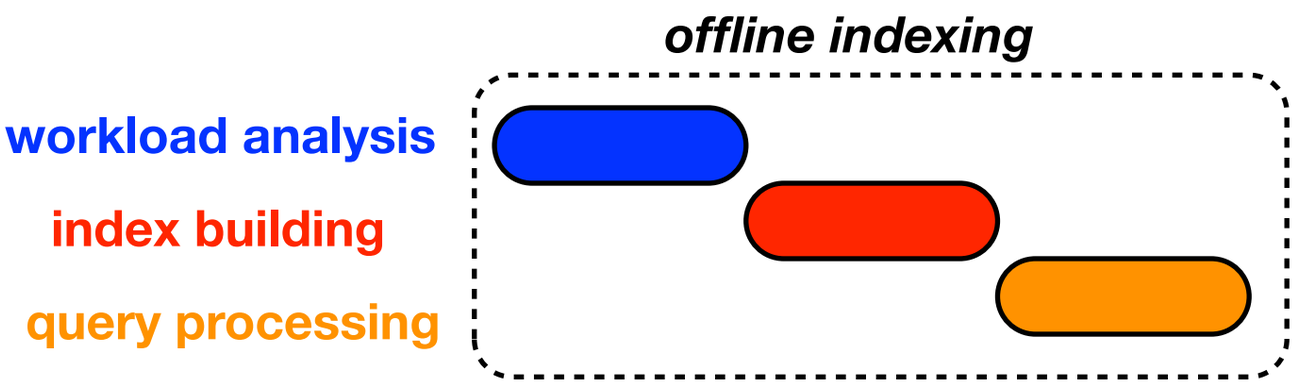
Self-organizing behavior (TPC-H)

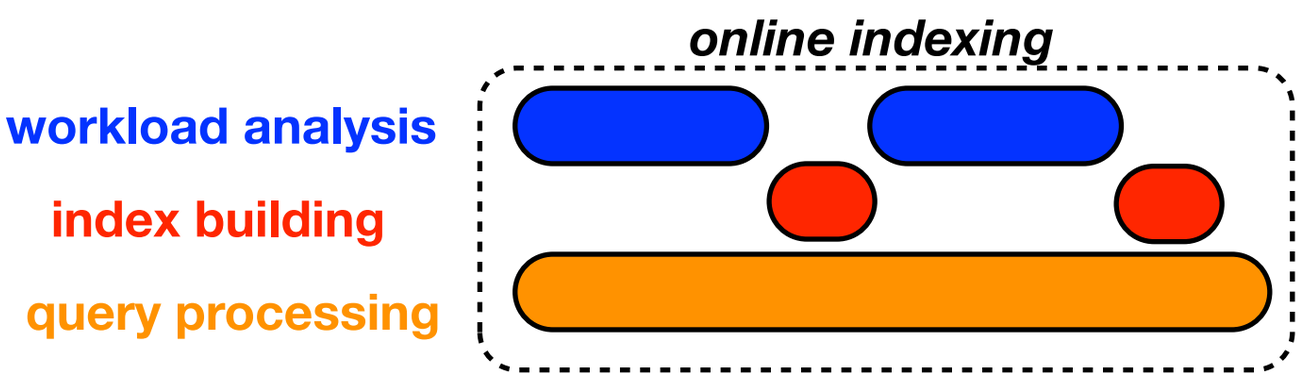


Indexing Overview

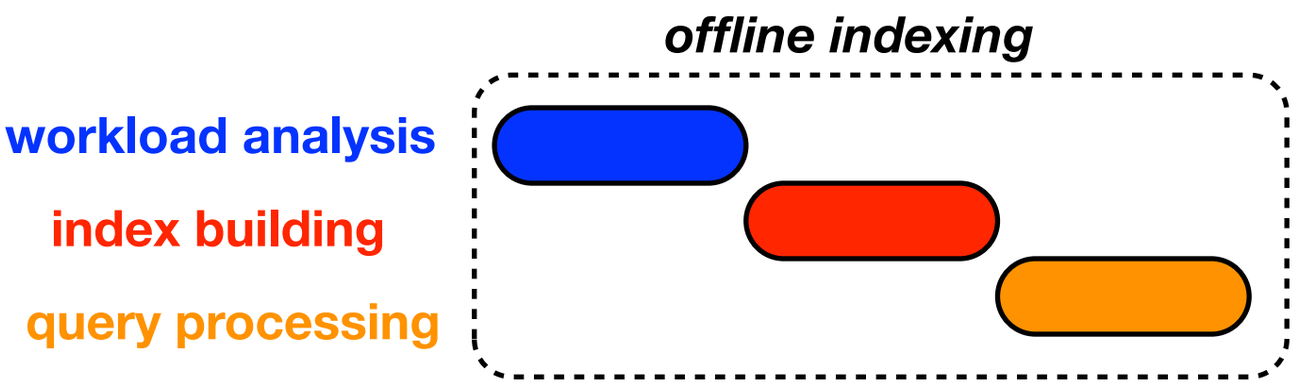


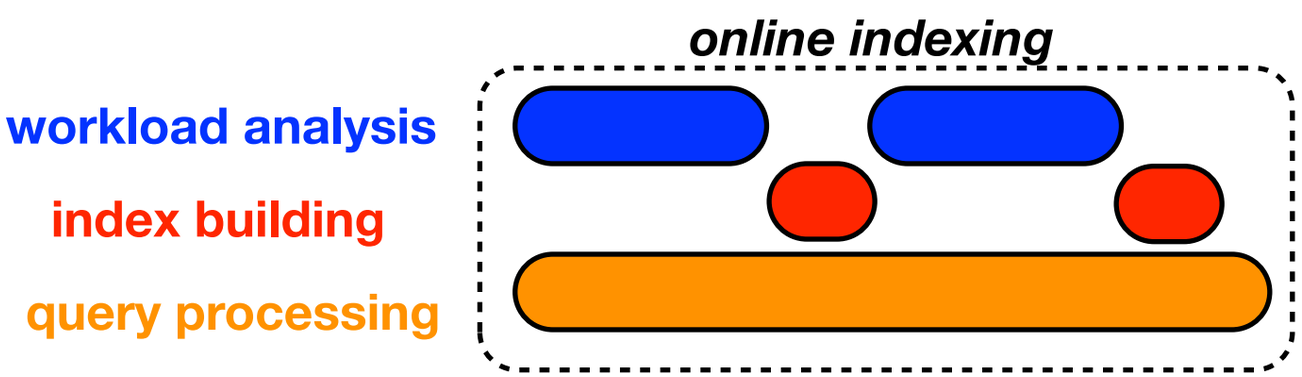
Indexing Overview



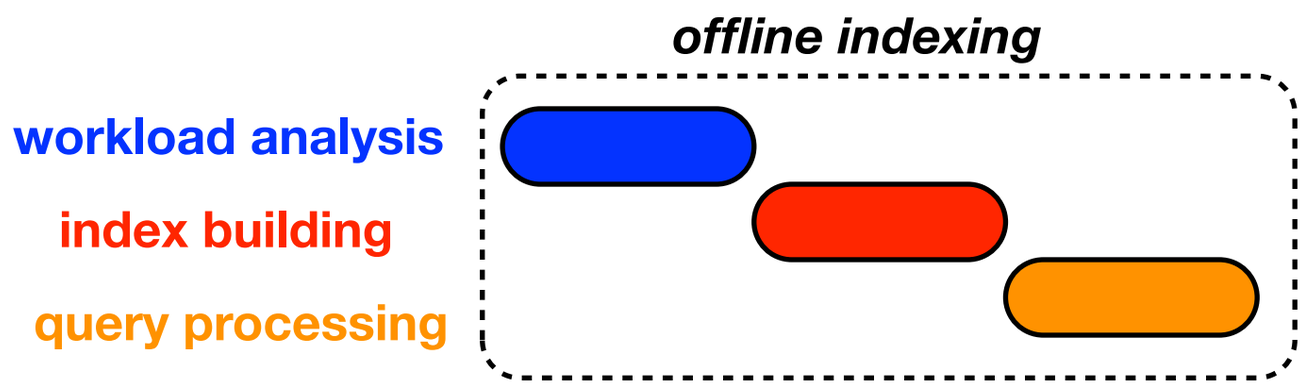


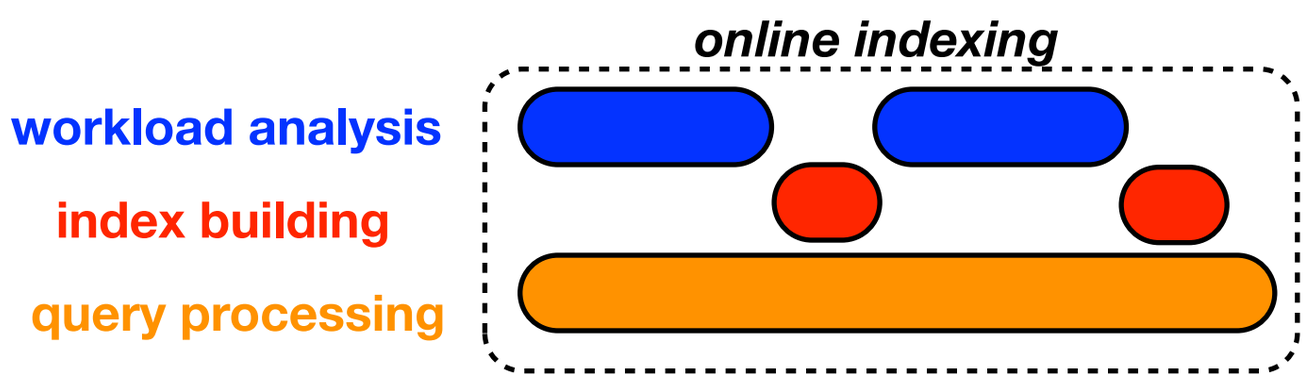
Indexing Overview

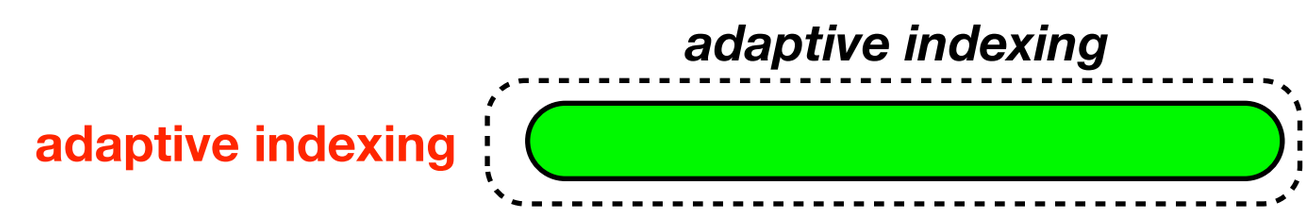


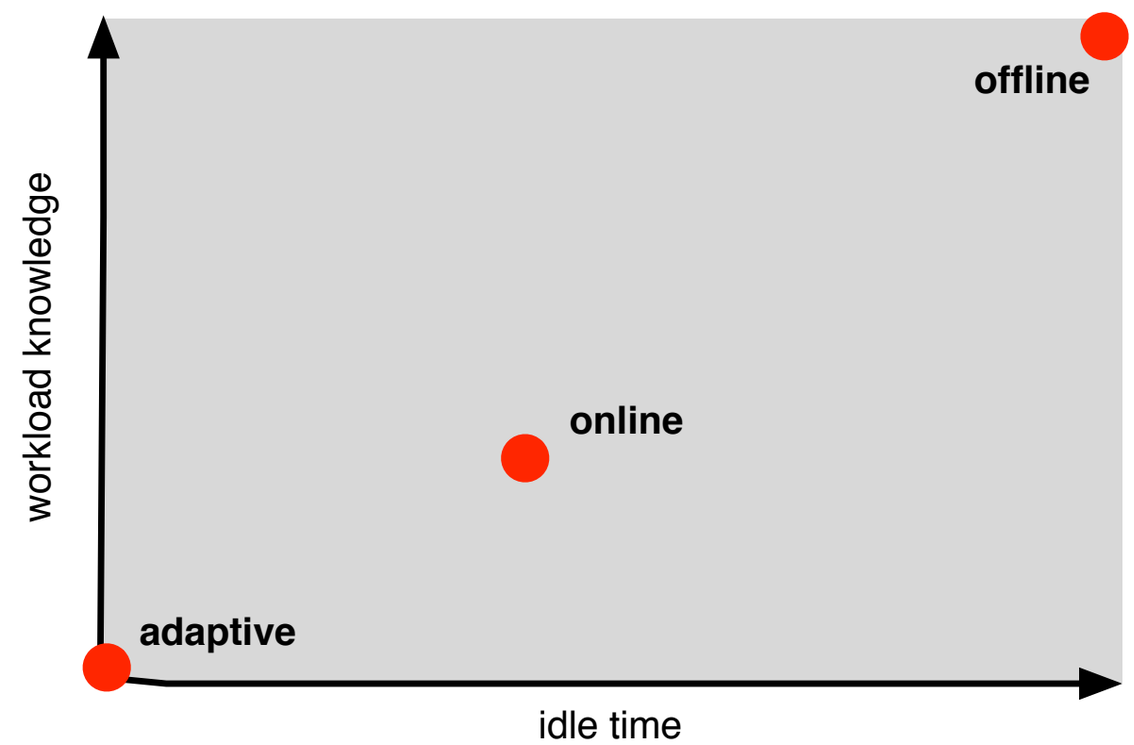


Indexing Overview









CWI

Database Cracking

Each query is treated as an advice on how data should be stored

CIDR'07 Selection cracking

SIGMOD'07 Updates

SIGMOD'09 Sideways and partial cracking

Can be thought of as an incremental quicksort

hp invent

monetdb

CWI

Database Cracking

Each query is treated as an advice on how data should be stored

CIDR'07 Selection cracking

SIGMOD'07 Updates

SIGMOD'09 Sideways and partial cracking

Can be thought of as an incremental quicksort

The core cracking algorithm is extremely lazy

hp invent

monetdb

CWI

Adaptive Merging

EDBT'10, **SMDB'10**, Goetz Graefe and Harumi Kuno

Incremental sort via external merge sort steps

invent

monetdb

CWI

Adaptive Merging

EDBT'10, **SMDB'10**, Goetz Graefe and Harumi Kuno

Incremental sort via external merge sort steps

Adaptive Merging

EDBT'10, **SMDB'10**, Goetz Graefe and Harumi Kuno

Incremental sort via external merge sort steps

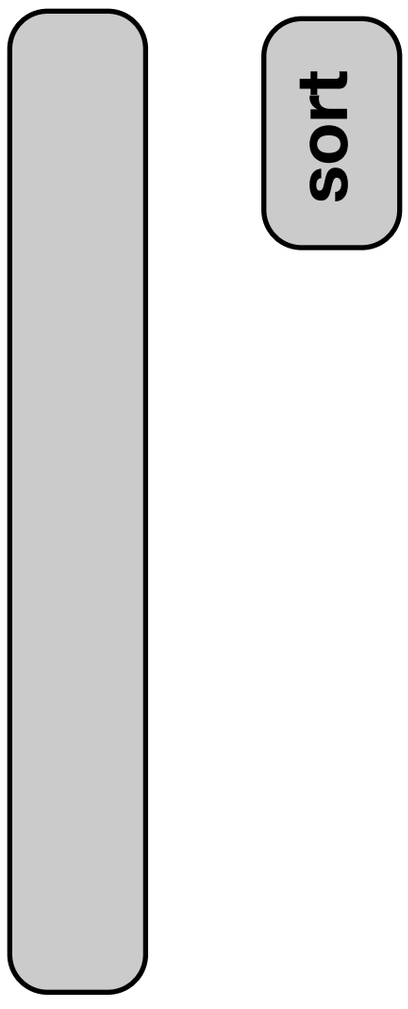
select(A,50,100)

Adaptive Merging

EDBT'10, **SMDB'10**, Goetz Graefe and Harumi Kuno

Incremental sort via external merge sort steps

select(A,50,100)

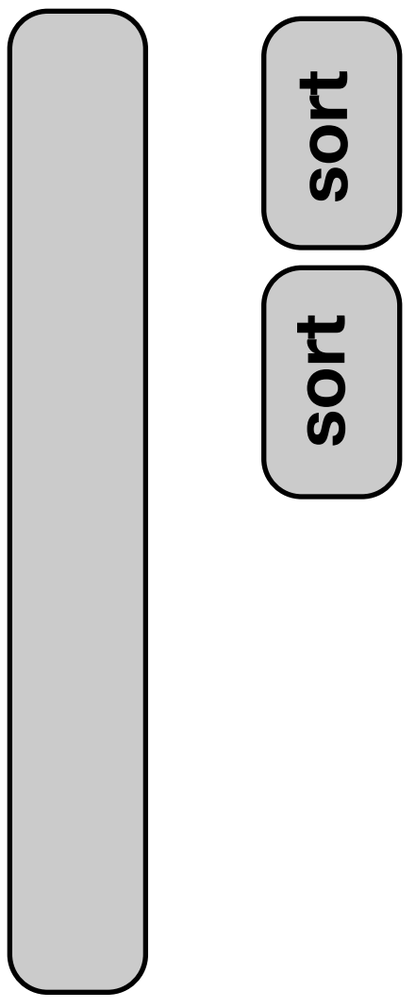


Adaptive Merging

EDBT'10, **SMDB'10**, Goetz Graefe and Harumi Kuno

Incremental sort via external merge sort steps

select(A,50,100)

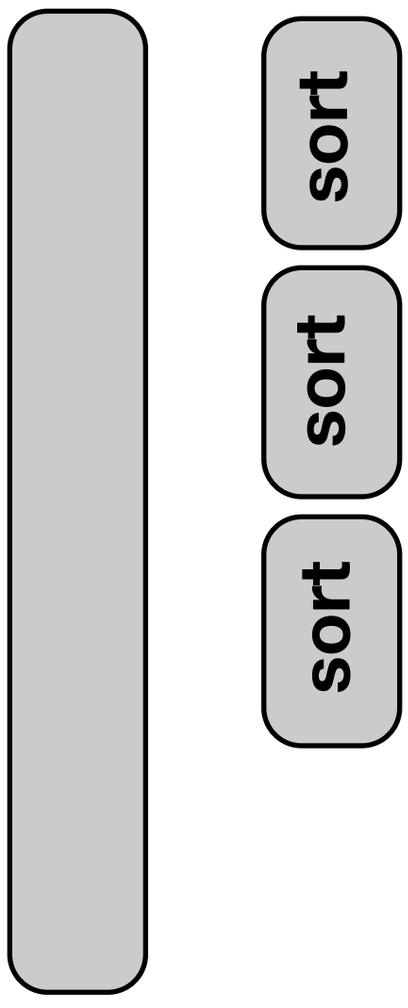


Adaptive Merging

EDBT'10, **SMDB'10**, Goetz Graefe and Harumi Kuno

Incremental sort via external merge sort steps

select(A,50,100)

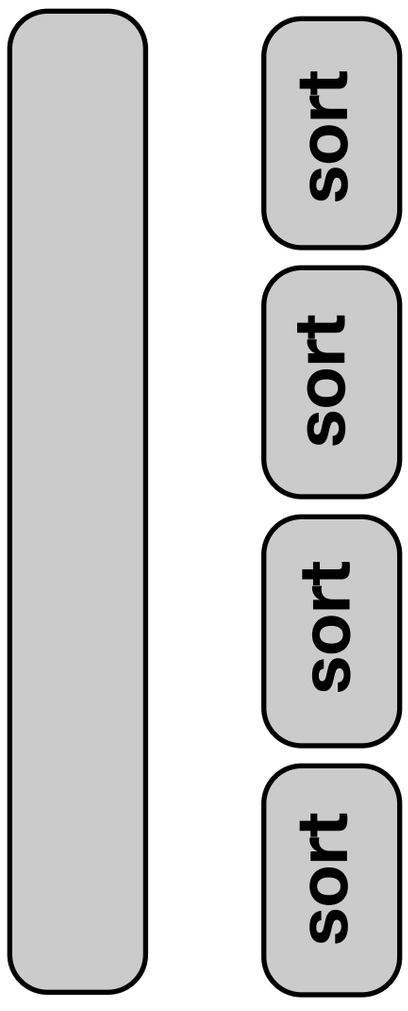


Adaptive Merging

EDBT'10, **SMDB'10**, Goetz Graefe and Harumi Kuno

Incremental sort via external merge sort steps

select(A,50,100)



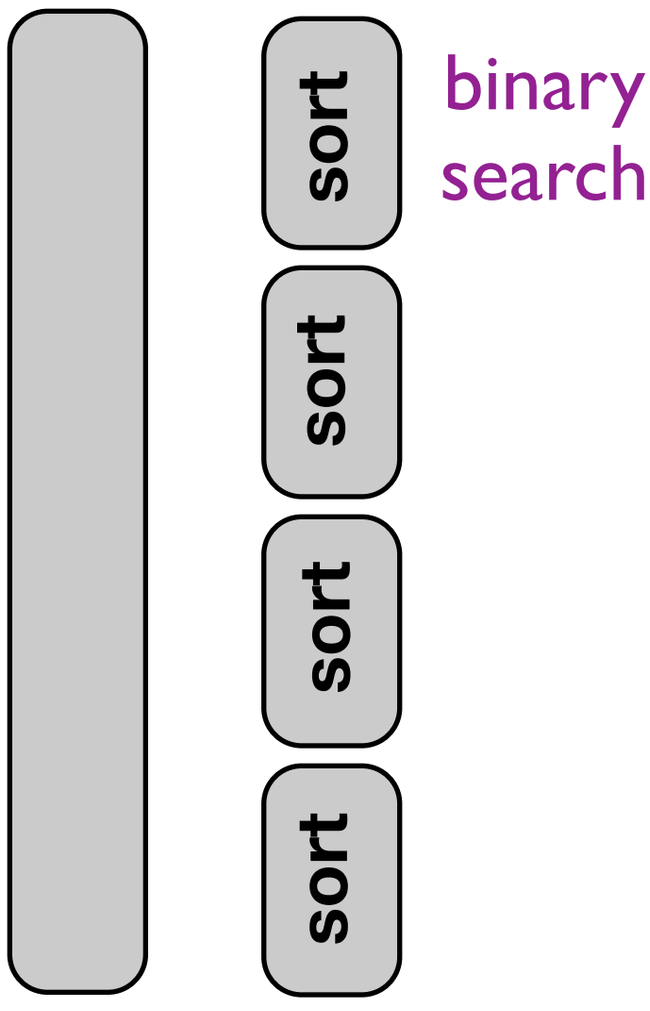
CWI

Adaptive Merging

EDBT'10, **SMDB'10**, Goetz Graefe and Harumi Kuno

Incremental sort via external merge sort steps

select(A,50,100)



