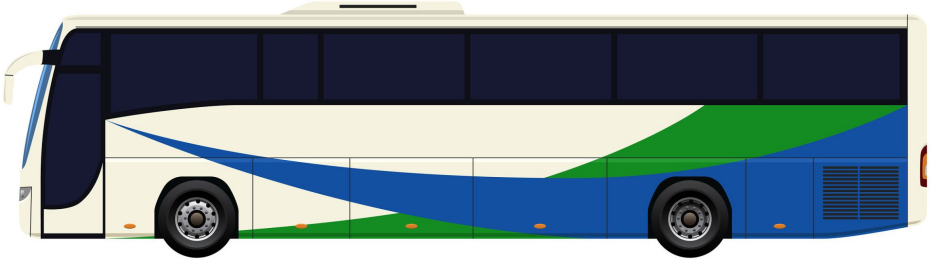
Access Path Selection in Main-Memory Optimized Data Systems:

# Should I Scan or Should I Probe?


Michael S. Kester  Manos Athanassoulis  Stratos Idreos


DASlab @ Harvard SEAS

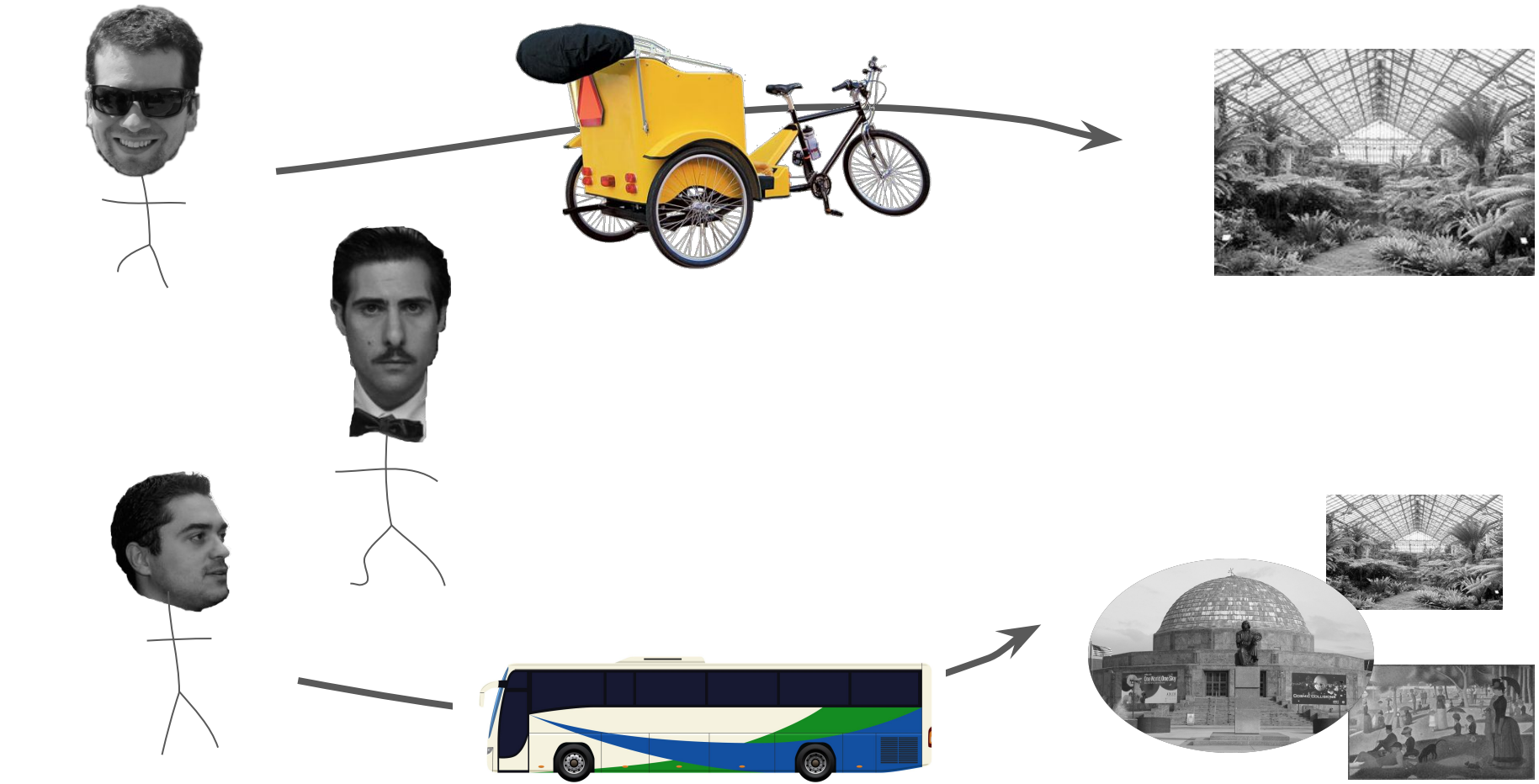



DASlab @ Harvard SEAS

?