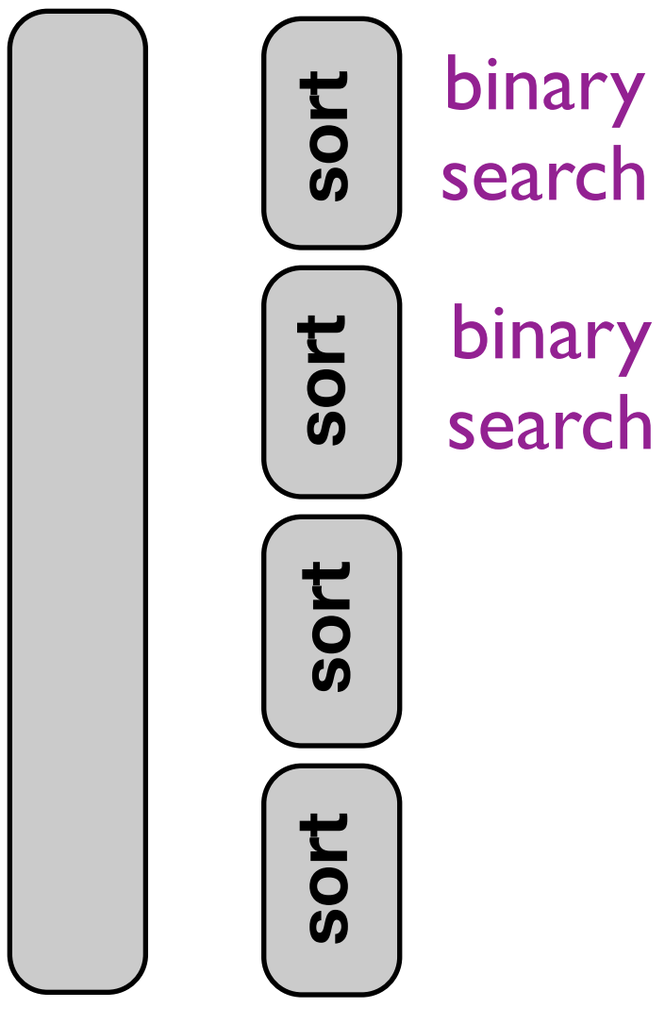
CWI

Adaptive Merging

EDBT'10, **SMDB'10**, Goetz Graefe and Harumi Kuno

Incremental sort via external merge sort steps

select(A,50,100)



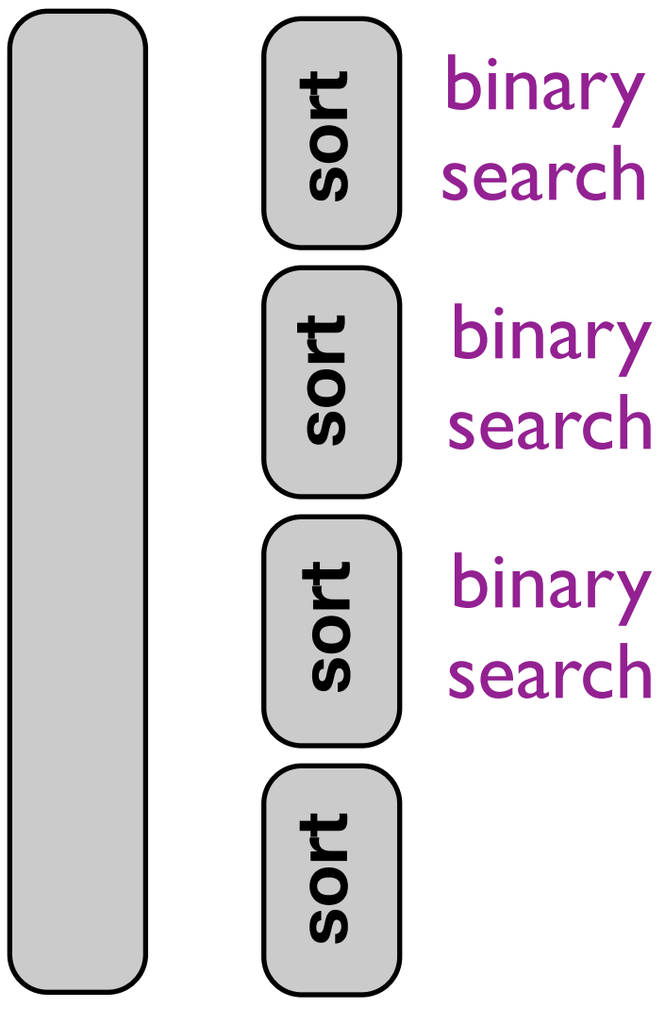
CWI

Adaptive Merging

EDBT'10, **SMDB'10**, Goetz Graefe and Harumi Kuno

Incremental sort via external merge sort steps

select(A,50,100)



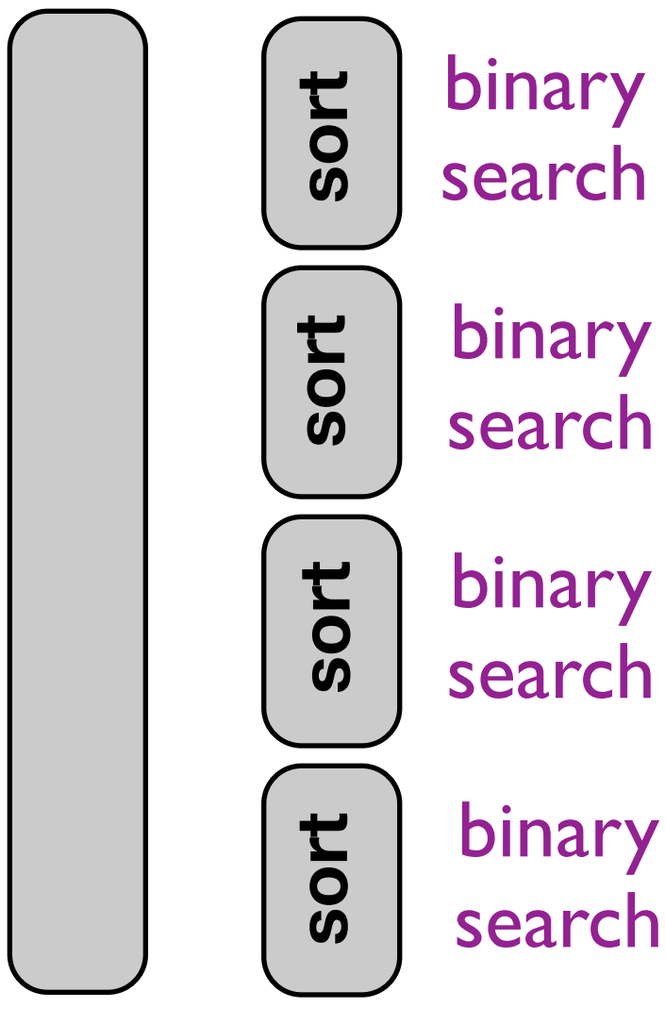
CWI

Adaptive Merging

EDBT'10, **SMDB'10**, Goetz Graefe and Harumi Kuno

Incremental sort via external merge sort steps

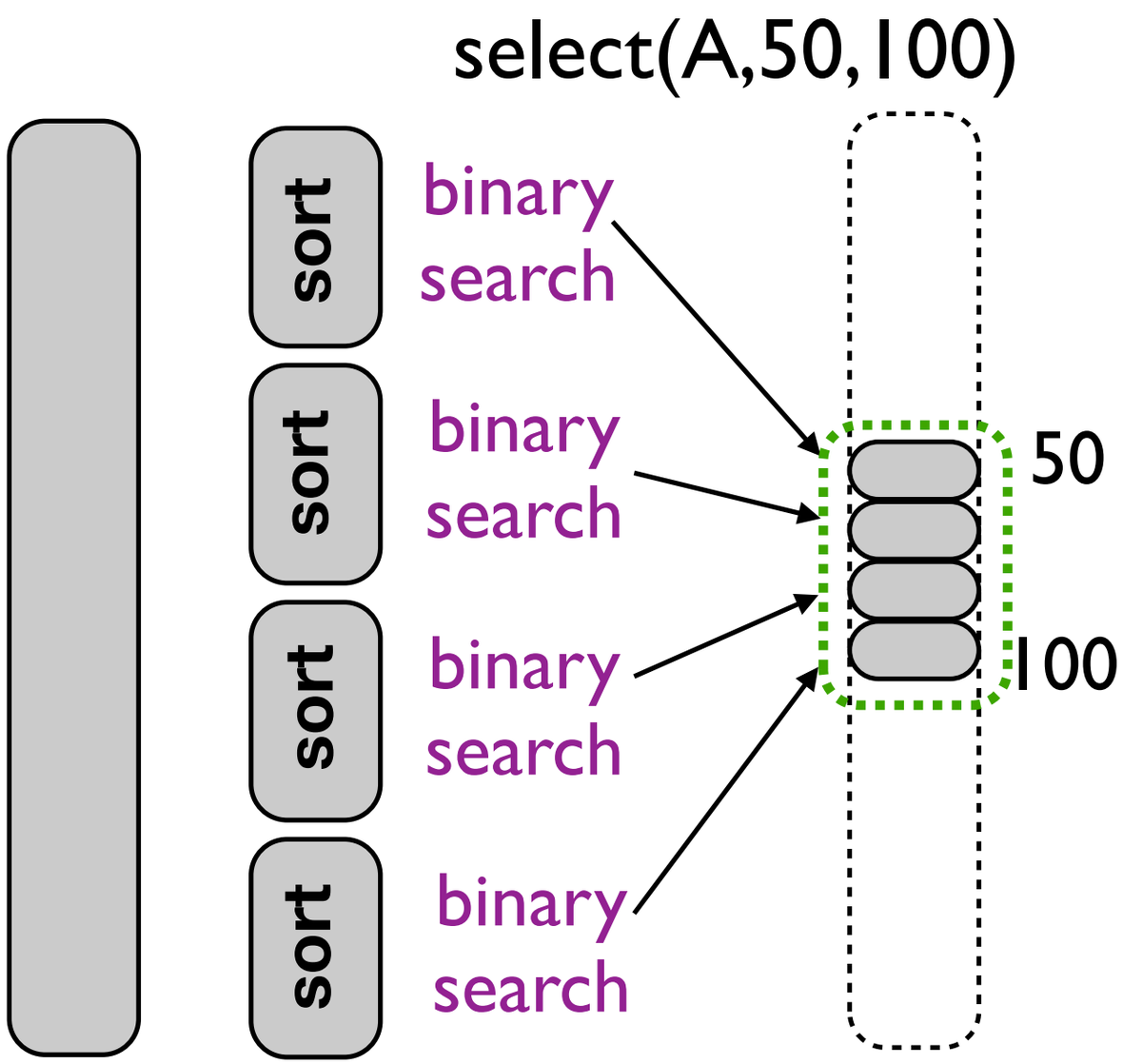
select(A,50,100)



Adaptive Merging

EDBT'10, **SMDB'10**, Goetz Graefe and Harumi Kuno

Incremental sort via external merge sort steps

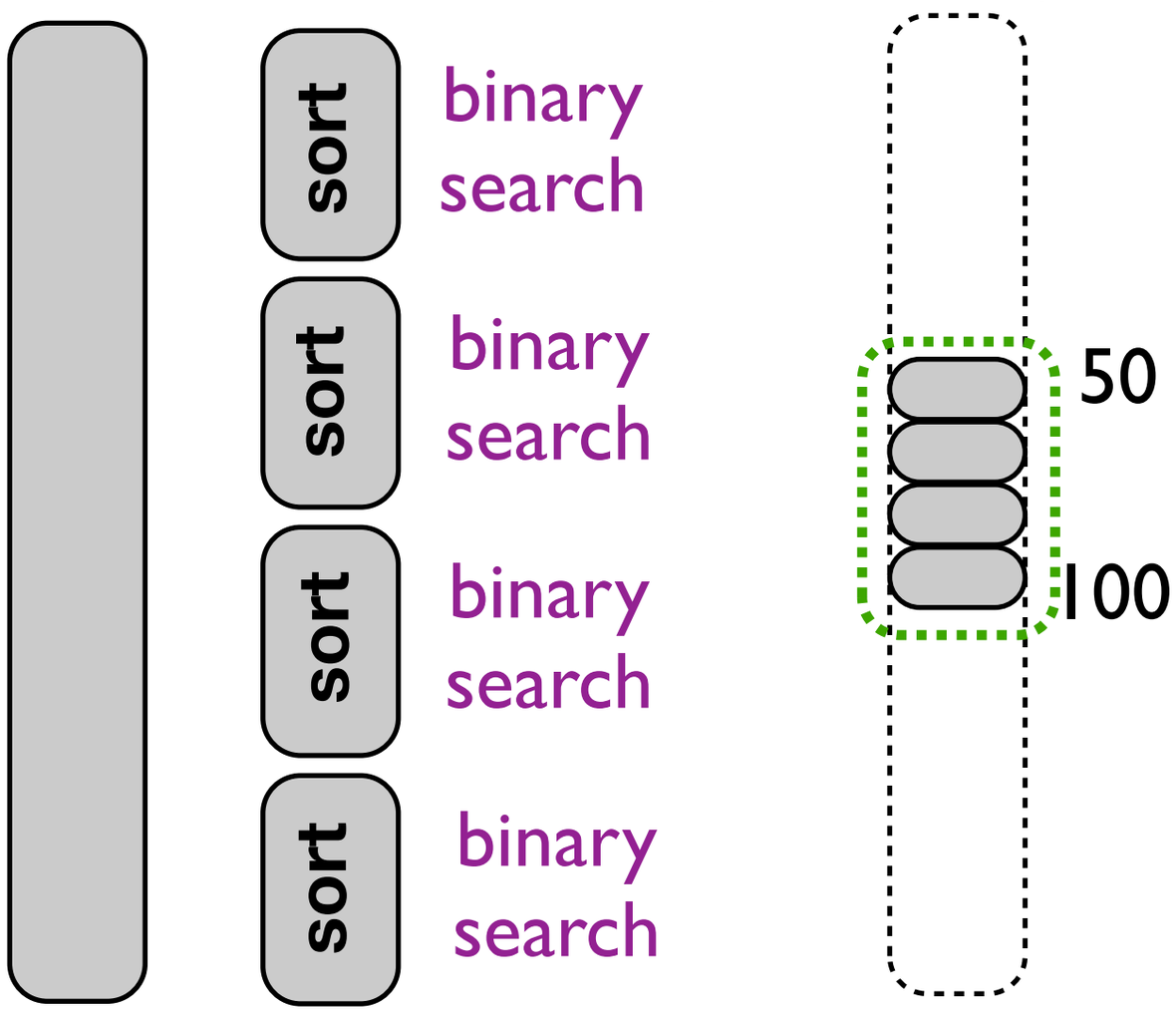


Adaptive Merging

EDBT'10, **SMDB'10**, Goetz Graefe and Harumi Kuno

Incremental sort via external merge sort steps

select(A,50,100)

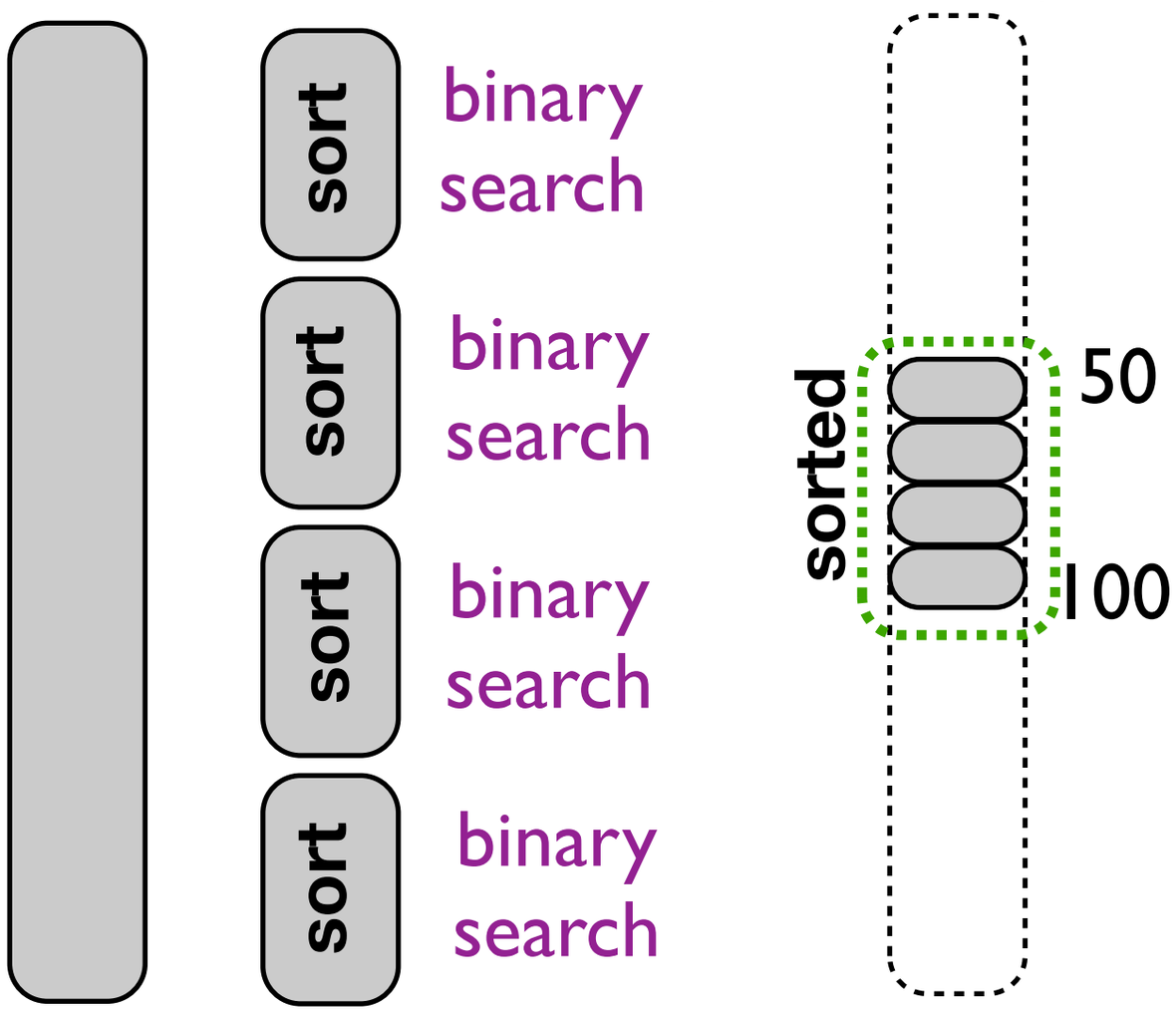


Adaptive Merging

EDBT'10, **SMDB'10**, Goetz Graefe and Harumi Kuno

Incremental sort via external merge sort steps

select(A,50,100)

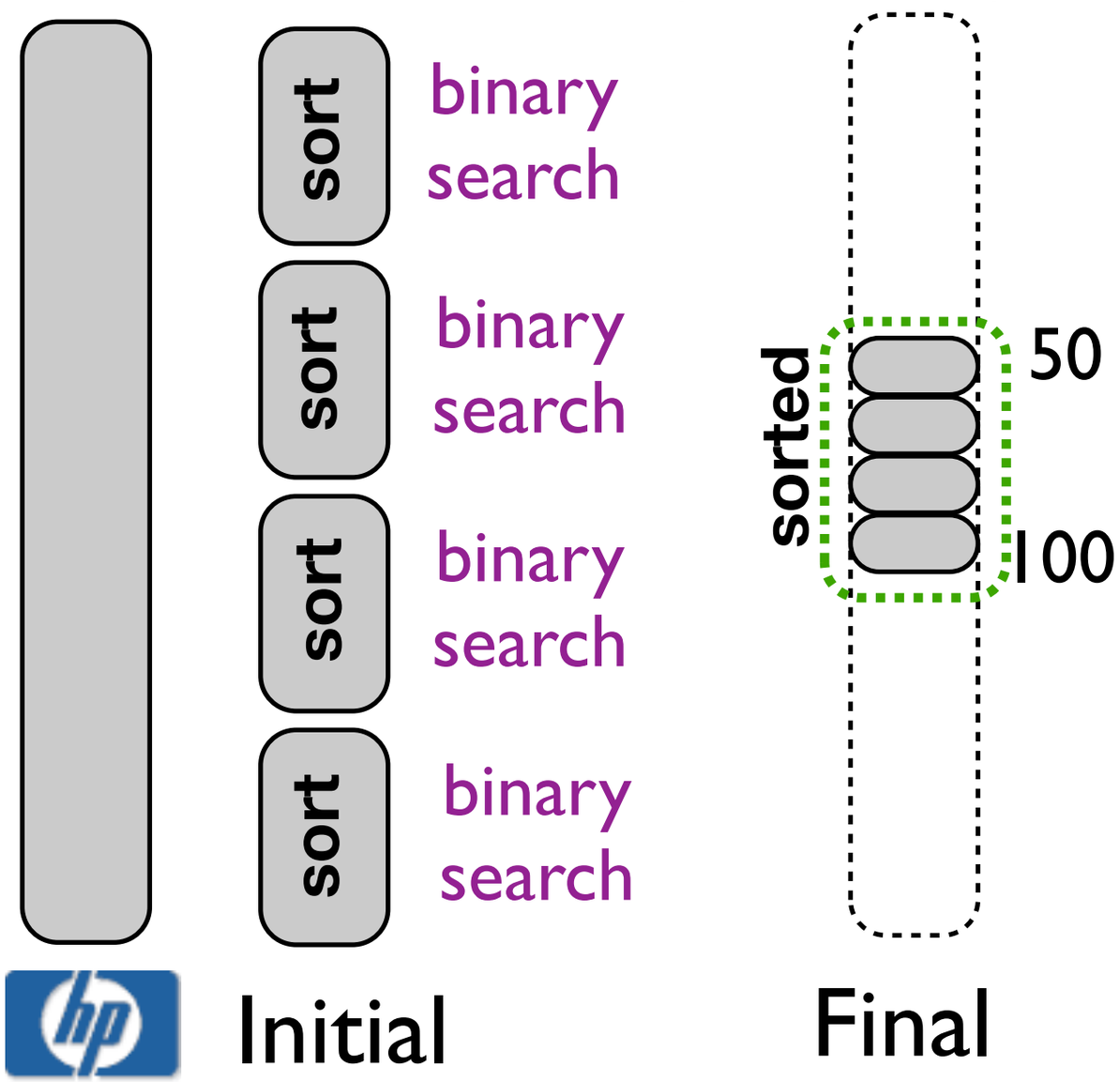


Adaptive Merging

EDBT'10, **SMDB'10**, Goetz Graefe and Harumi Kuno

Incremental sort via external merge sort steps

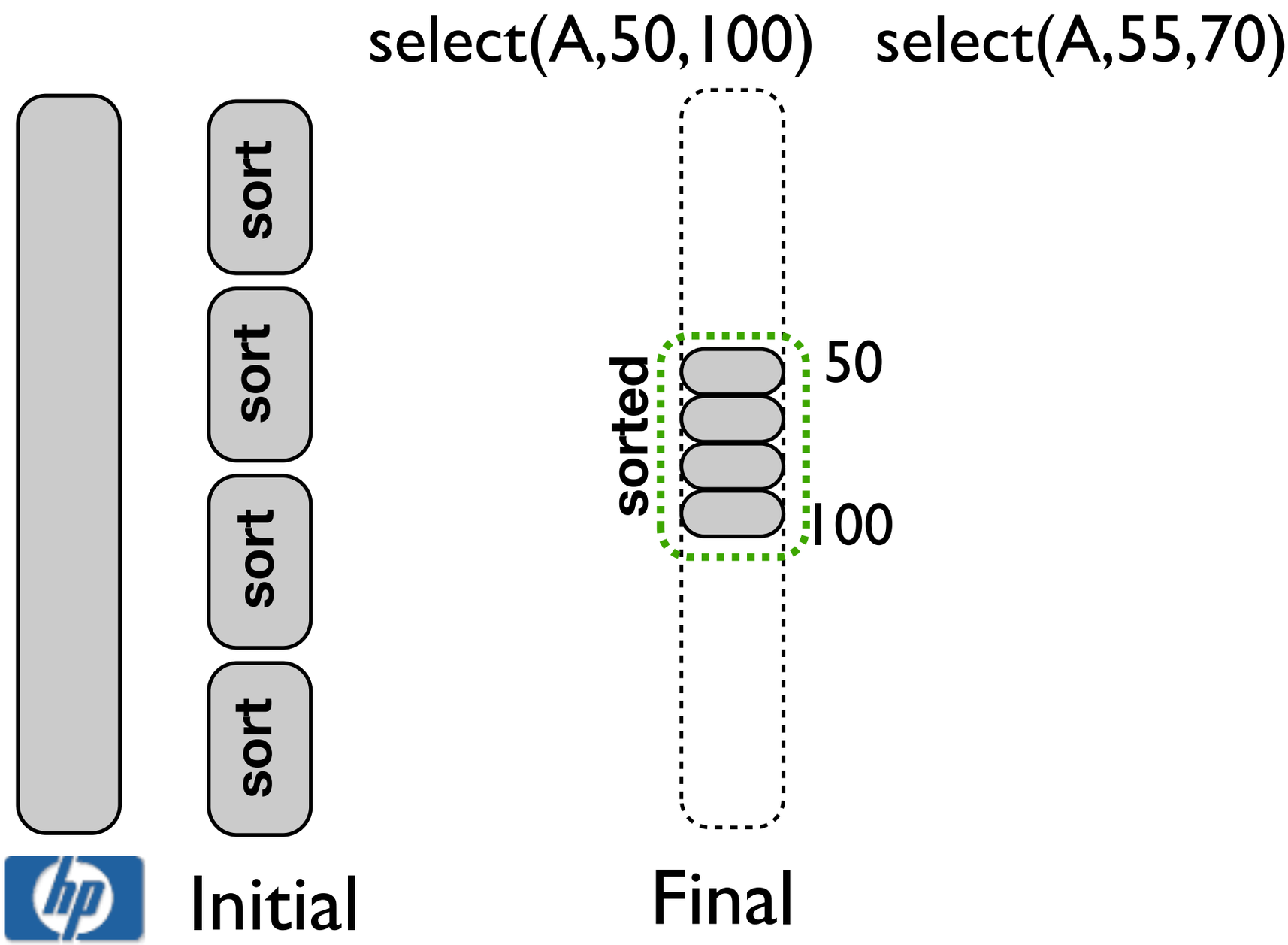
select(A,50,100)



Adaptive Merging

EDBT'10, **SMDB'10**, Goetz Graefe and Harumi Kuno

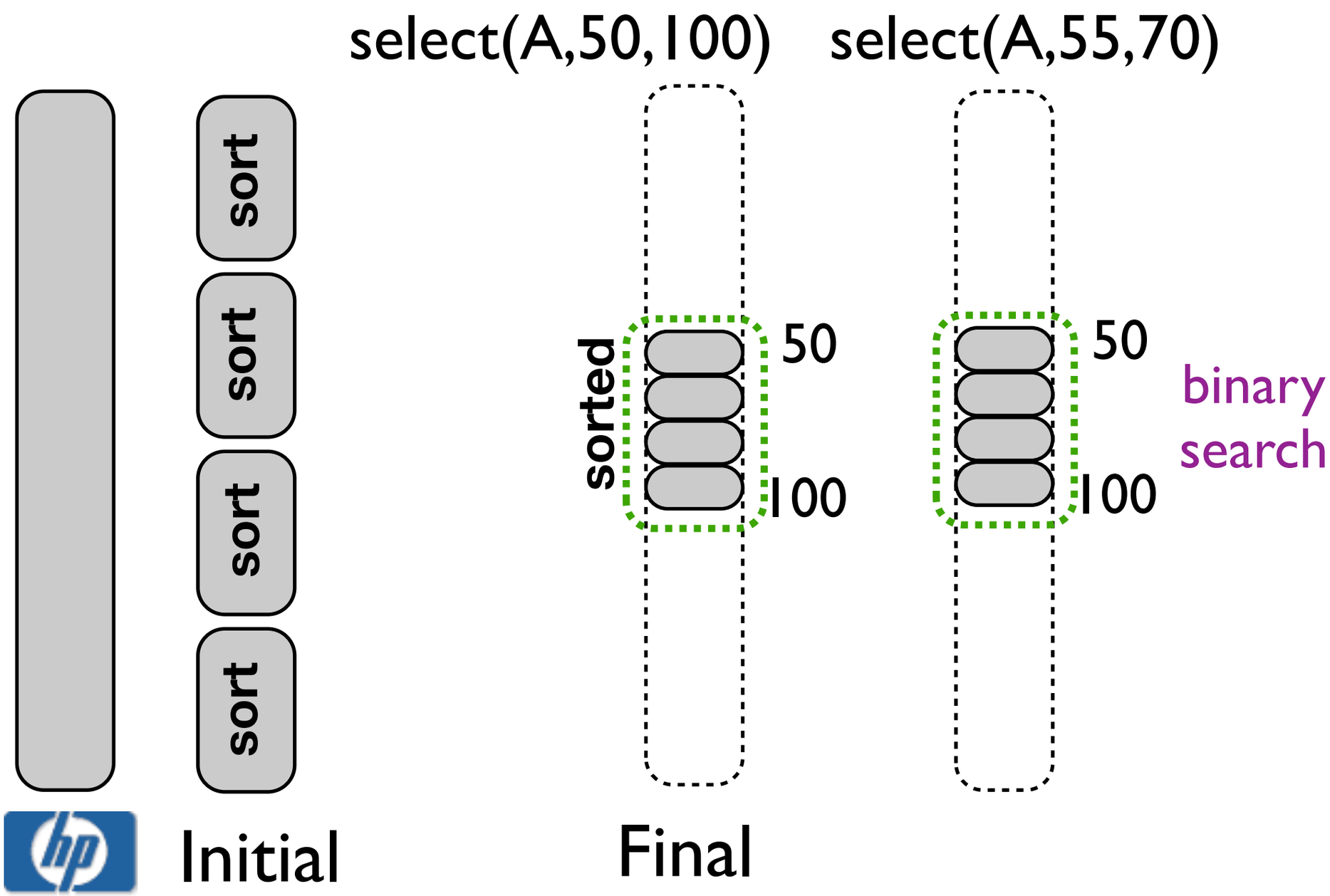
Incremental sort via external merge sort steps



Adaptive Merging

EDBT'10, **SMDB'10**, Goetz Graefe and Harumi Kuno

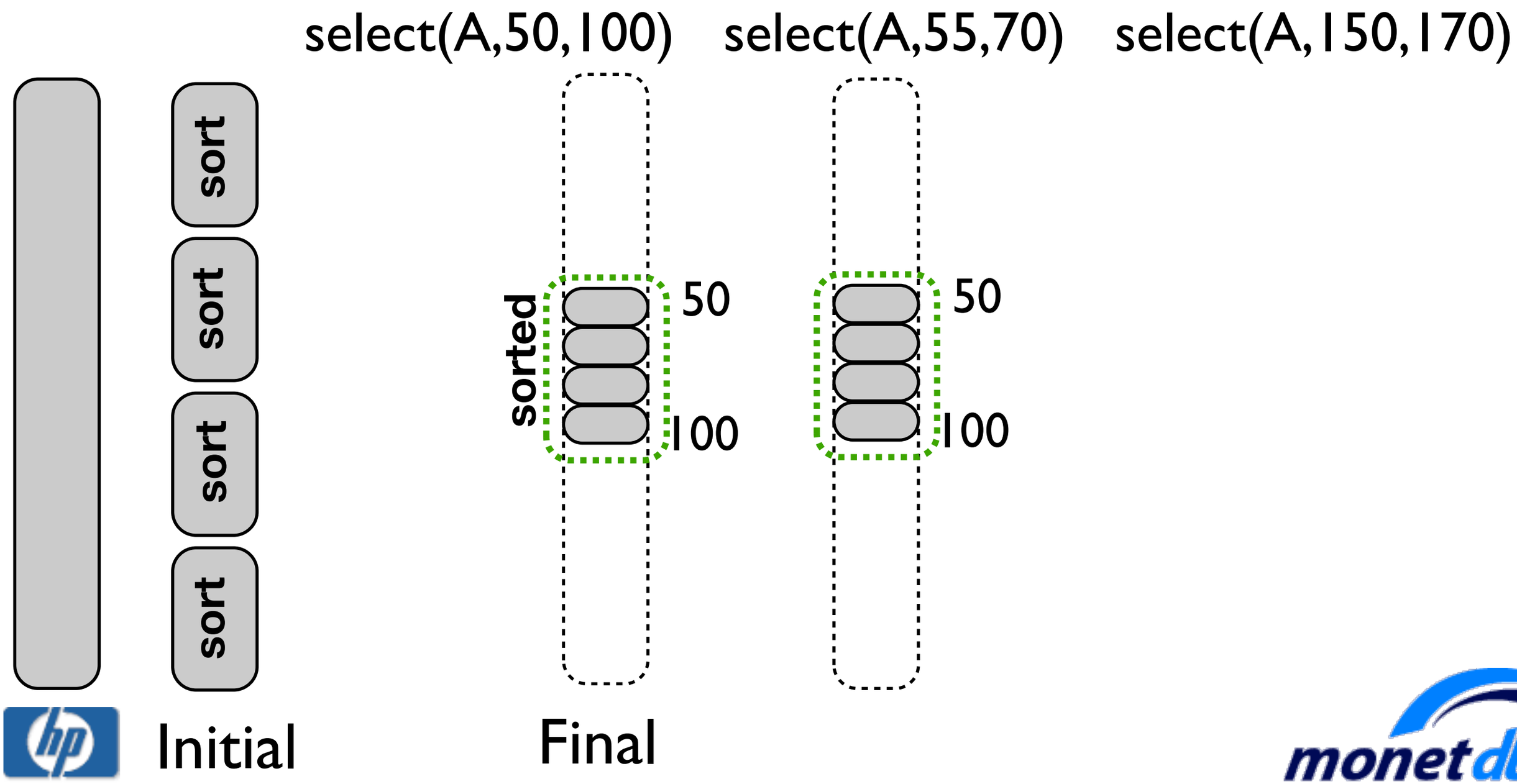
Incremental sort via external merge sort steps



Adaptive Merging

EDBT'10, **SMDB'10**, Goetz Graefe and Harumi Kuno

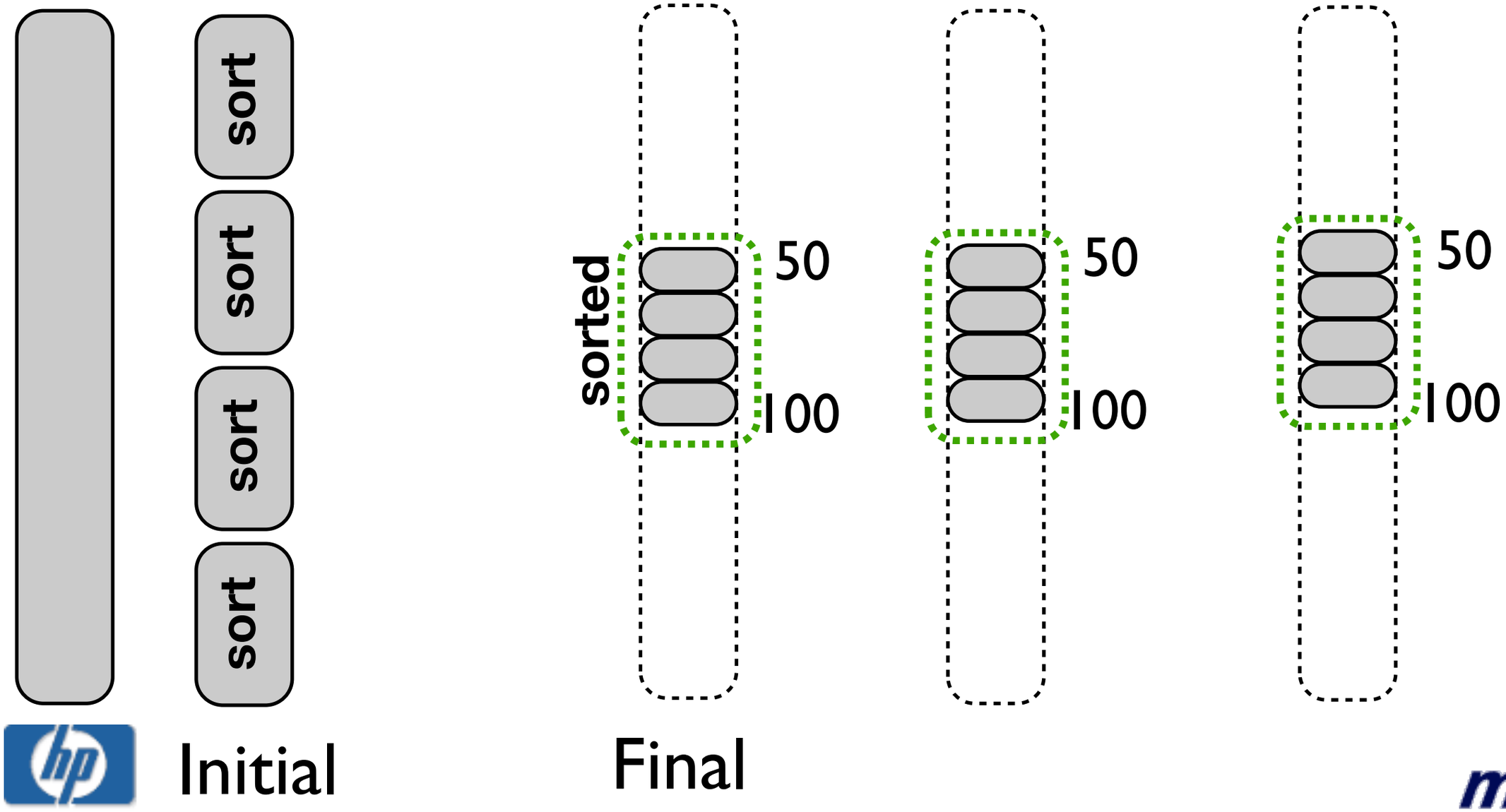
Incremental sort via external merge sort steps



Adaptive Merging

EDBT'10, **SMDB'10**, Goetz Graefe and Harumi Kuno

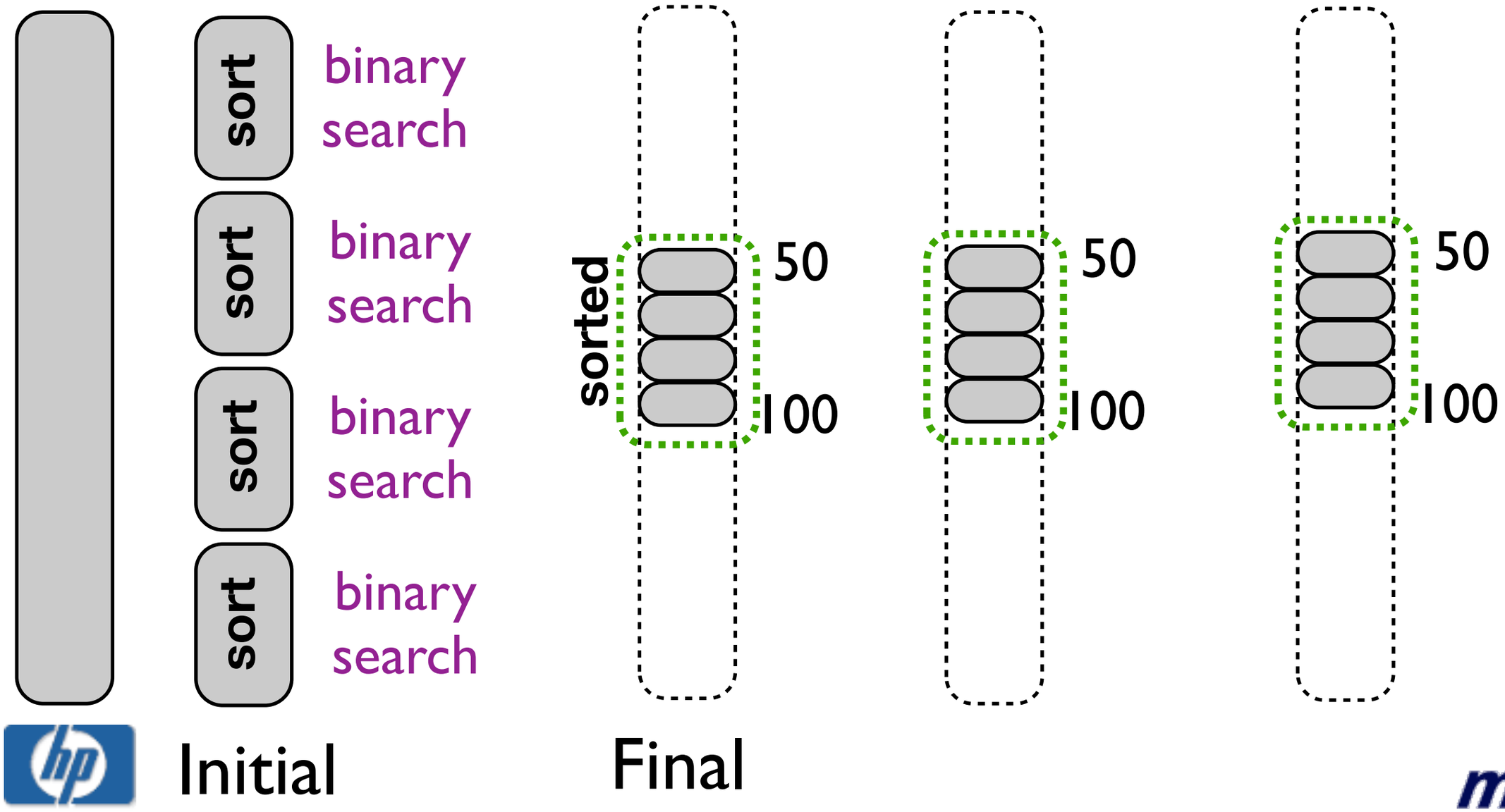
Incremental sort via external merge sort steps



Adaptive Merging

EDBT'10, **SMDB'10**, Goetz Graefe and Harumi Kuno

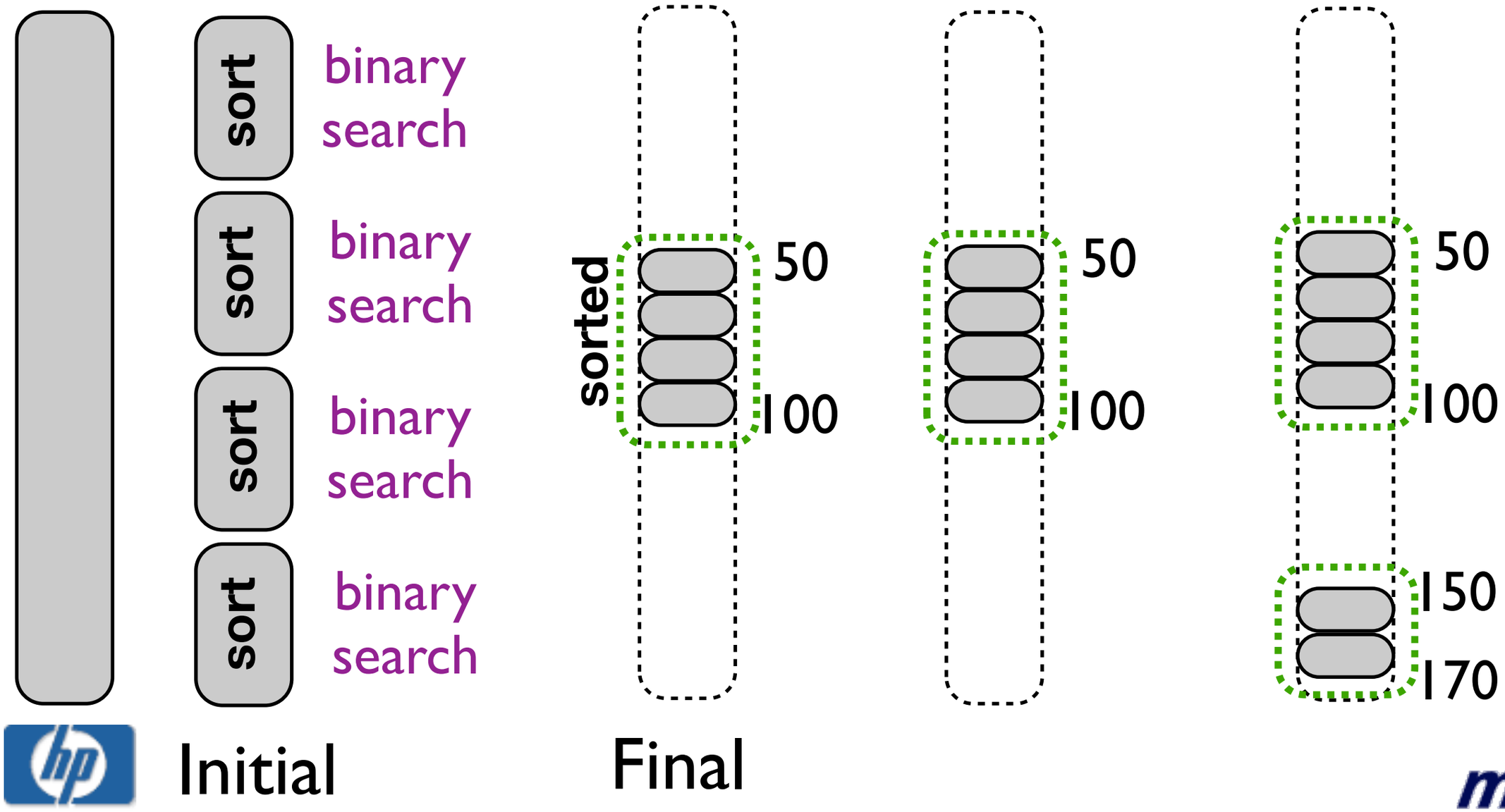
Incremental sort via external merge sort steps



Adaptive Merging

EDBT'10, **SMDB'10**, Goetz Graefe and Harumi Kuno

Incremental sort via external merge sort steps



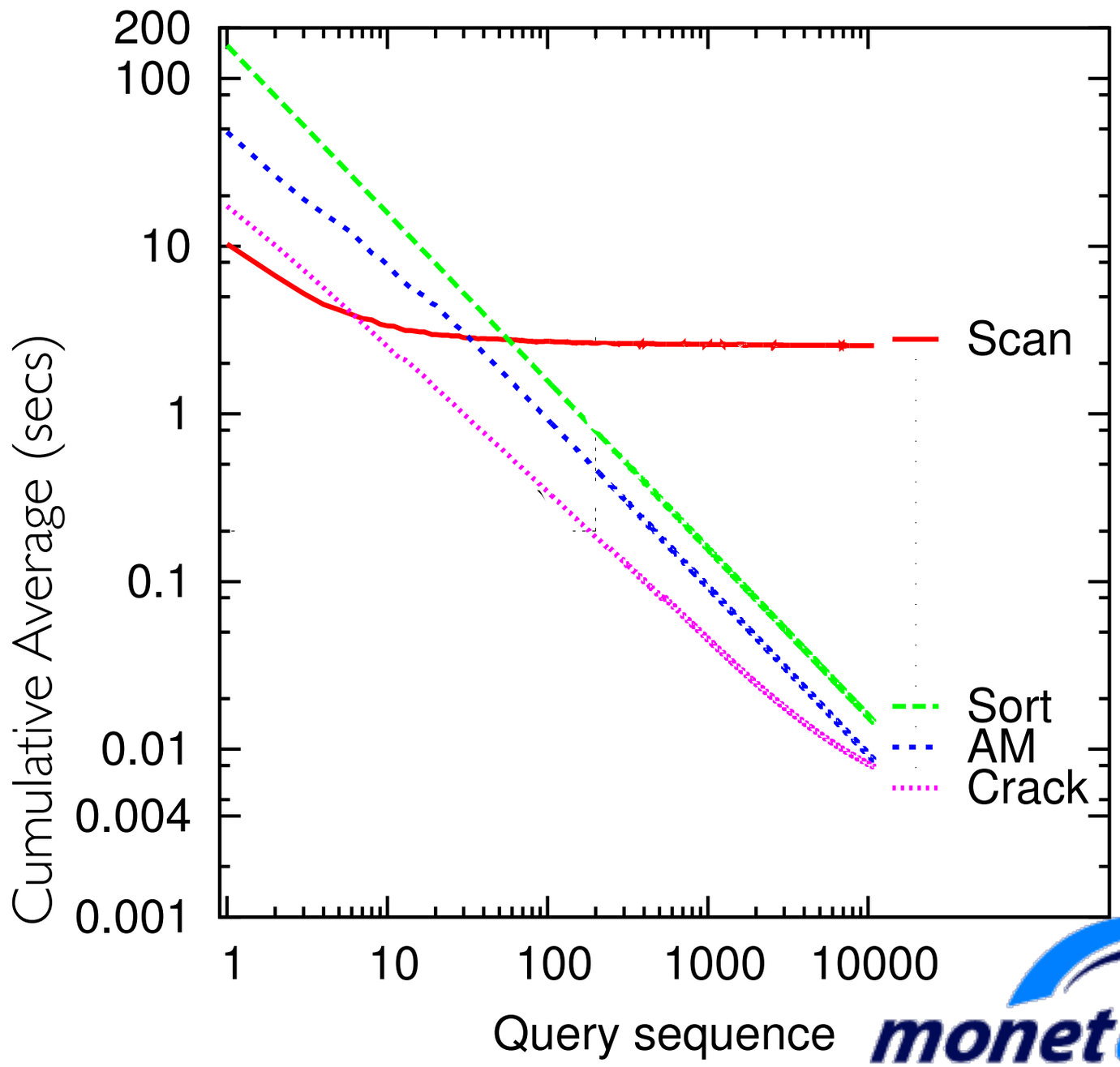
Questions

- Adaptive merging in column-stores?
- Adaptive merging Vs Cracking?
- Can we learn from both AM and Cracking?