DASlab @ Harvard SEAS


CONCIERGE
KEEP THIS DOOR CLOSED AT ALL TIMES
KEEP THIS DOOR CLOSED ALL TIMES
CONCIERGE HOURS
M-F: 9 AM – 8 PM
LUNCH: 12 PM – 2 PM
SUNDAY: CLOSED
RING FOR SERVICE
GUEST DELIVERIES: SIGN REGISTER

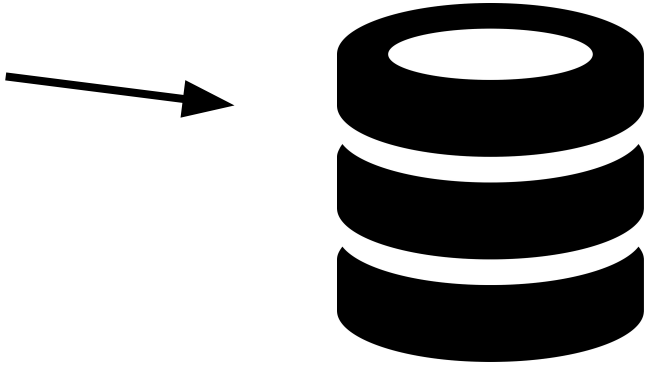



Hilton Chicago
Forward Together

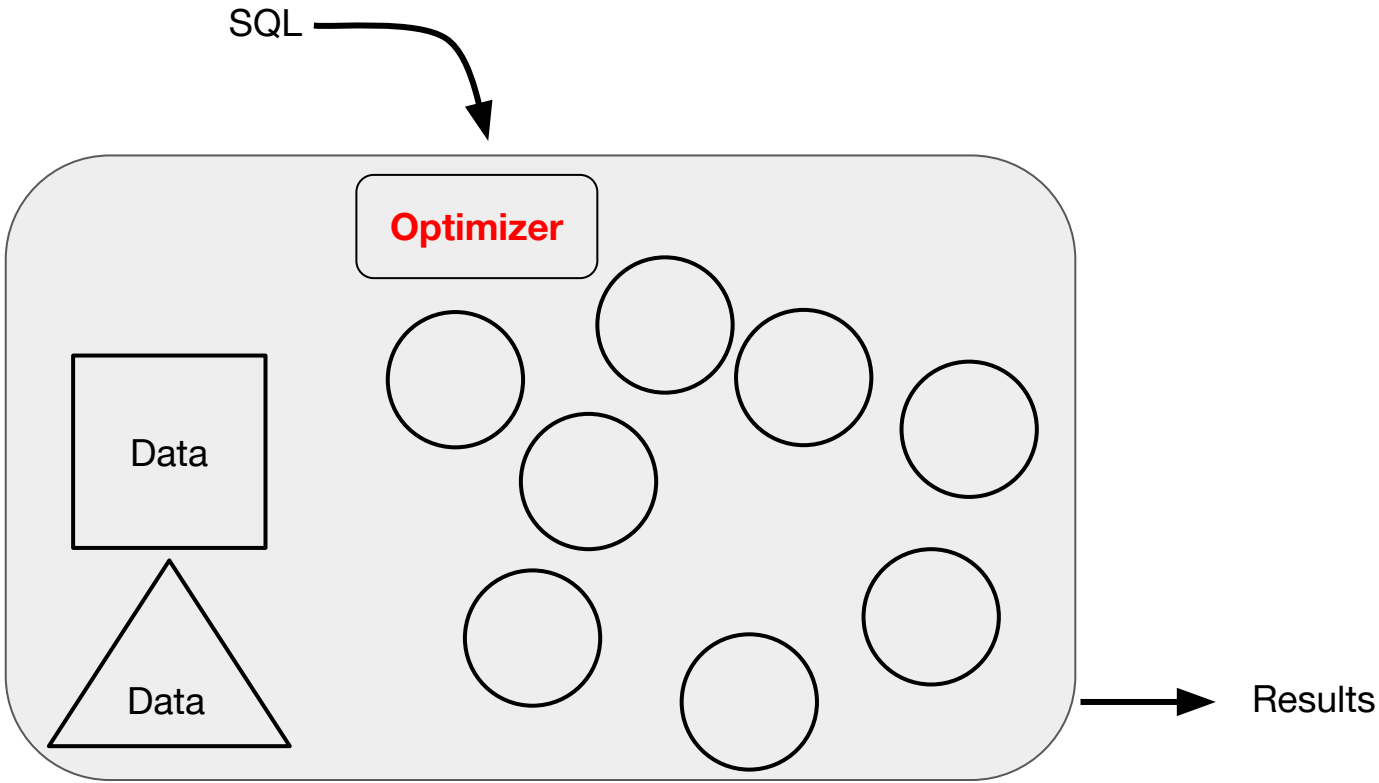

DASlab @ Harvard SEAS

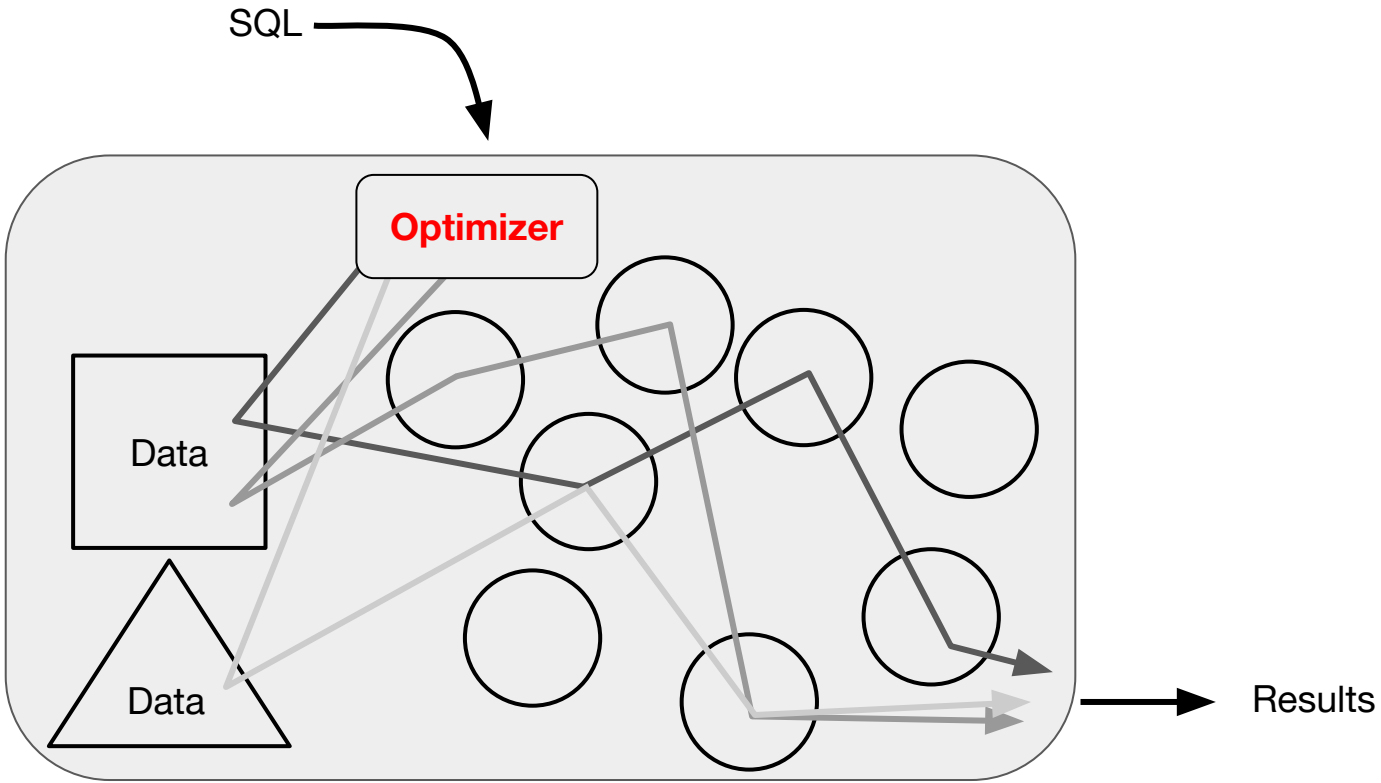
# Ask for what you want...