Performance Analysis

set-up

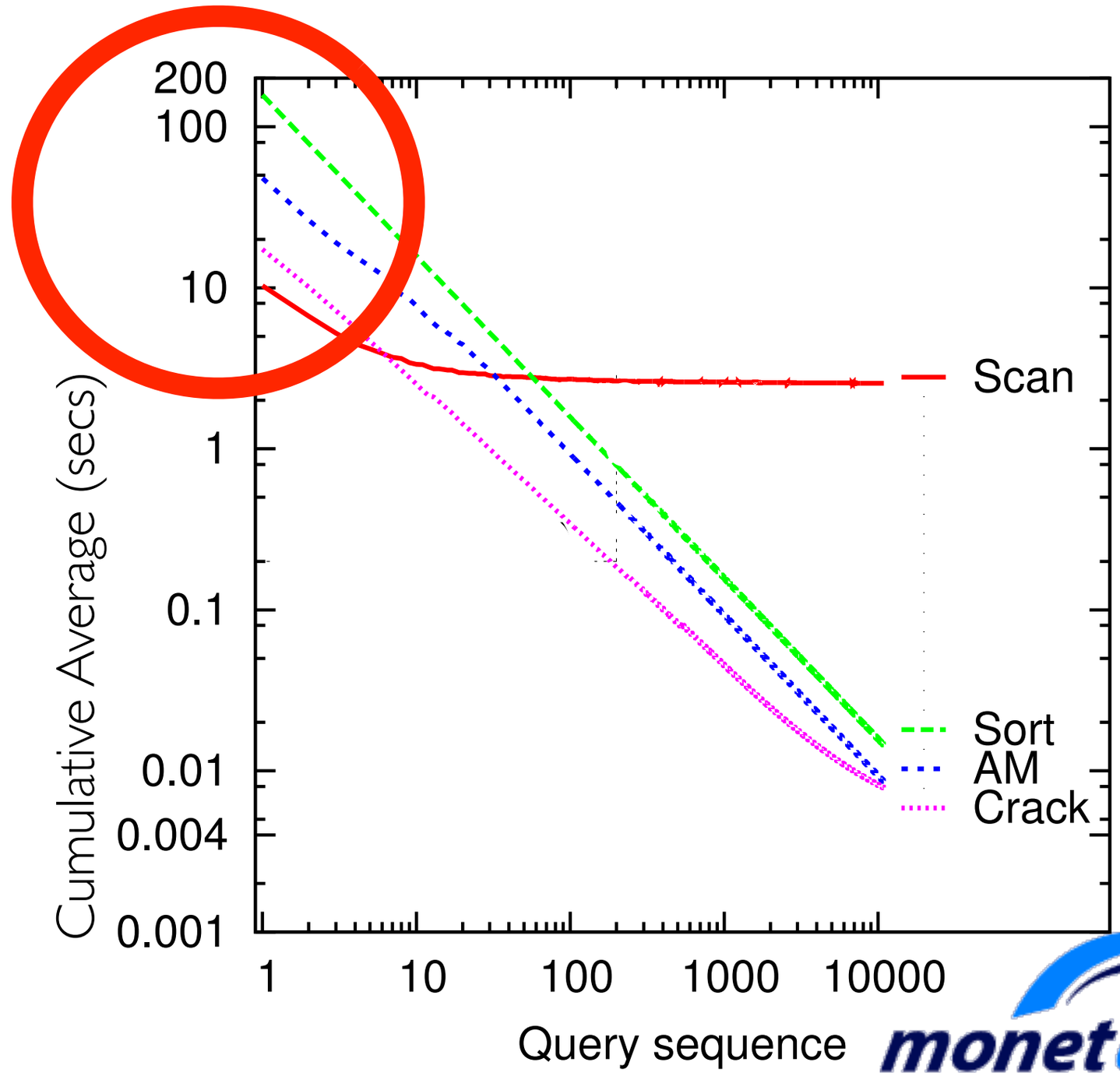
10K random selections
selectivity 10%
random value ranges
in a 30 million integer column



Performance Analysis

set-up

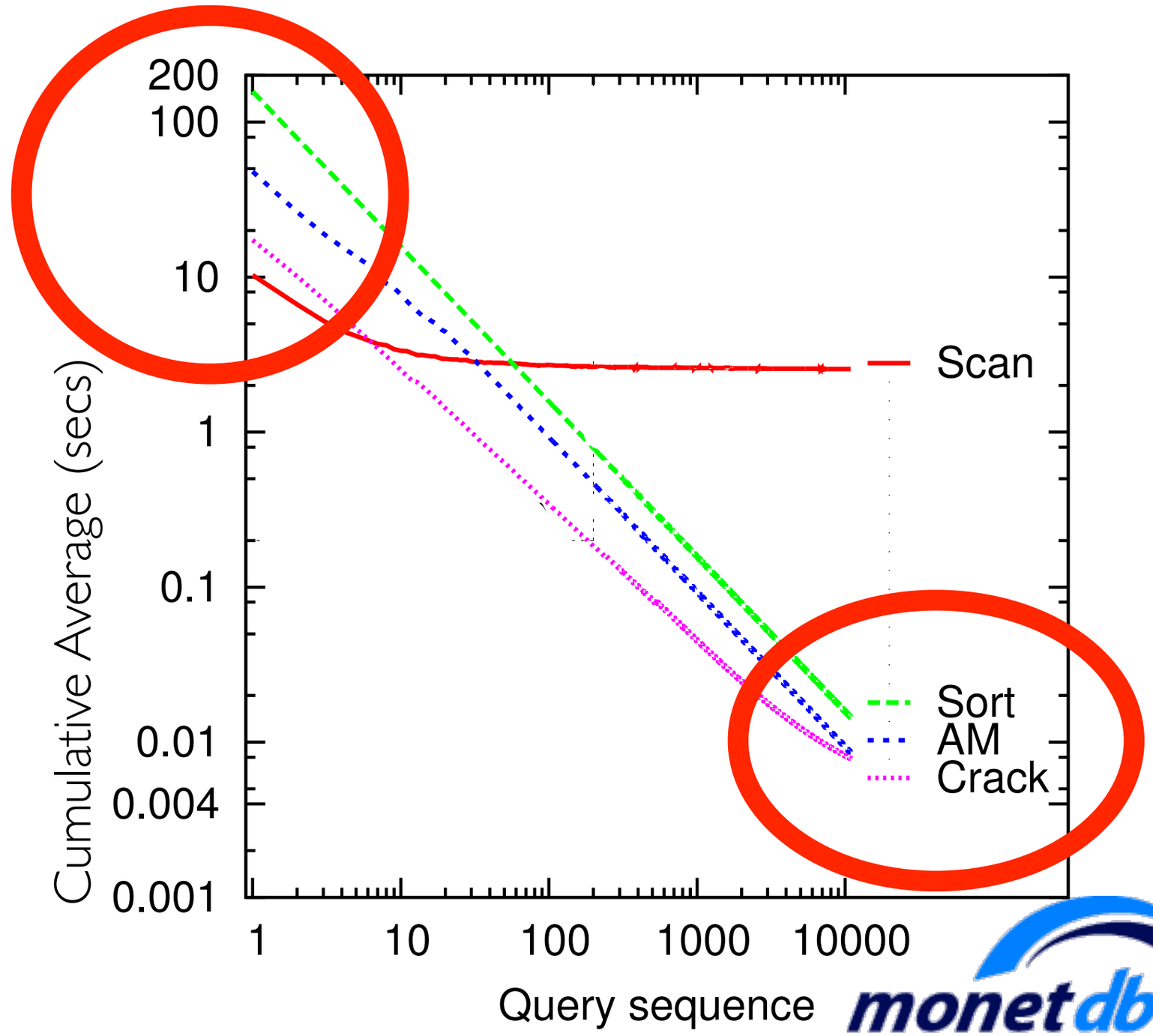
10K random selections
selectivity 10%
random value ranges
in a 30 million integer column



Performance Analysis

set-up

10K random selections
selectivity 10%
random value ranges
in a 30 million integer column

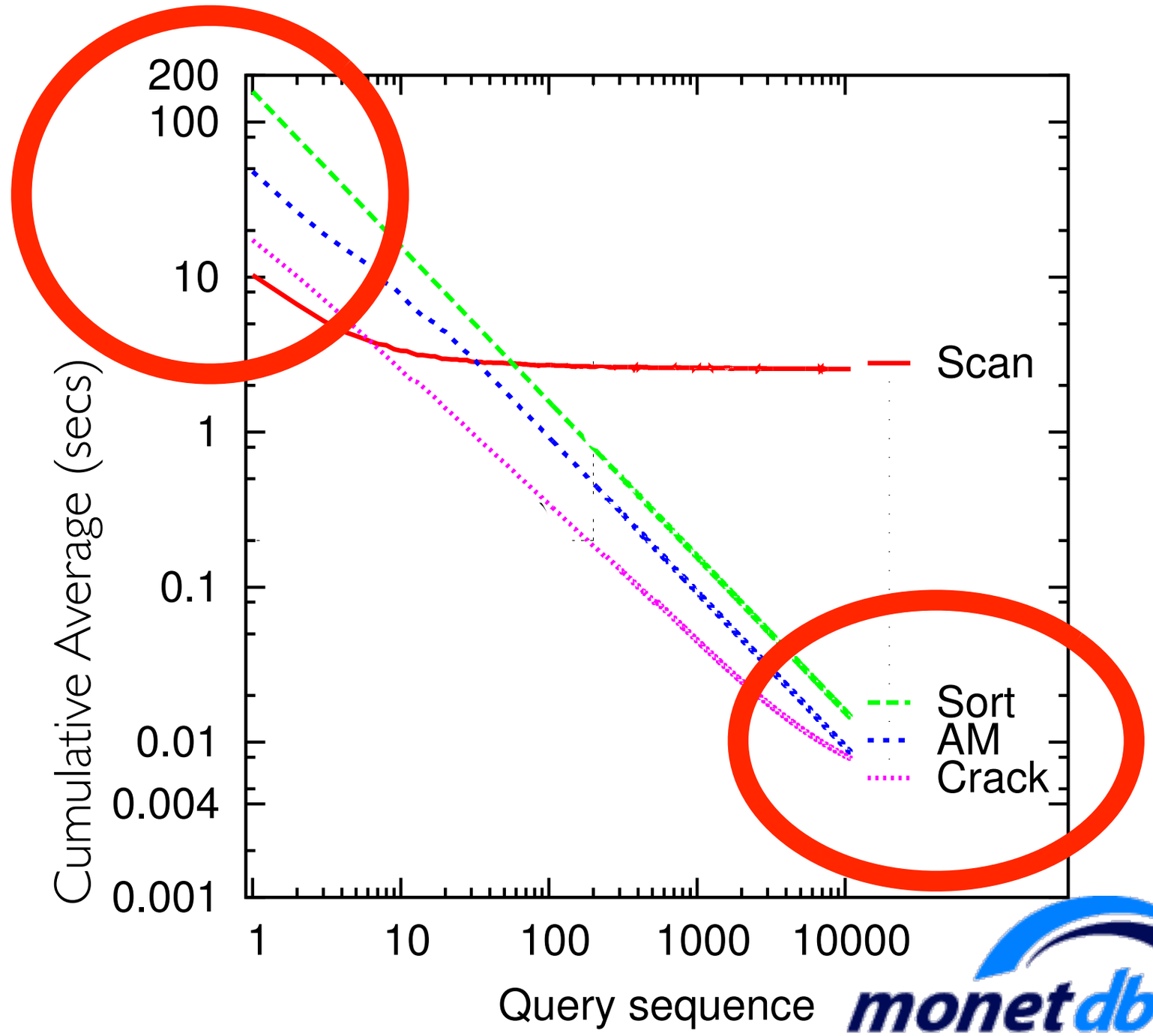


Performance Analysis

set-up

10K random selections
selectivity 10%
random value ranges
in a 30 million integer column

AM: high init overhead but fast convergence



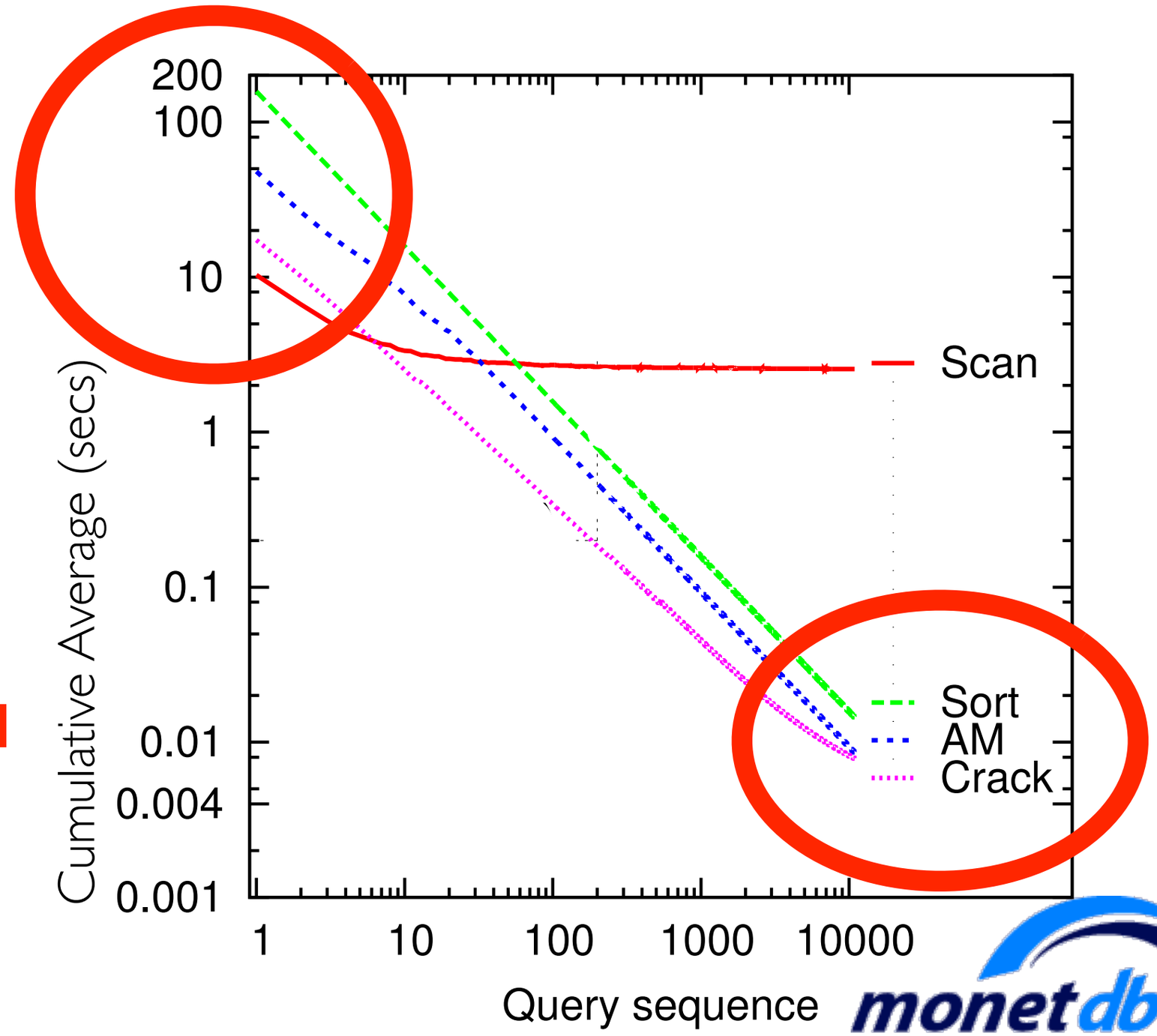
Performance Analysis

set-up

10K random selections
selectivity 10%
random value ranges
in a 30 million integer column

AM: high init overhead but fast convergence

Crack: low init overhead but slow convergence



Questions

Adaptive merging and Cracking are extremes

Questions

Adaptive merging and Cracking are extremes

What is there in between?

Crack-Crack

vary initialization and incremental steps taken

Crack-Crack

vary initialization and incremental steps taken

Crack-Crack

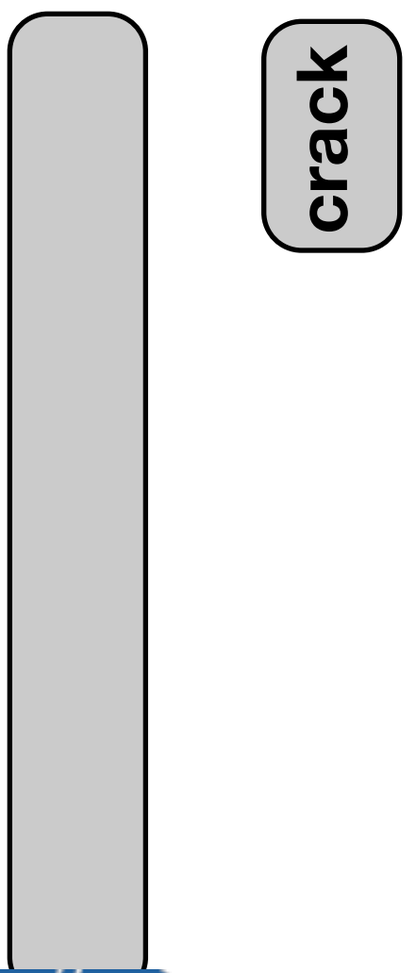
vary initialization and incremental steps taken

select(A,50,100)

Crack-Crack

vary initialization and incremental steps taken

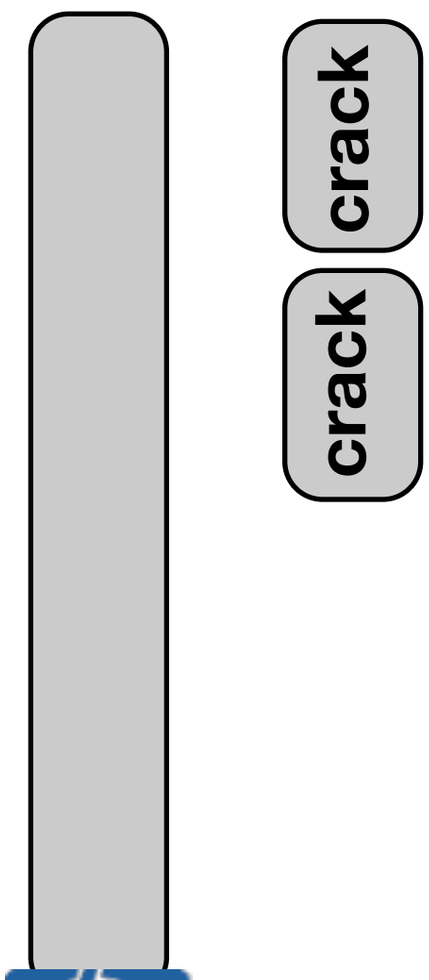
select(A,50,100)



Crack-Crack

vary initialization and incremental steps taken

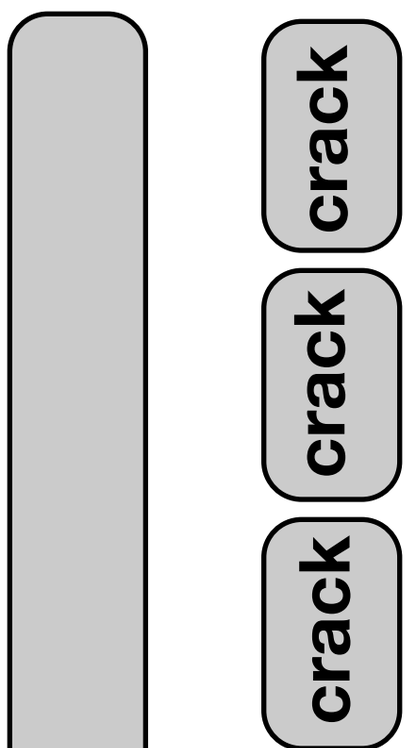
select(A,50,100)



Crack-Crack

vary initialization and incremental steps taken

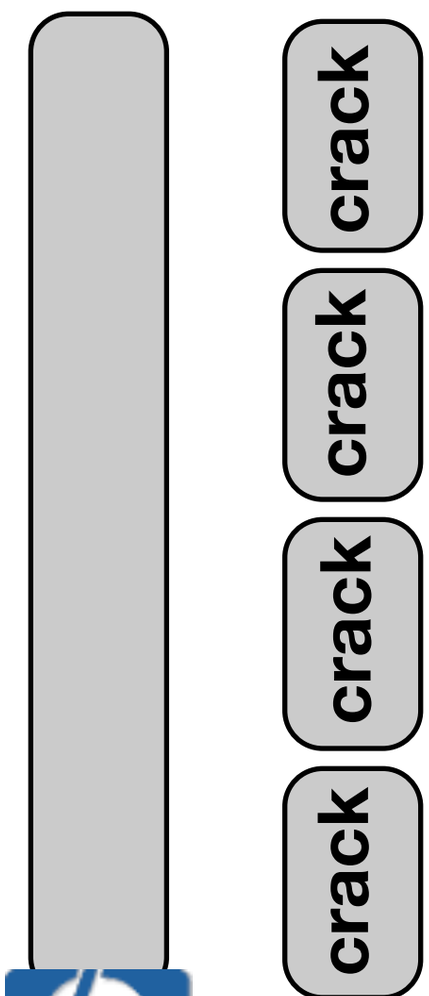
select(A,50,100)



Crack-Crack

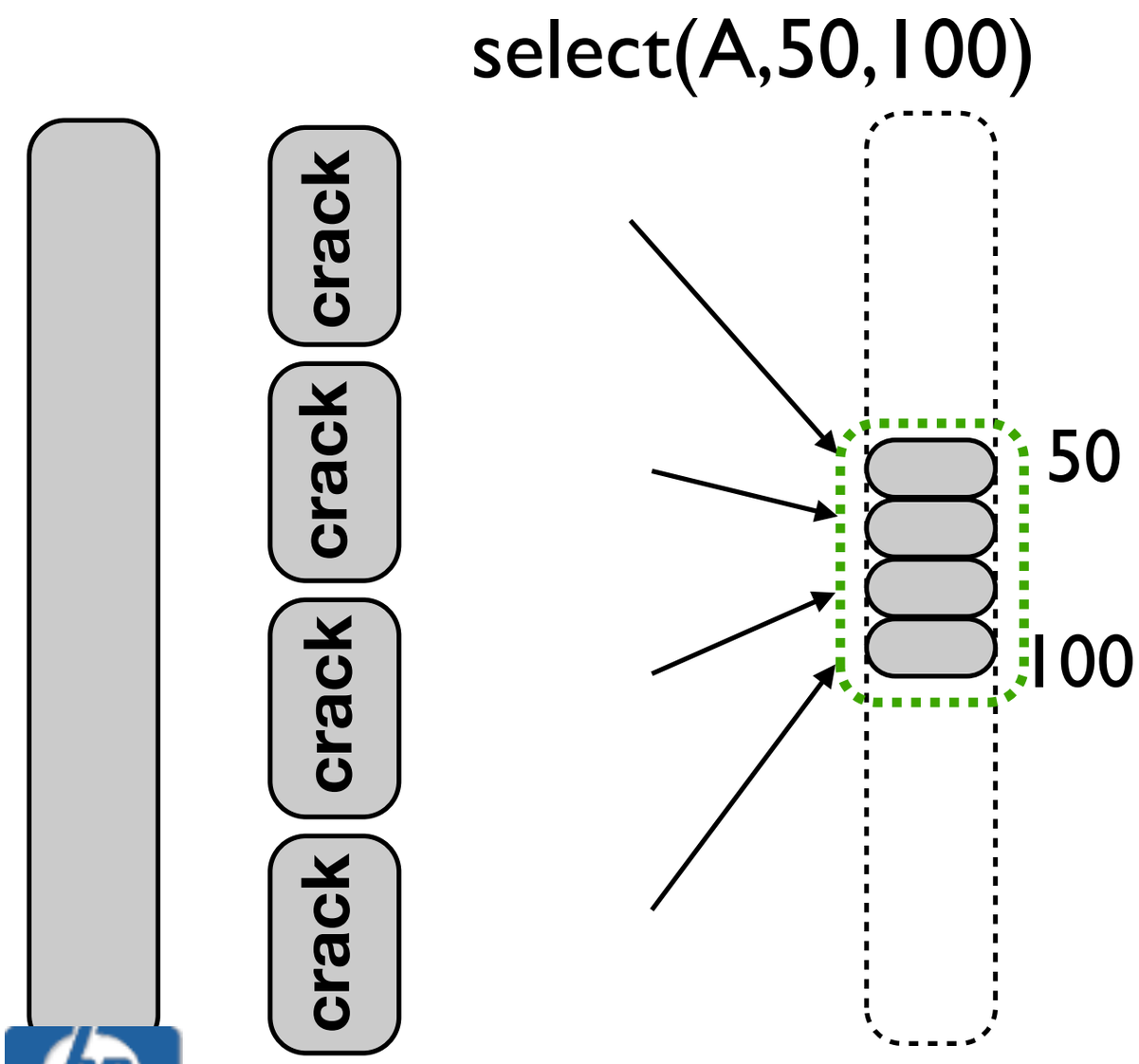
vary initialization and incremental steps taken

select(A,50,100)