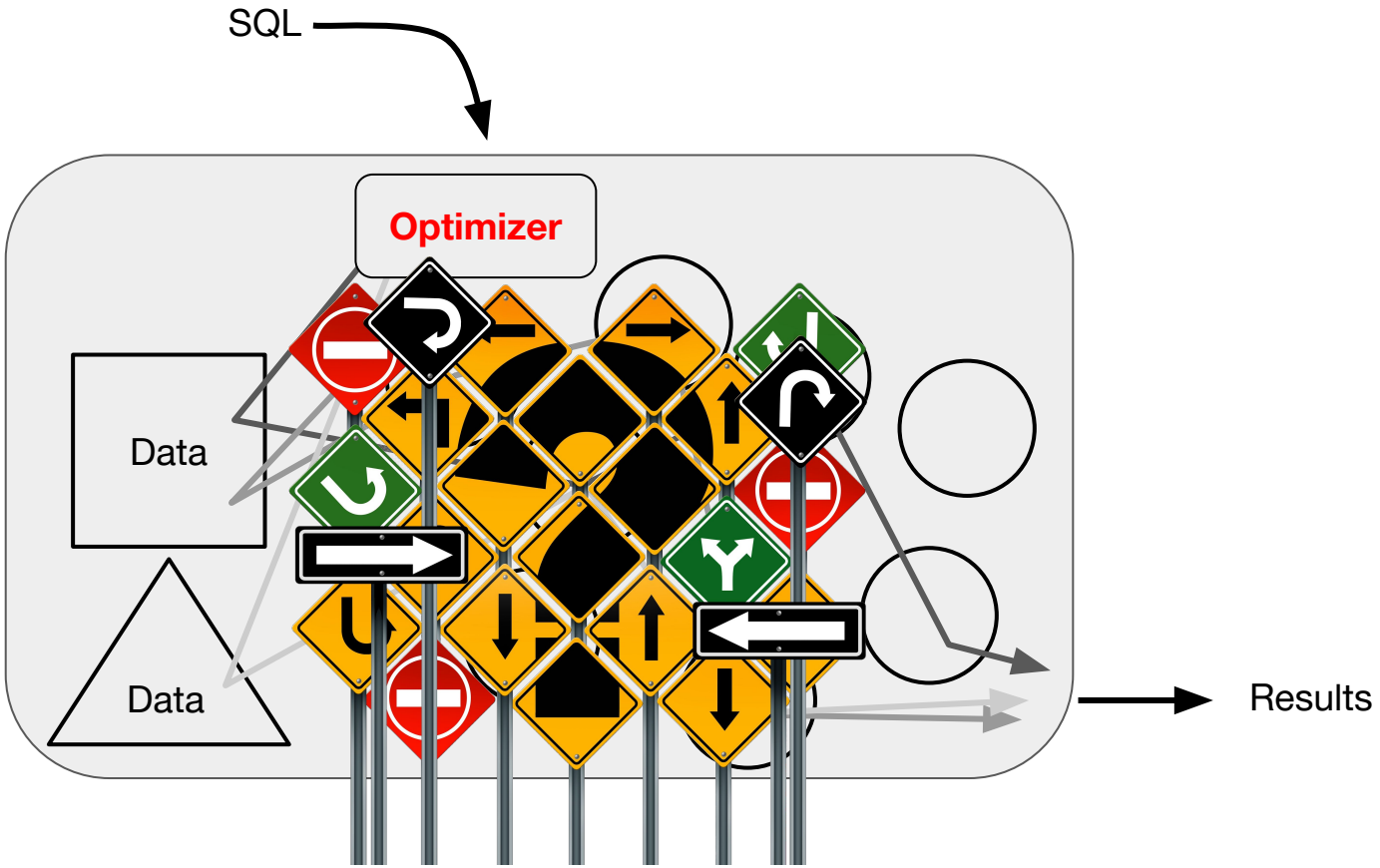
```
SELECT x
FROM table_A
WHERE
      y < 10;
```