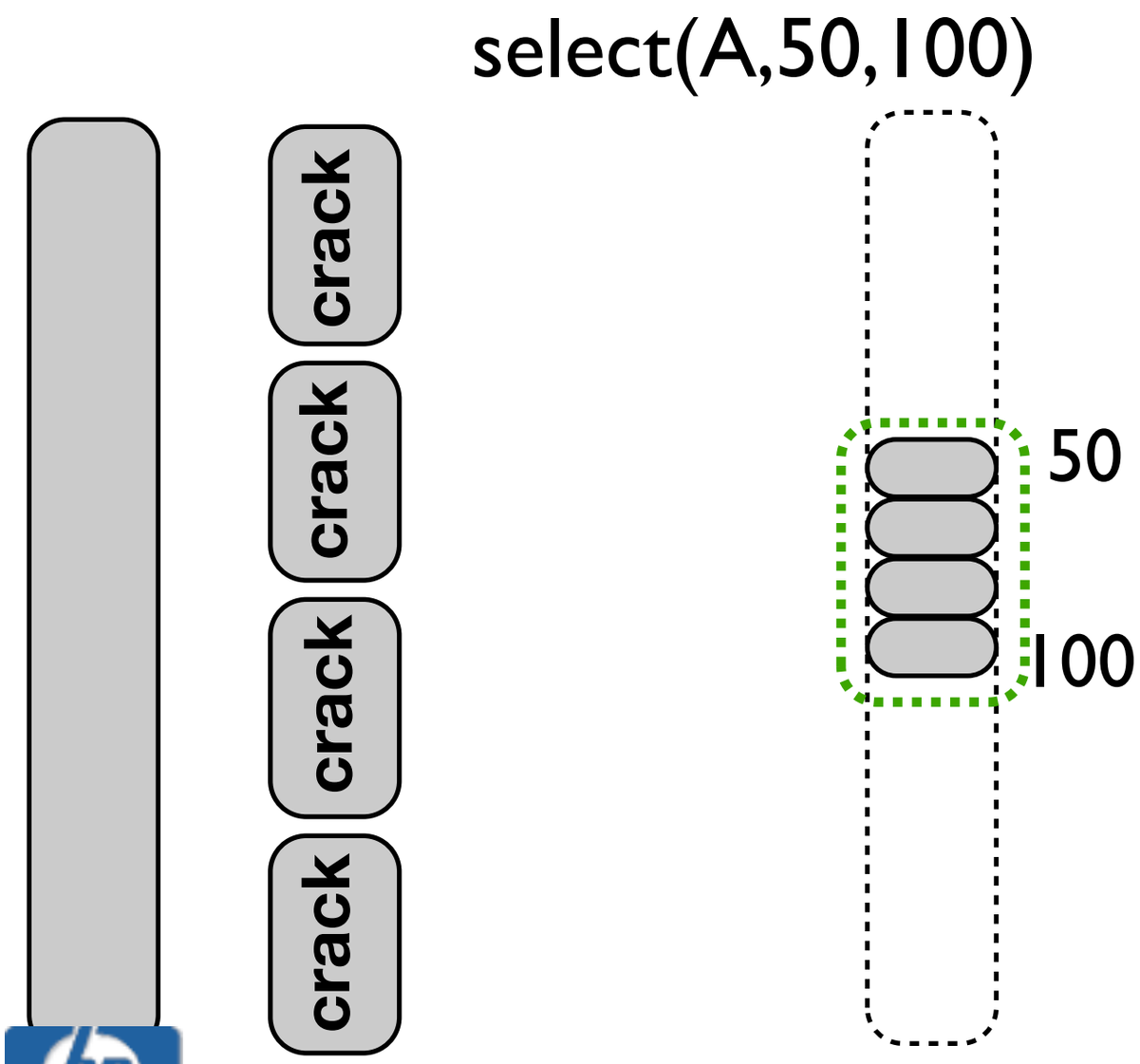
Crack-Crack

vary initialization and incremental steps taken



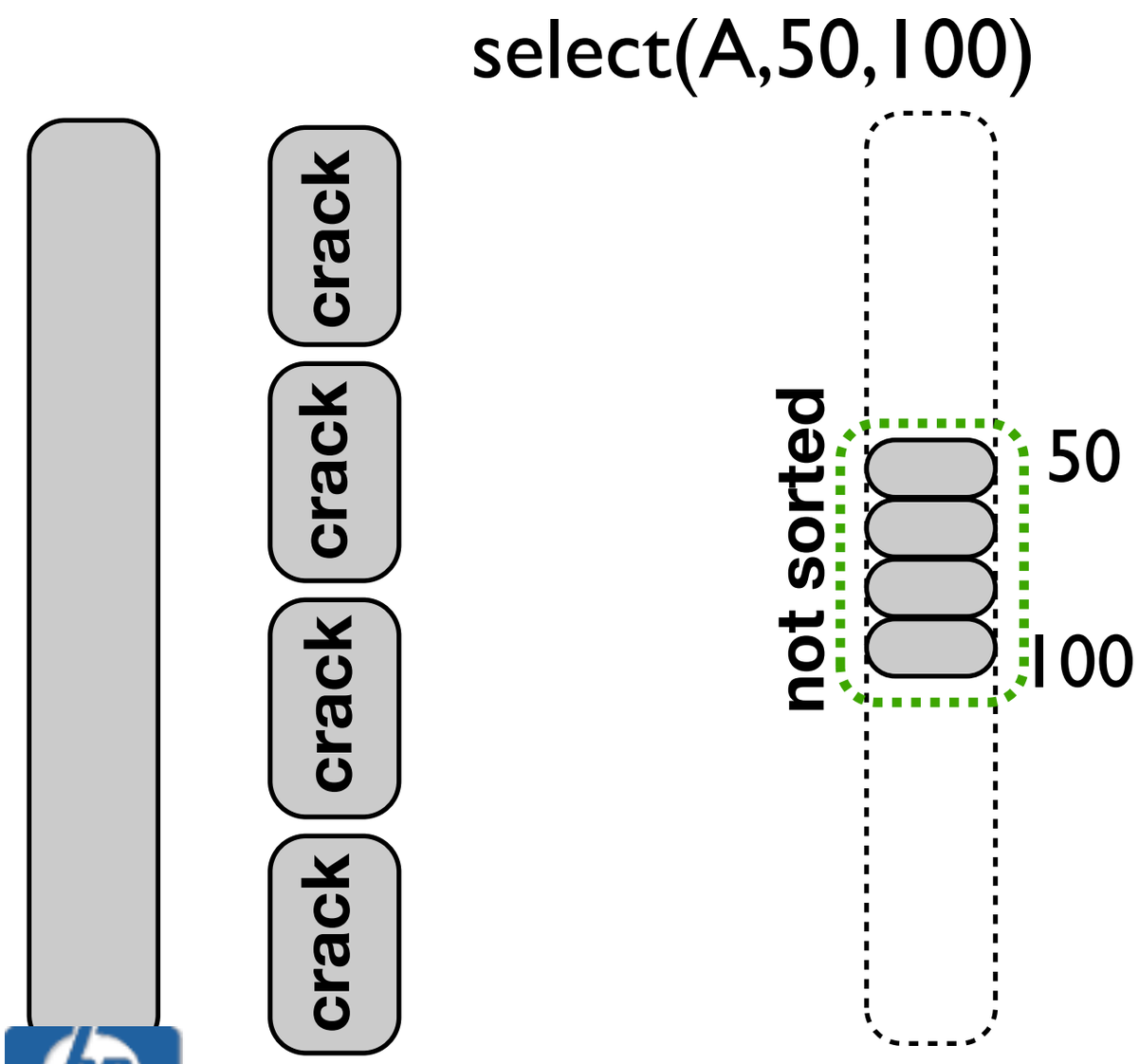
Crack-Crack

vary initialization and incremental steps taken



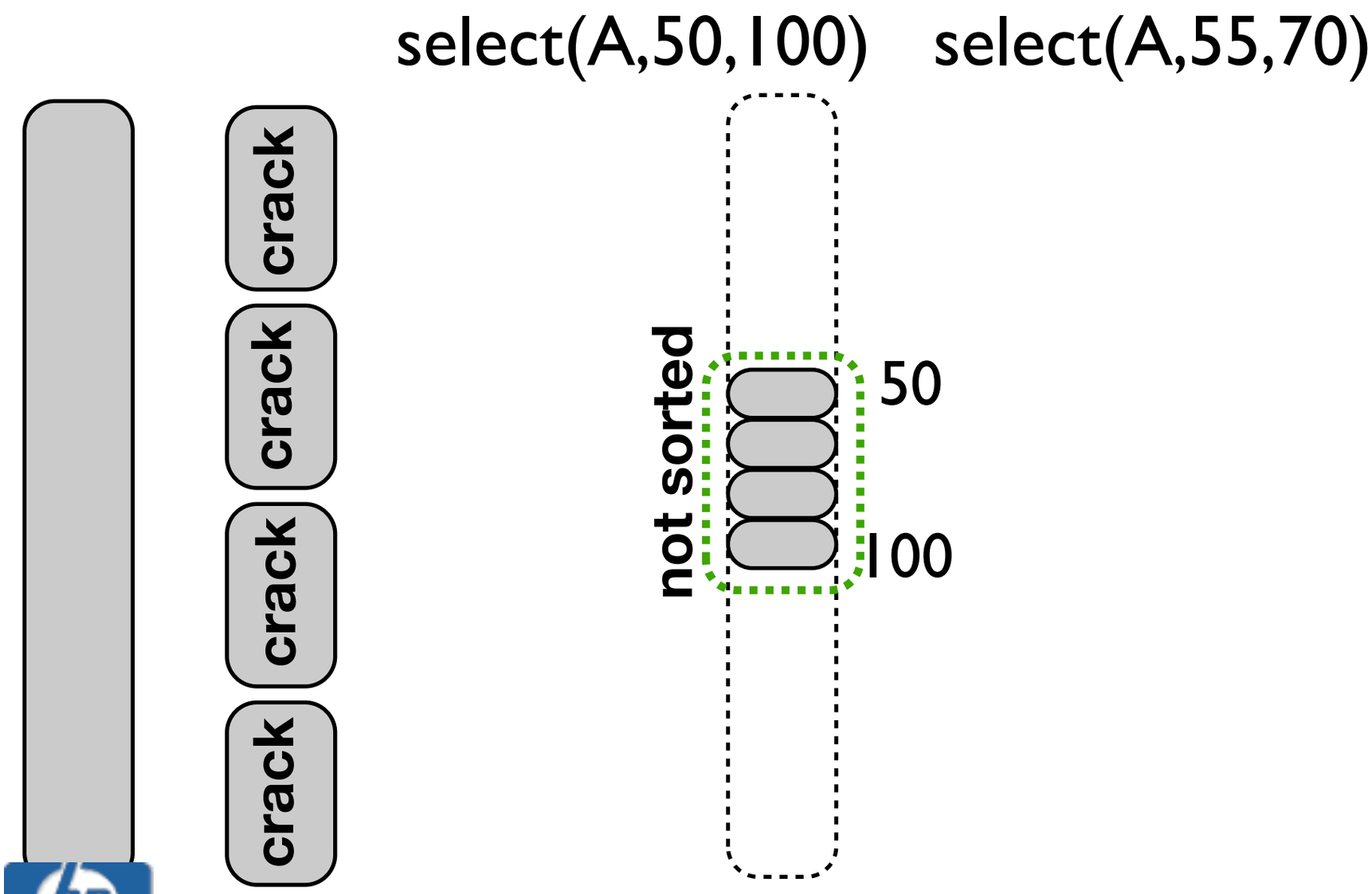
Crack-Crack

vary initialization and incremental steps taken



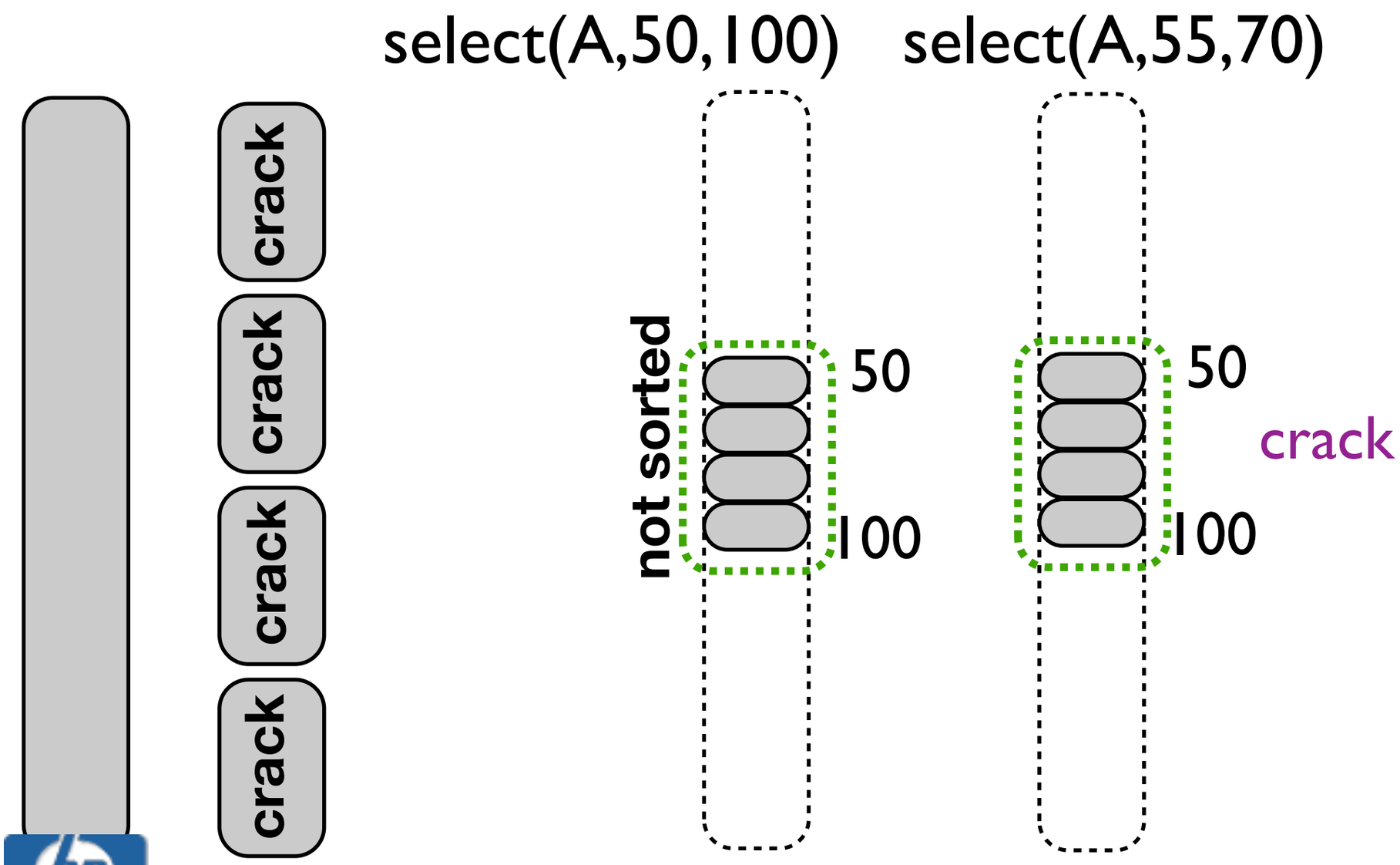
Crack-Crack

vary initialization and incremental steps taken



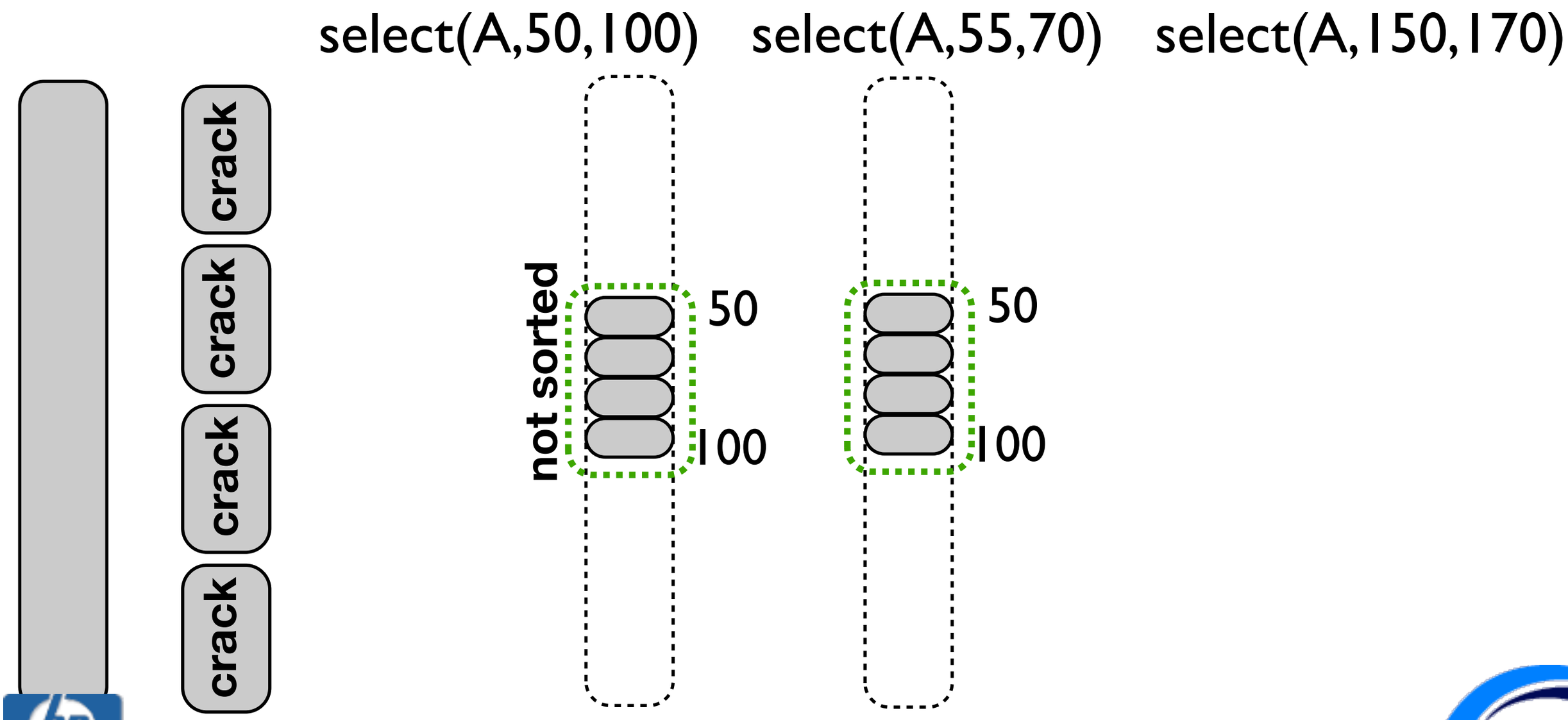
Crack-Crack

vary initialization and incremental steps taken



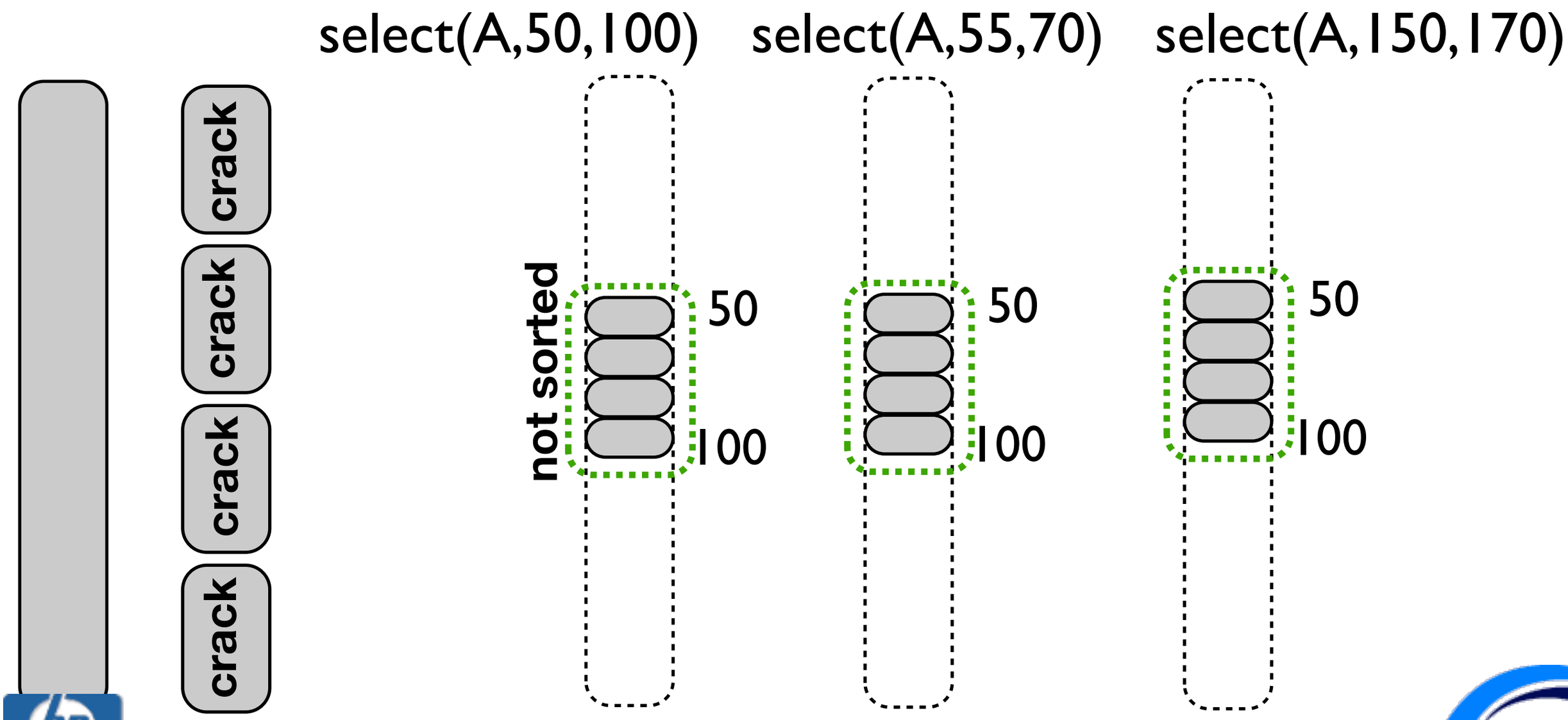
Crack-Crack

vary initialization and incremental steps taken



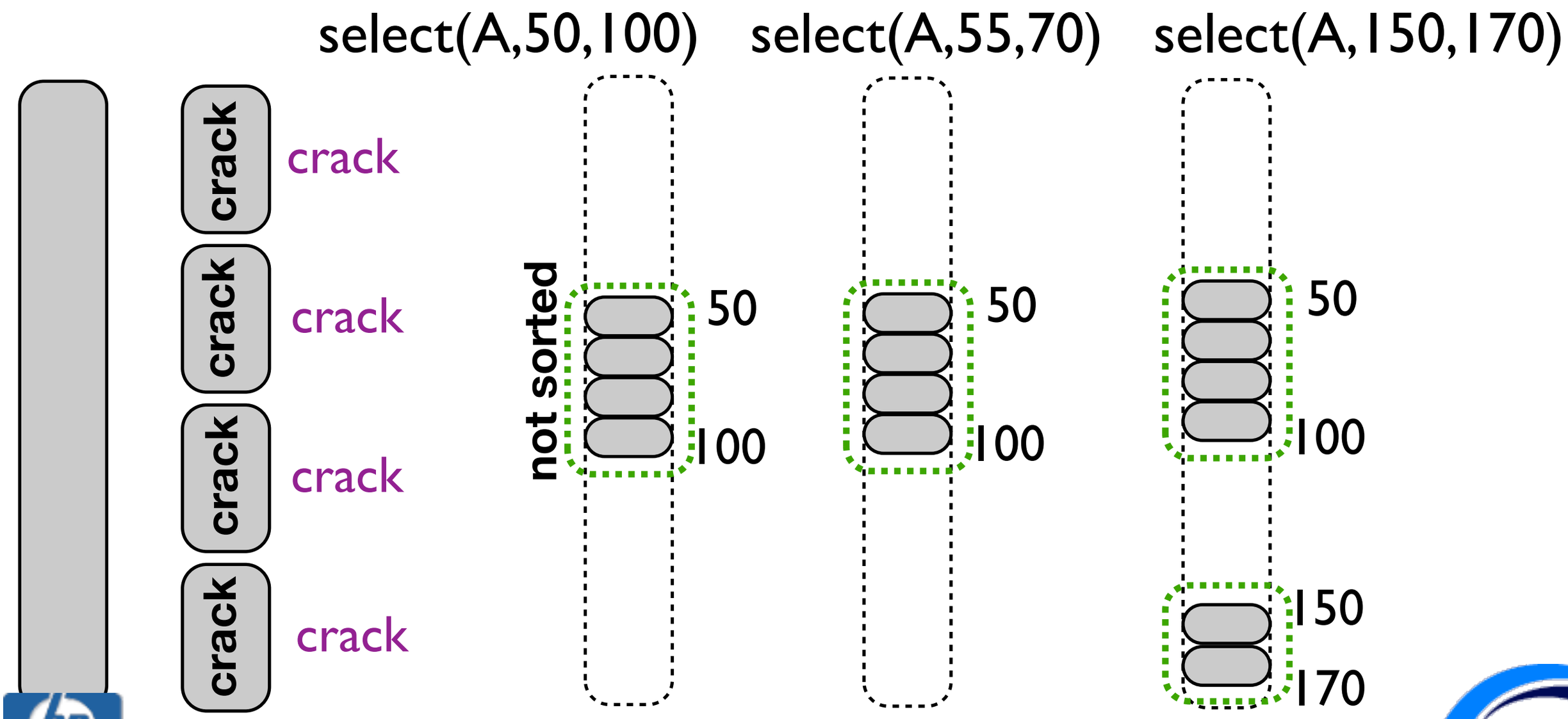
Crack-Crack

vary initialization and incremental steps taken



Crack-Crack

vary initialization and incremental steps taken



Adaptive Indexing

high – overhead – low

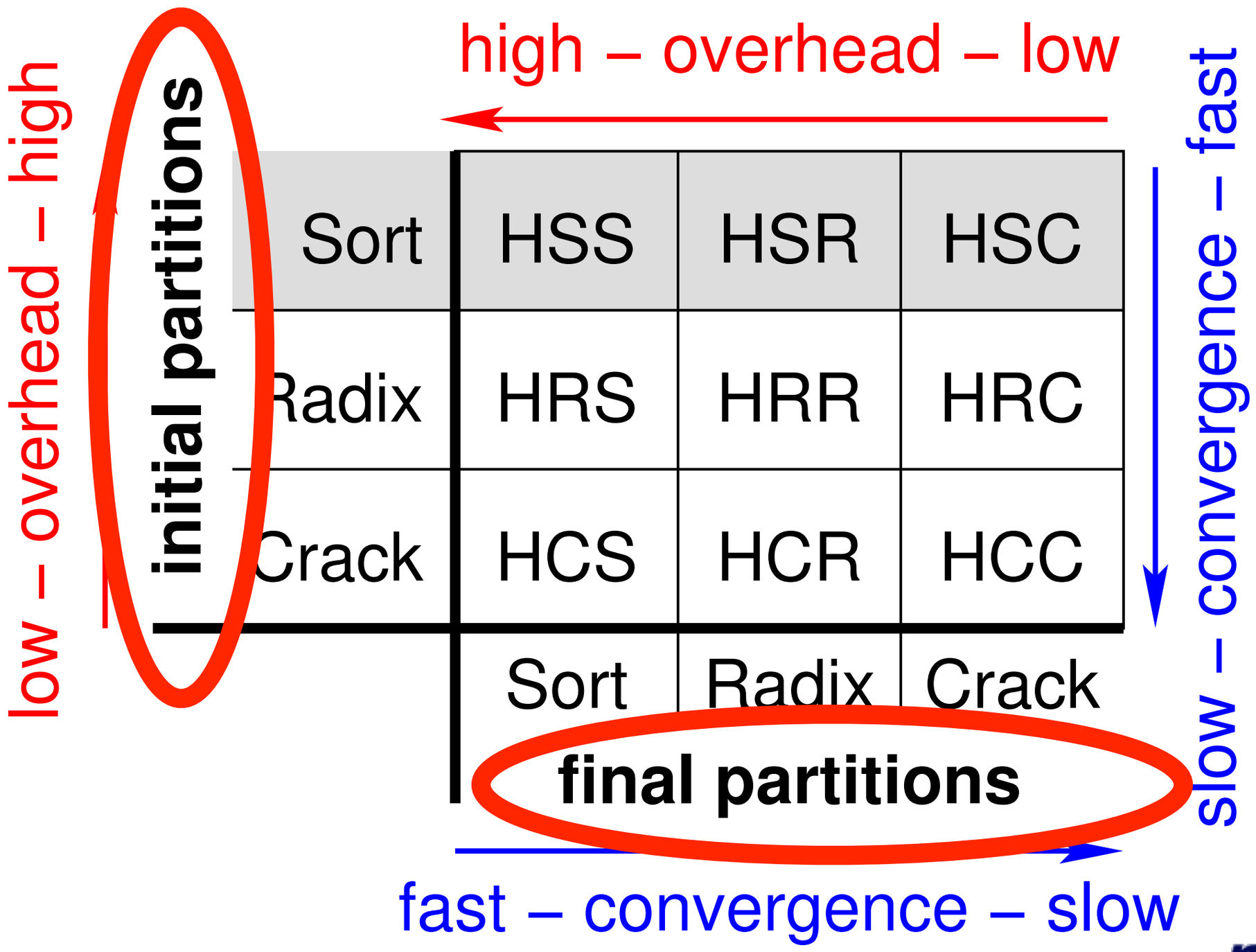
low – overhead – high

initial partitions	Sort	HSS	HSR	HSC
	Radix	HRS	HRR	HRC
	Crack	HCS	HCR	HCC
		Sort	Radix	Crack
		final partitions		

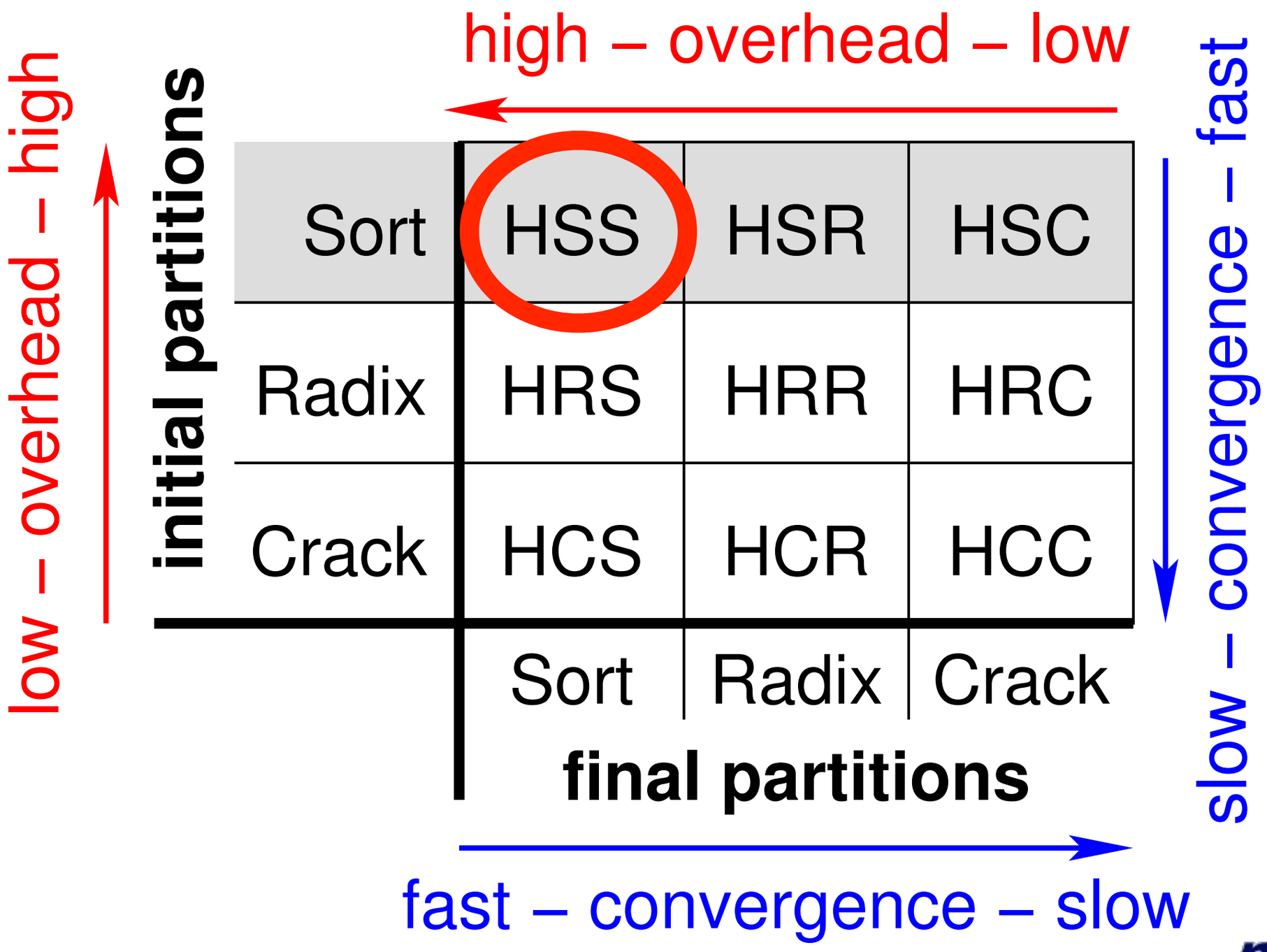
slow – convergence – fast

fast – convergence – slow

Adaptive Indexing



Adaptive Indexing



Adaptive Indexing

high – overhead – low

low – overhead – high

initial partitions \ final partitions	Sort	Radix	Crack
Sort	HSS	HSR	HSC
Radix	HRS	HRR	HRC
Crack	HCS	HCR	HCC

slow – convergence – fast

fast – convergence – slow

Adaptive Indexing

high – overhead – low

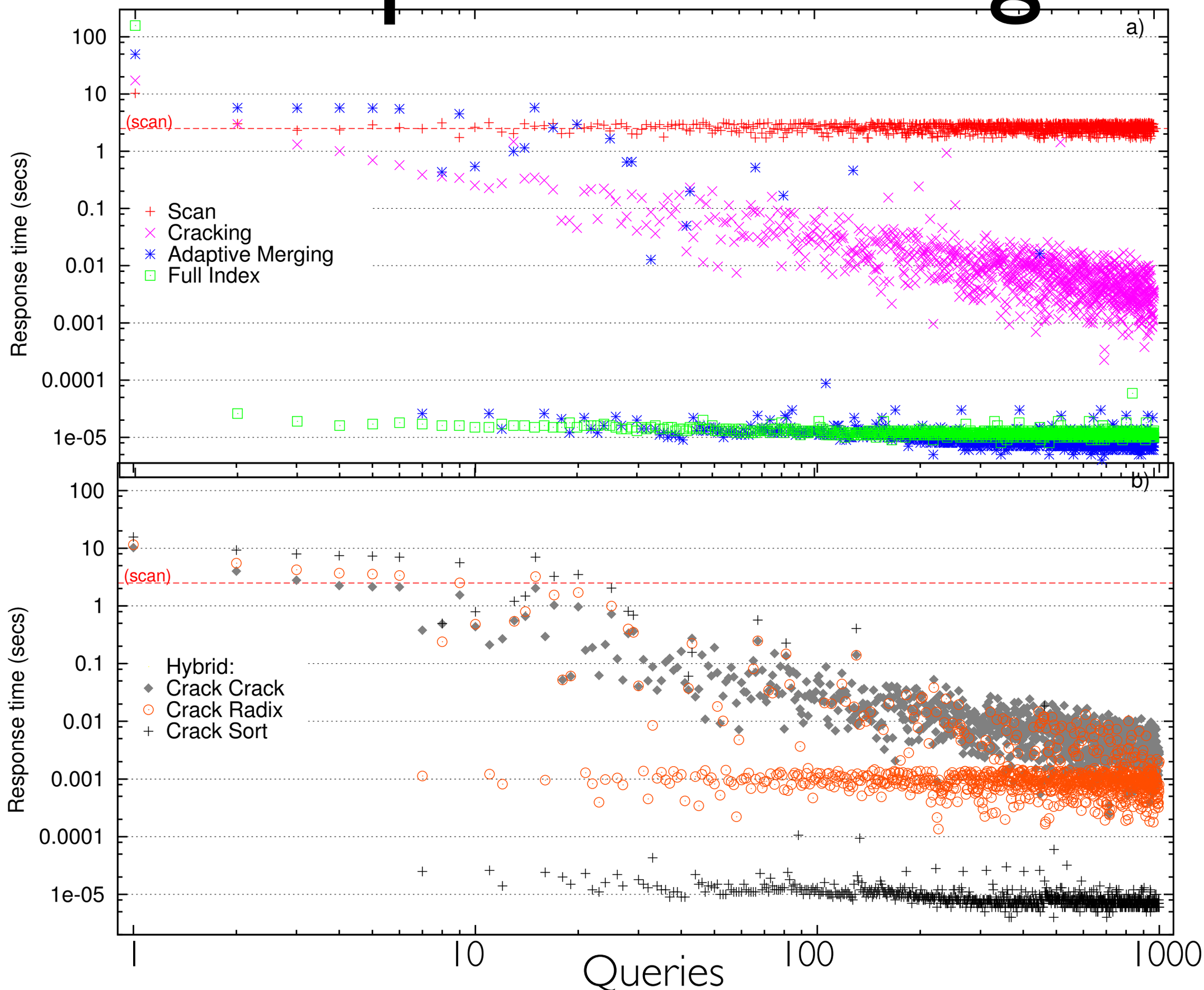
low – overhead – high

initial partitions \ final partitions	Sort	Radix	Crack
Sort	HSS	HSR	HSC
Radix	HRS	HRR	HRC
Crack	HCS	HCR	HCC

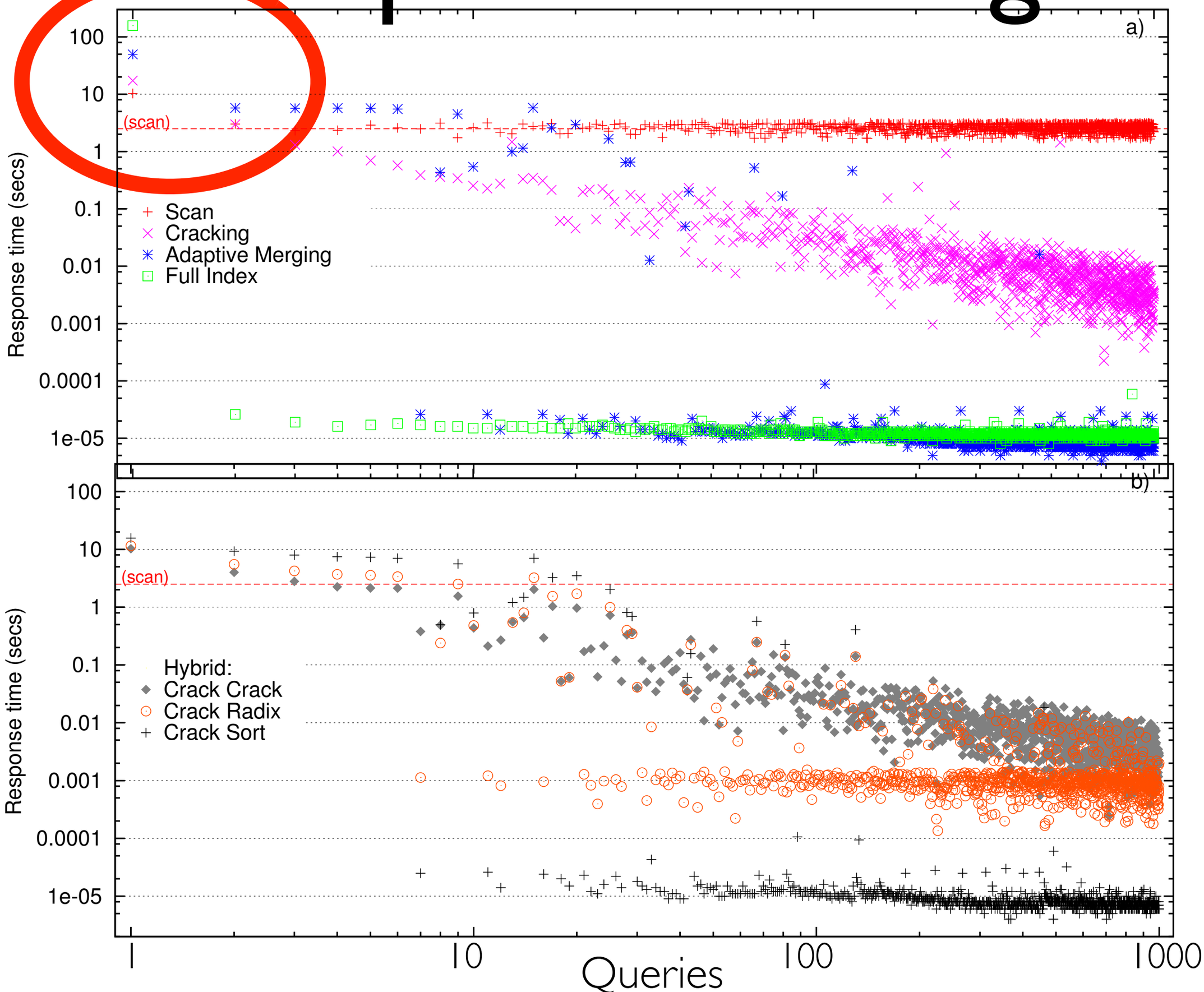
slow – convergence – fast

fast – convergence – slow

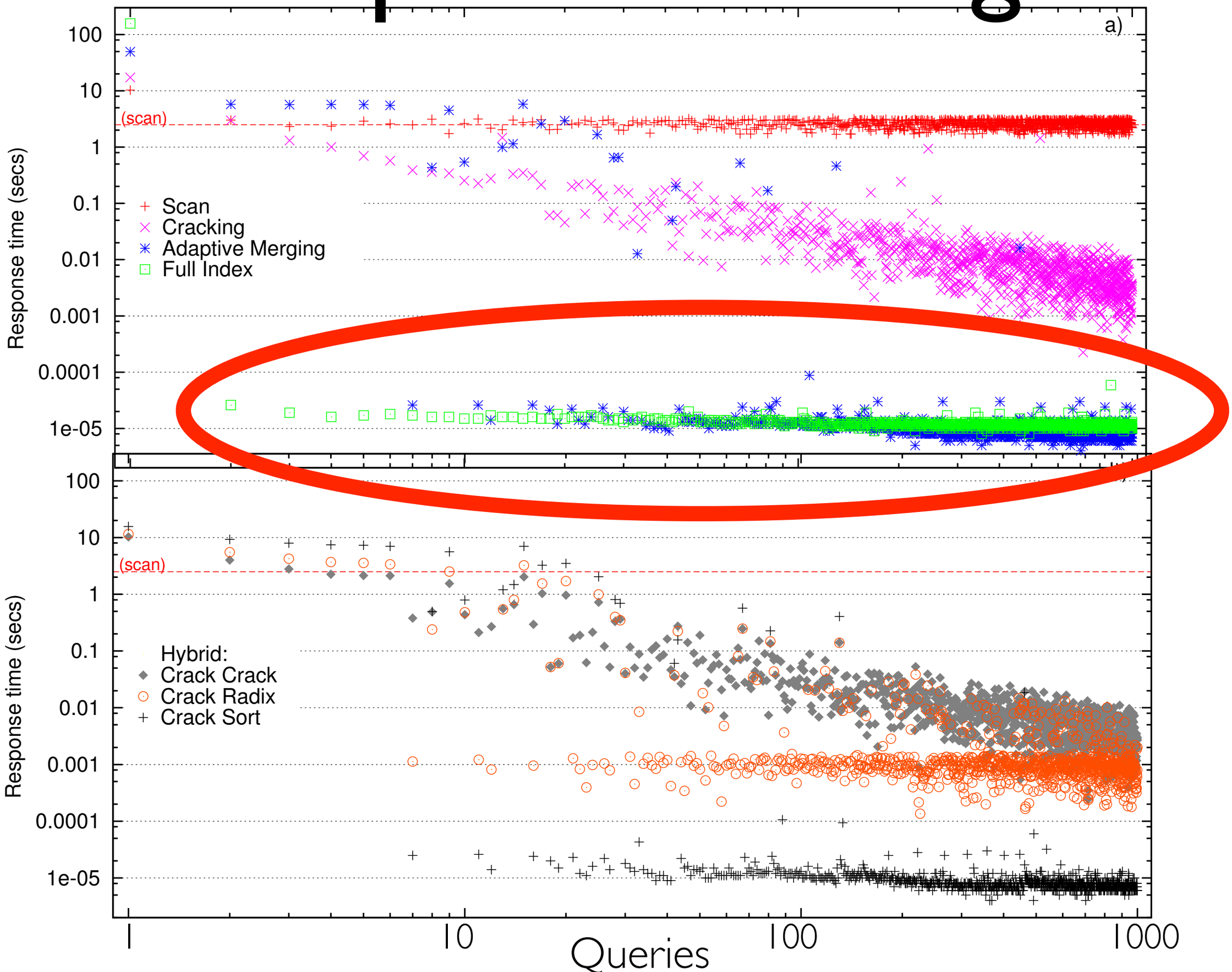
Adaptive Indexing



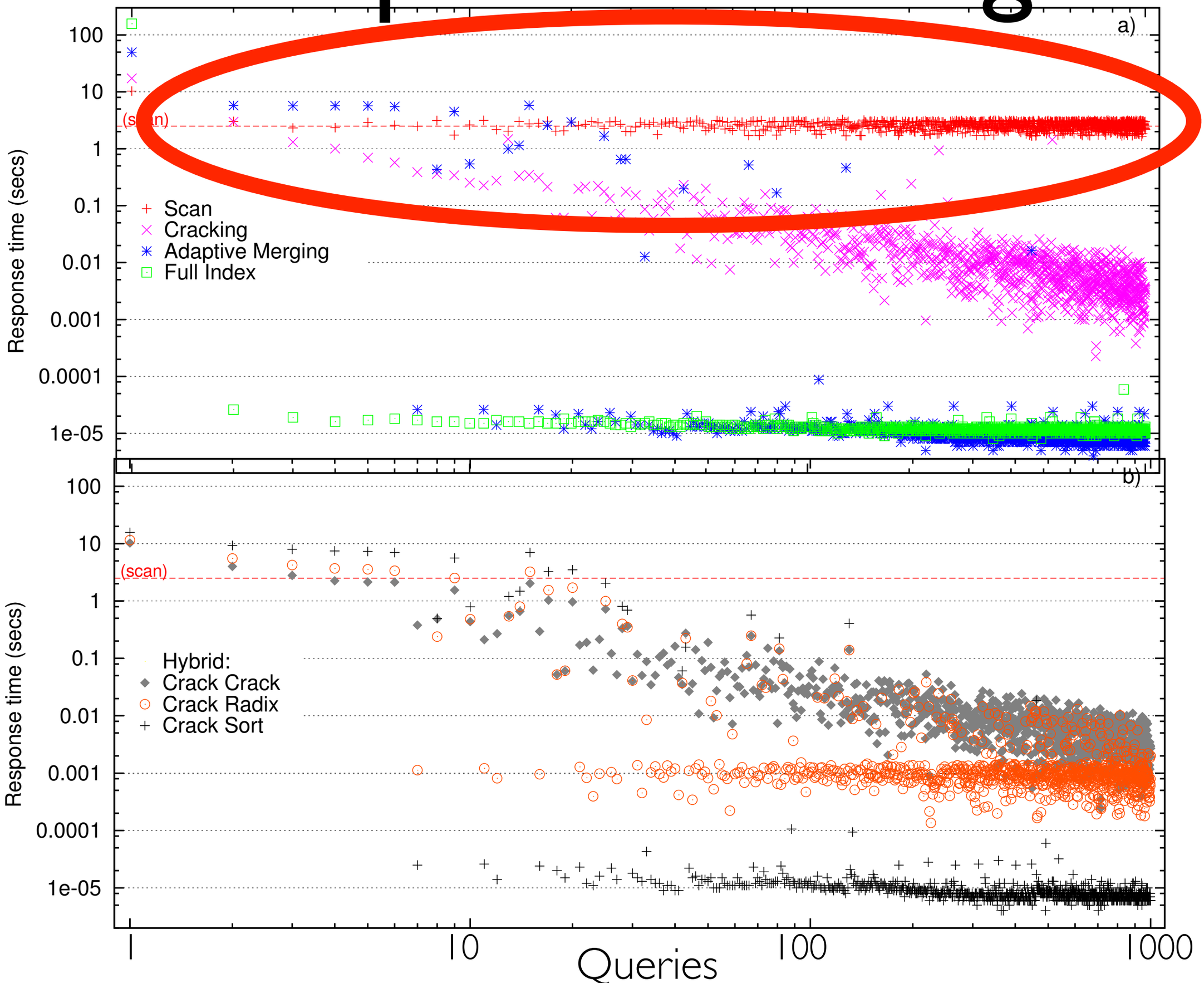
Adaptive Indexing



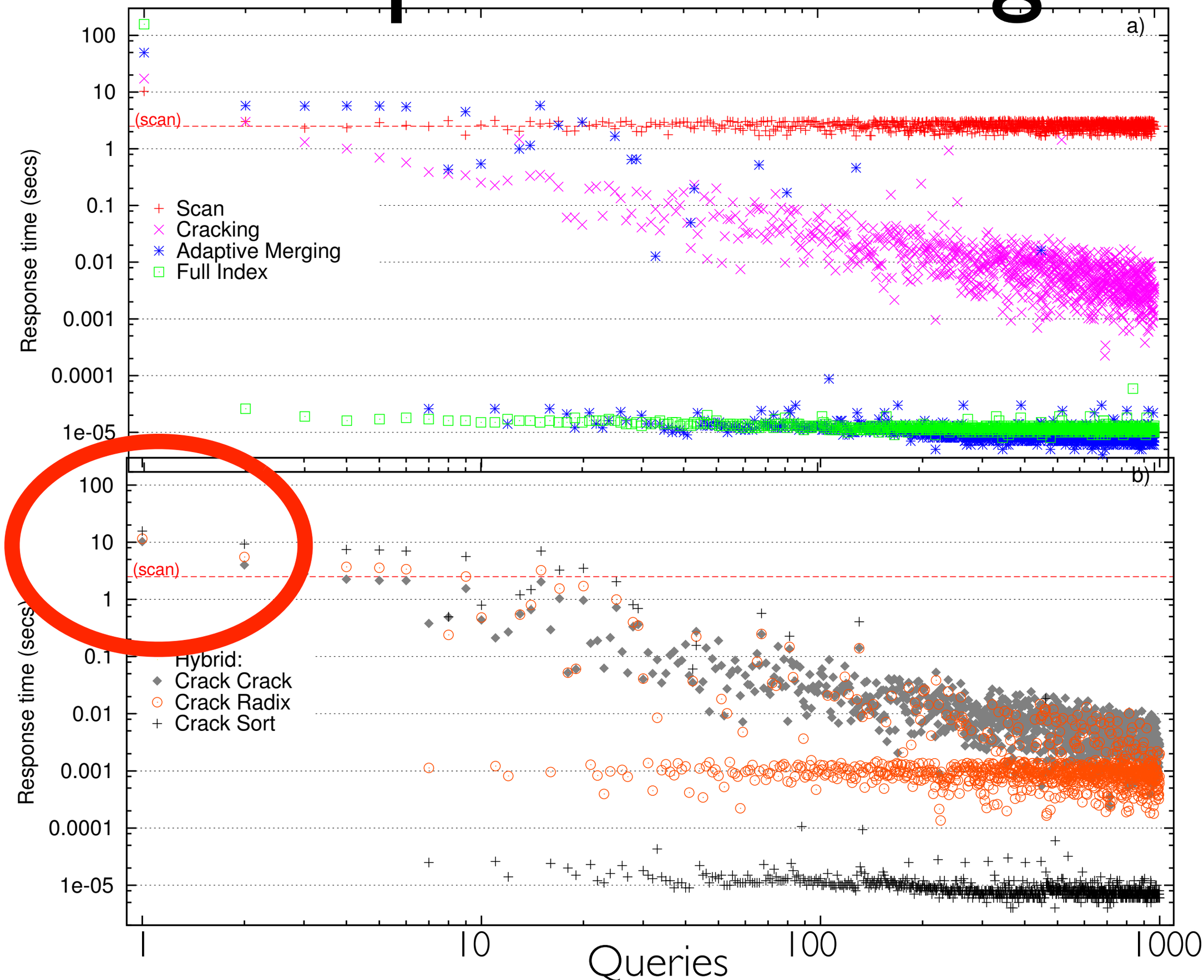
Adaptive Indexing



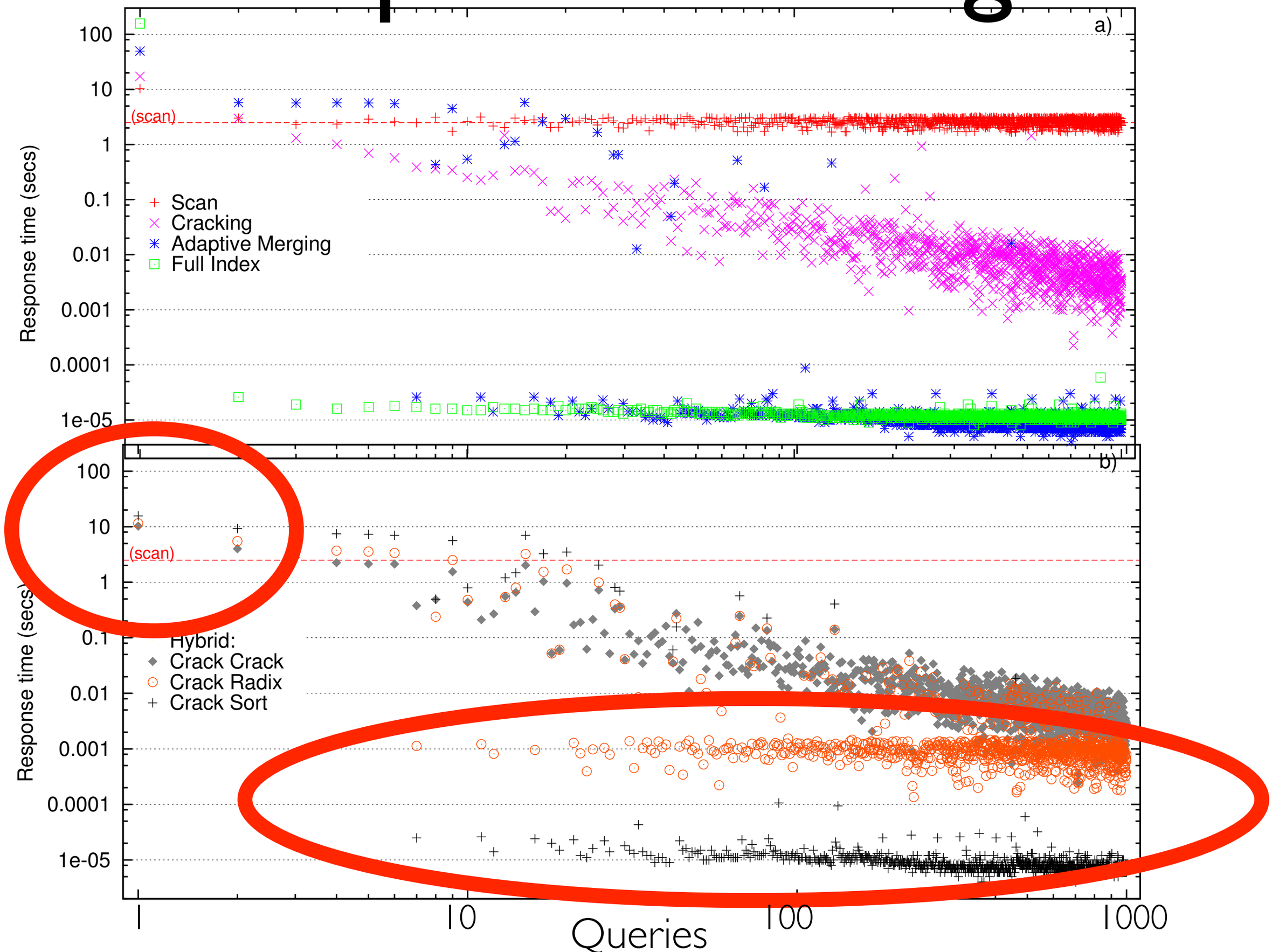
Adaptive Indexing



Adaptive Indexing

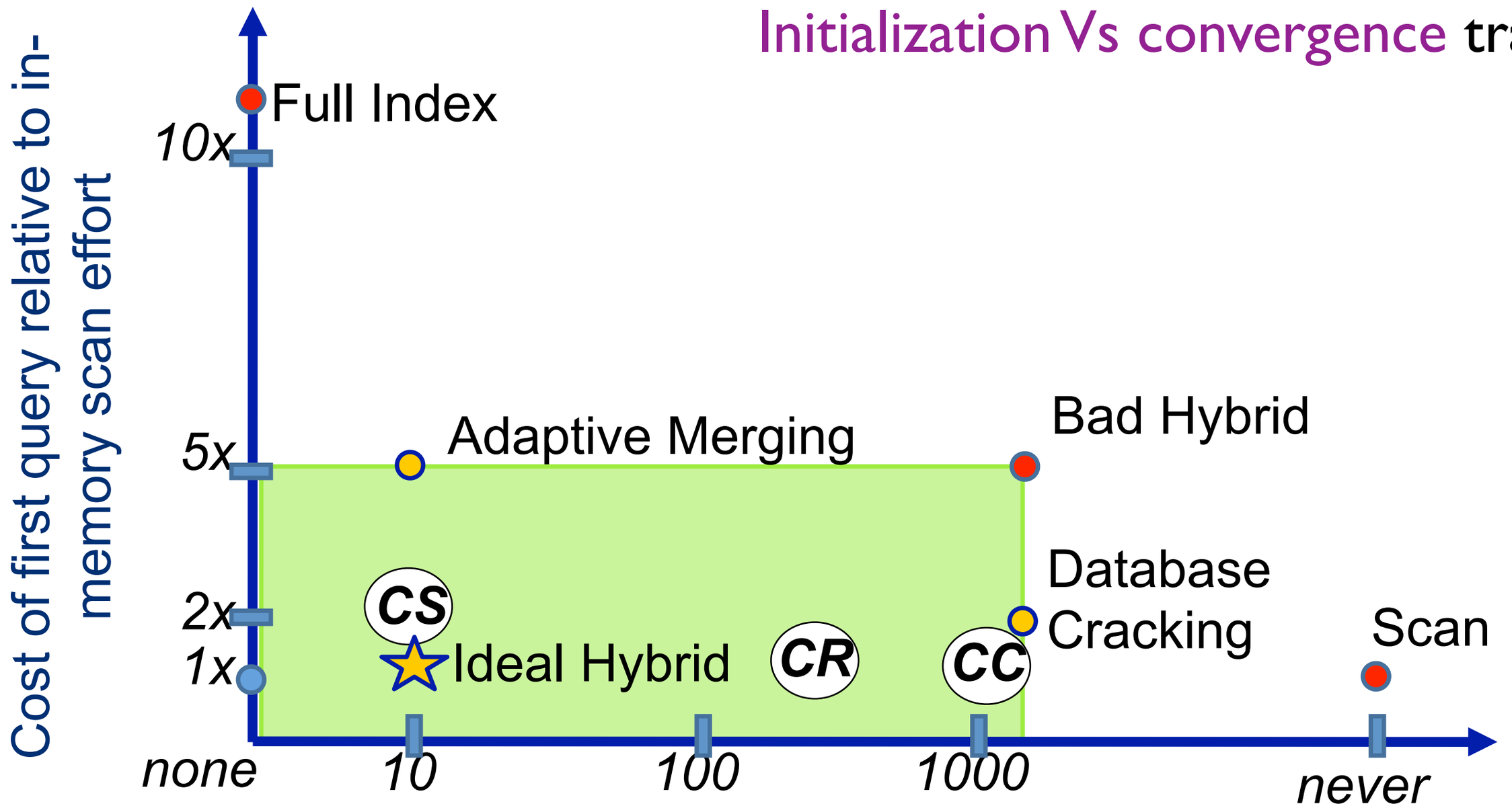


Adaptive Indexing



Adaptive Indexing

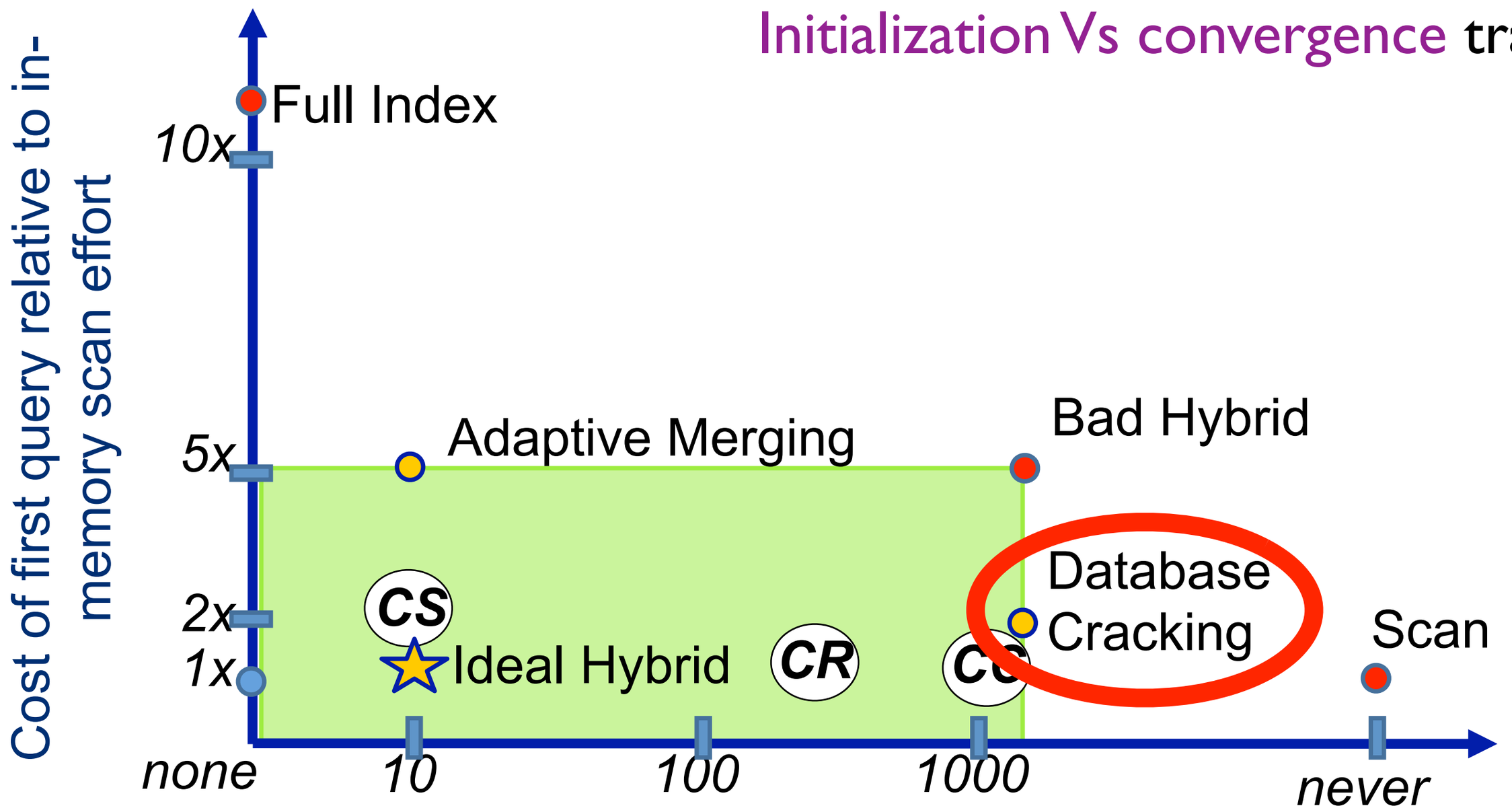
Initialization Vs convergence tradeoff



How many queries before the index fully supports a random query?

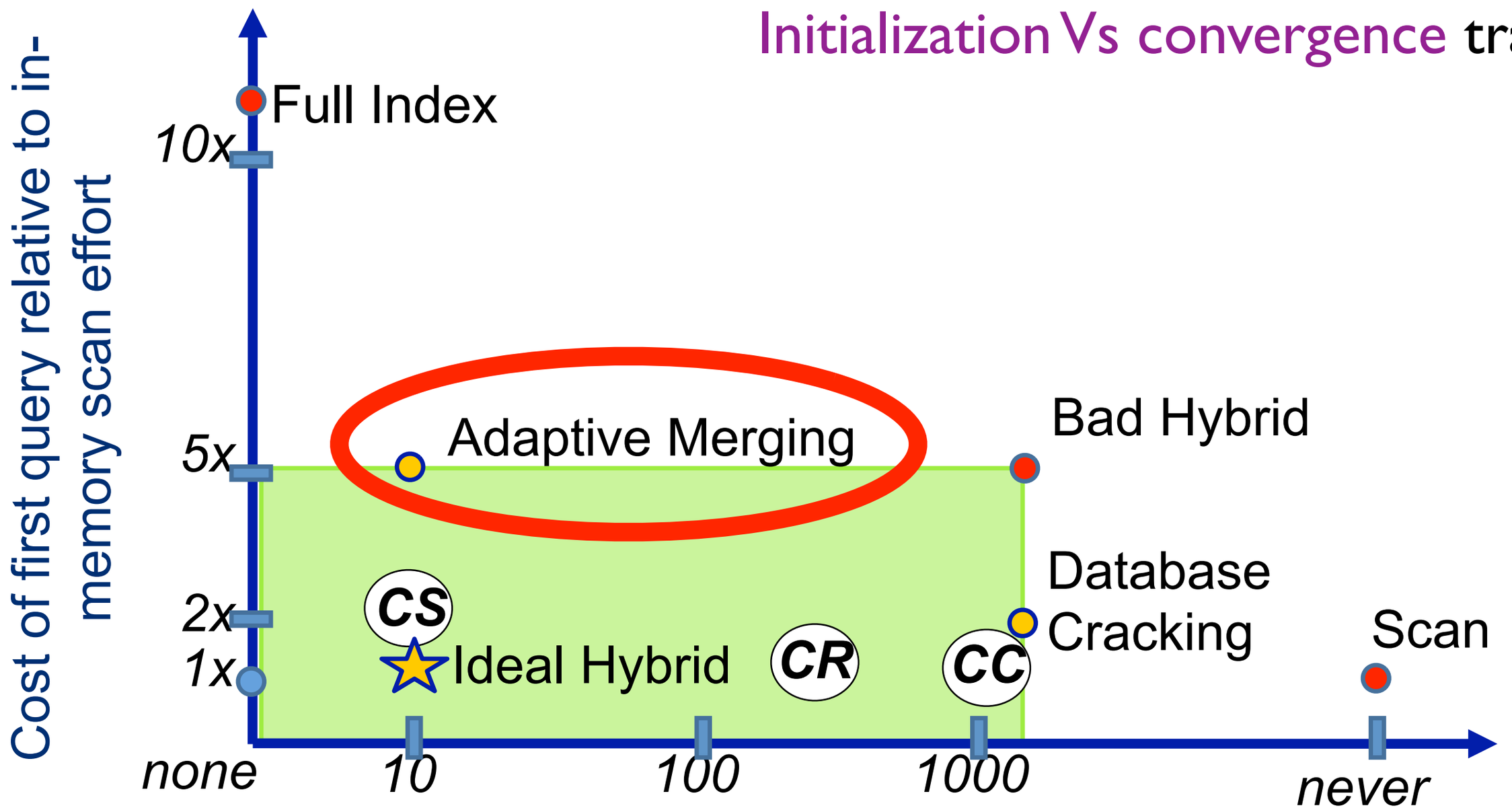
Adaptive Indexing

Initialization Vs convergence tradeoff



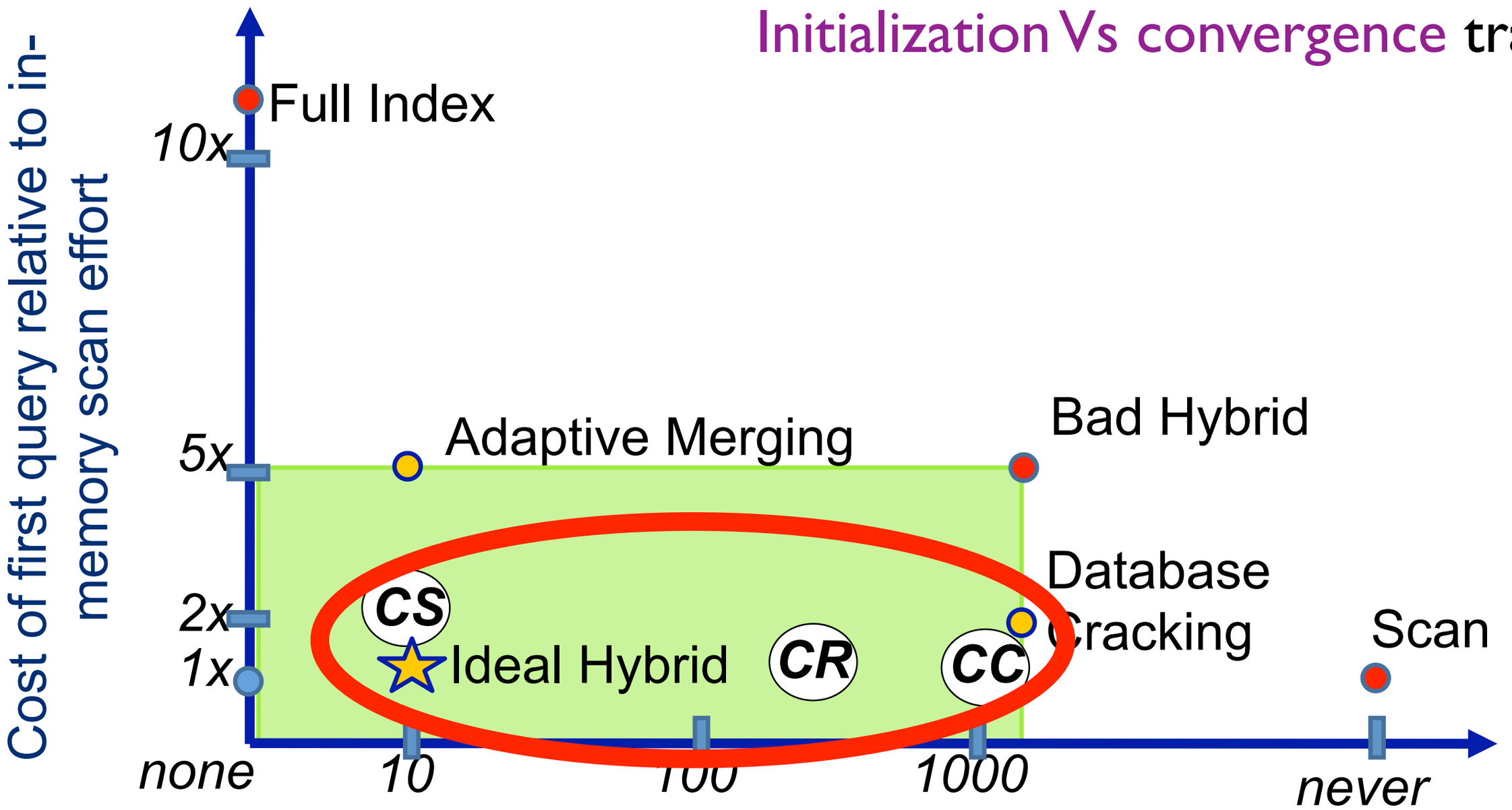
Adaptive Indexing

Initialization Vs convergence tradeoff



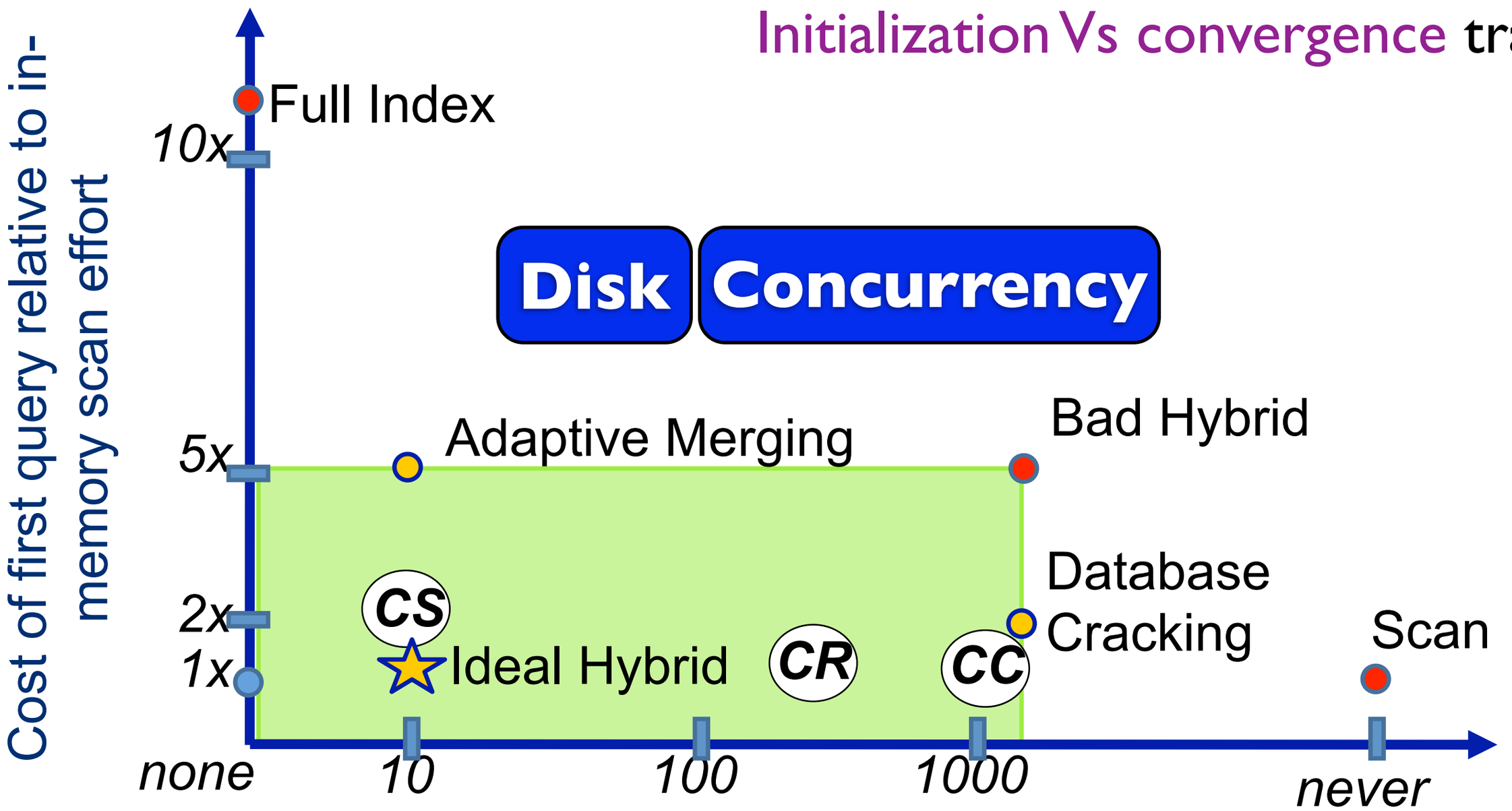
Adaptive Indexing

Initialization Vs convergence tradeoff



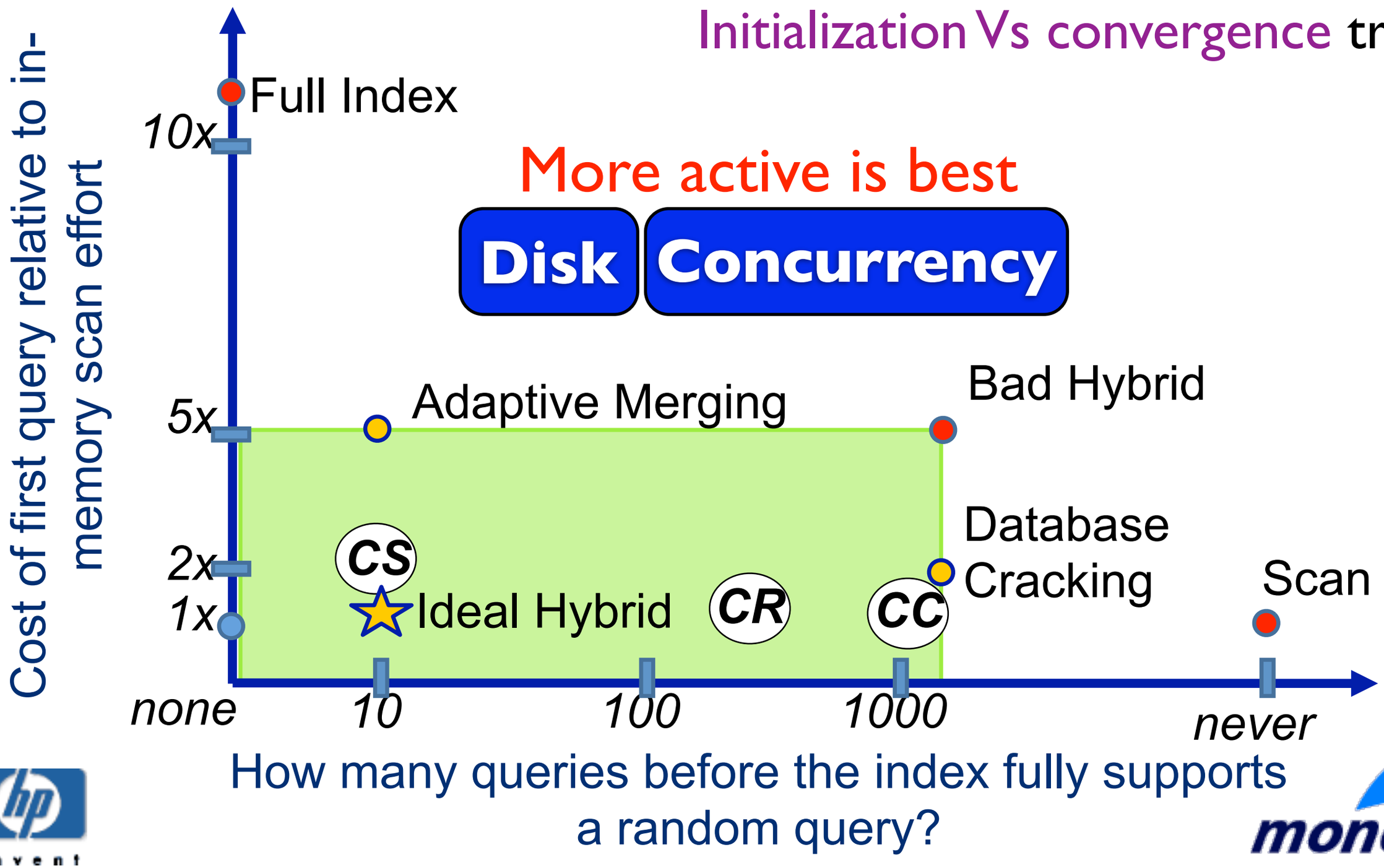
Adaptive Indexing

Initialization Vs convergence tradeoff



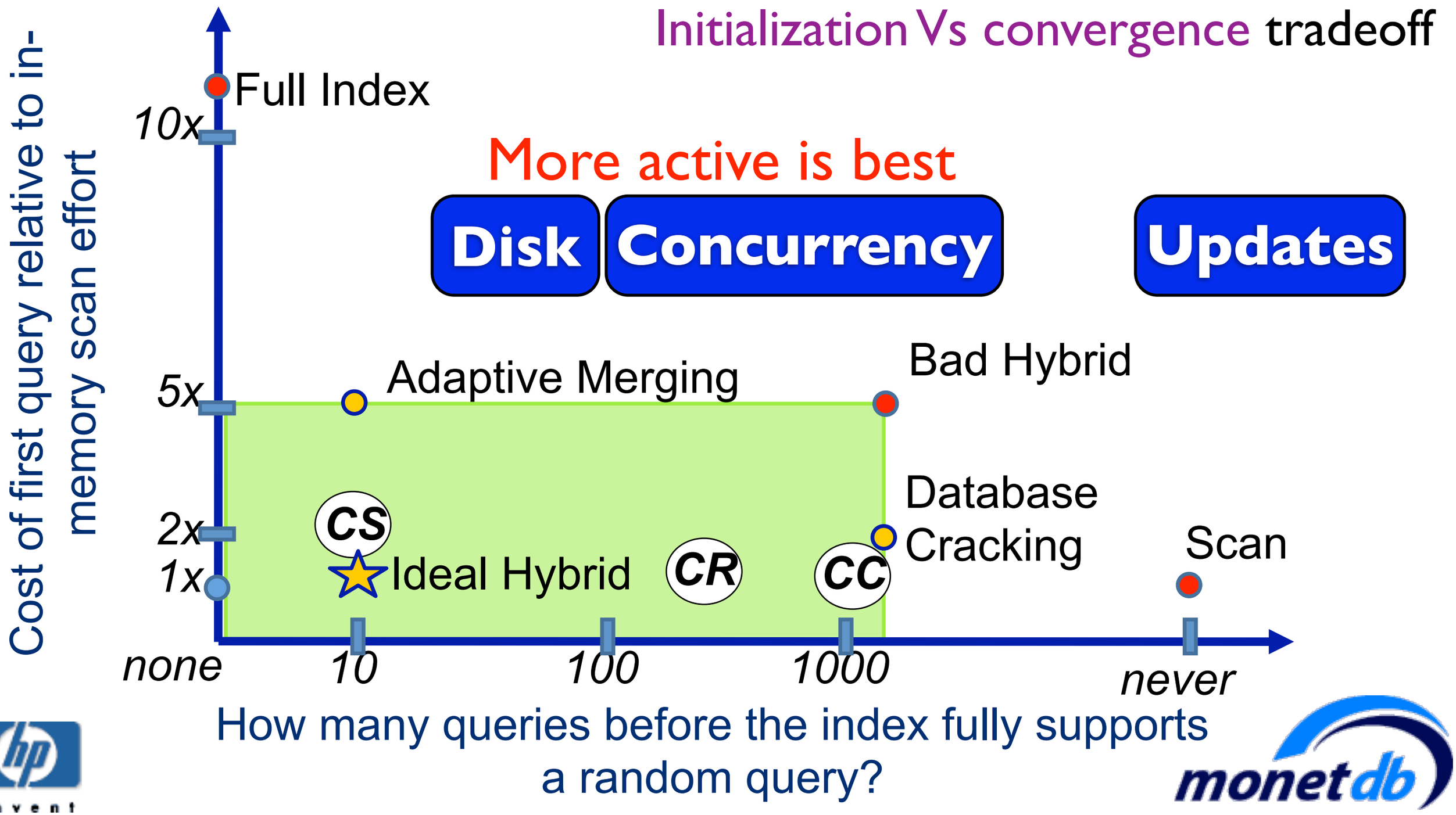
Adaptive Indexing

Initialization Vs convergence tradeoff

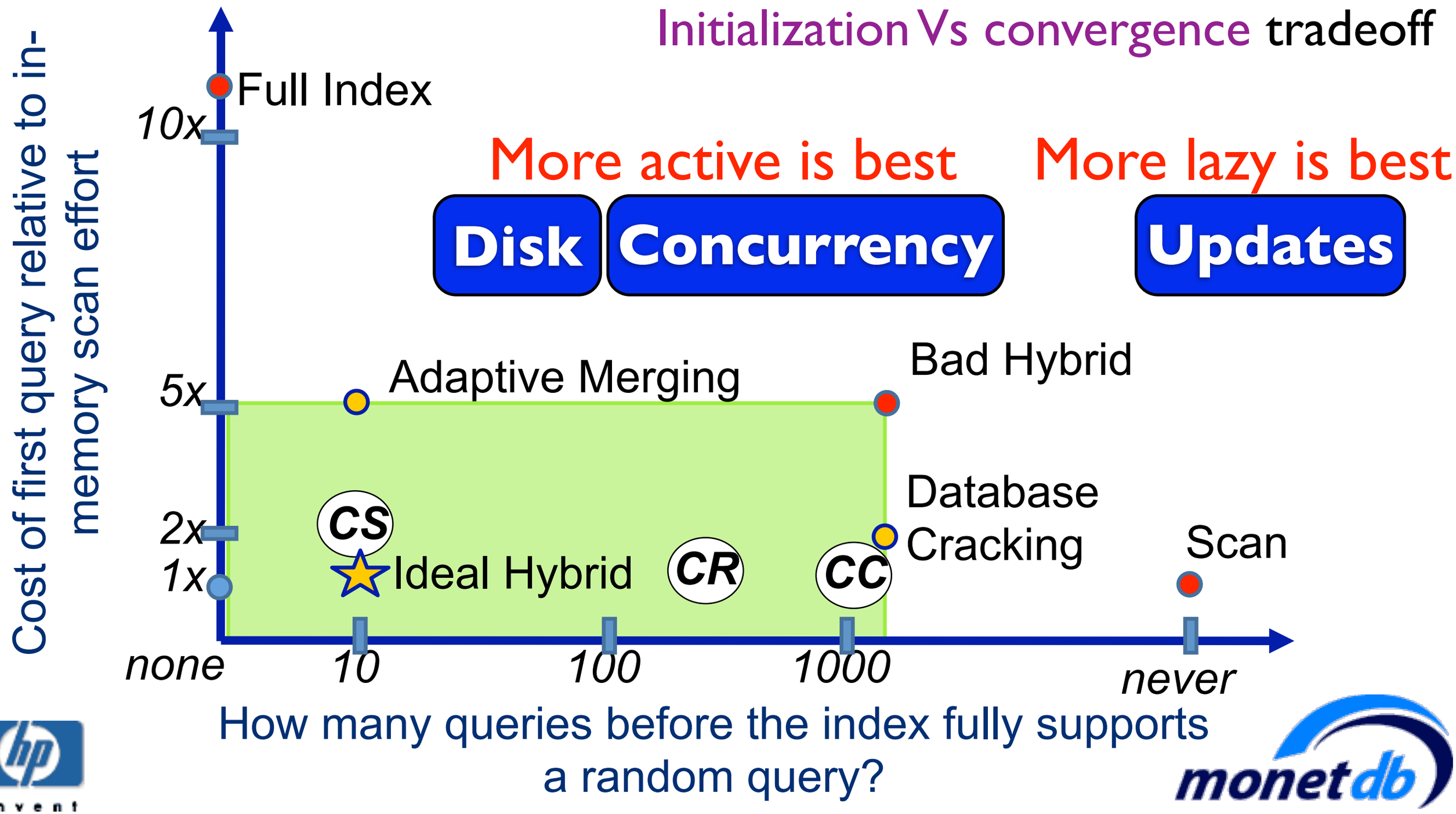


Adaptive Indexing

Initialization Vs convergence tradeoff



Adaptive Indexing



More in the paper...

Column-store design details and concerns

Selectivity effects

Concurrency control examples

and more...

CWI

Ongoing and open topics

Concurrency control

Compression

Multi-cores

Workload robustness

Disk based

Aggregations

Optimizer rules

Row-stores

Pipelining

Adaptive Indexing + Auto tuning tools

...and many more...

hp invent

monetdb

CWI

Ongoing and open topics

Concurrency control

Compression

Multi-cores

Workload robustness

Disk based

Thank you!

Aggregations

Optimizer rules

Row-stores

Pipelining

Adaptive Indexing + Auto tuning tools

...and many more...

hp invent

monetdb