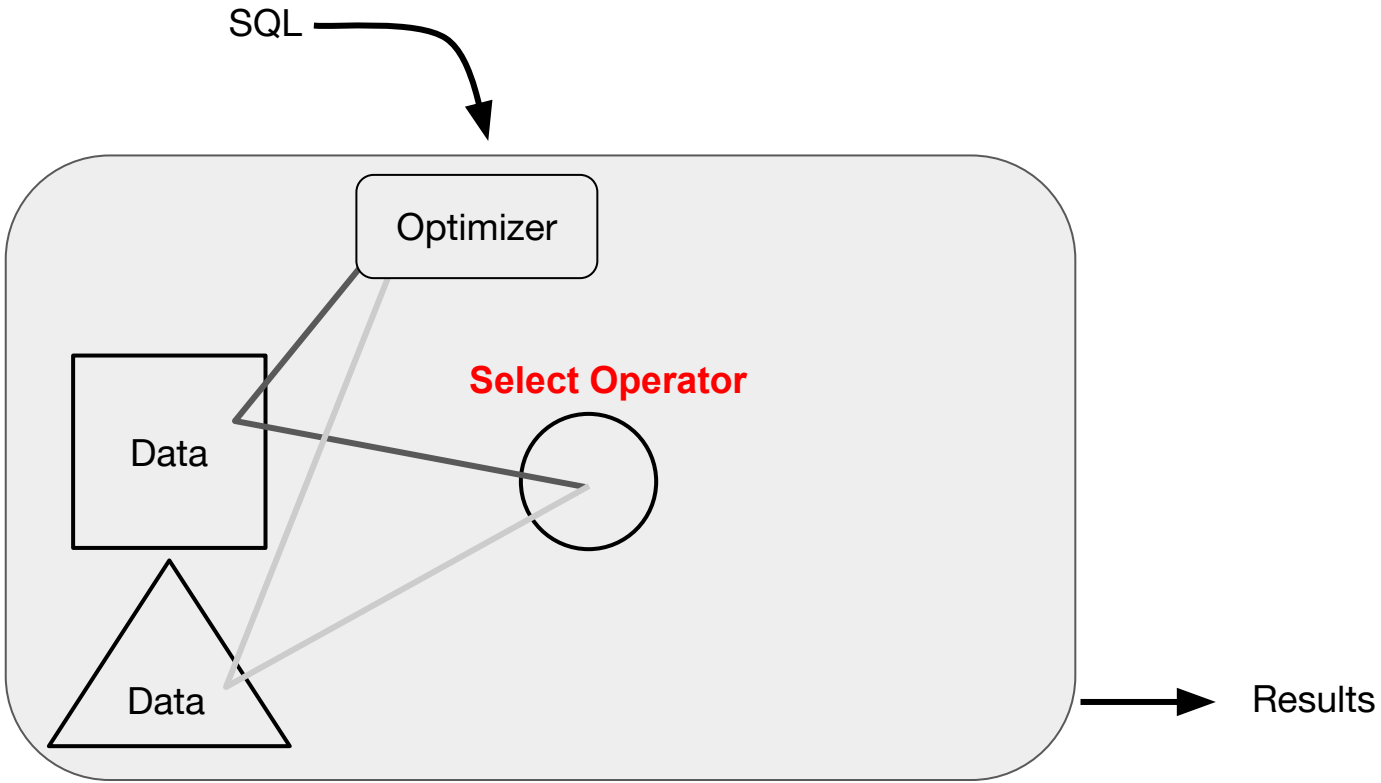
...the system chooses how to retrieve it.


SQL
Optimizer
Data
Data
Results


SQL
Optimizer
Data
Data
Results




DASlab @ Harvard SEAS


SQL
Optimizer
Select Operator
Data
Data
Results

```
SELECT x
FROM table_A
WHERE
    y < 10;
```

```
SELECT x
FROM table_A
WHERE
    y < 10;
```

```
SELECT x
FROM table_A
WHERE
      y < 10;
```

```
select /* TPC_H Query 17 */
sum(l_extendedprice) / 7.0 as
avg_yearly
from
lineitem,
part
where
p_partkey = l_partkey
and p_brand = '[BRAND]'
and p_container =
'[CONTAINER]'
and l_quantity < (
select
0.2 * avg(l_quantity)
from
lineitem
where
l_partkey = p_partkey
)
```

```
SELECT x
FROM table_A
WHERE
      y < 10;
```

```
select /* TPC_H Query 17 */
sum(l_extendedprice) / 7.0 as
avg_yearly
from
lineitem,
part
where
p_partkey = l_partkey
and p_brand = '[BRAND]'
and p_container =
'[CONTAINER]'
and l_quantity < (
select
0.2 * avg(l_quantity)
from
lineitem
where
l_partkey = p_partkey
)
```

```
SELECT x
FROM table_A
WHERE
      y < 10;
```

```
select /* TPC_H Query 17 */
sum(l_extendedprice) / 7.0 as
avg_yearly
from
lineitem,
part
where
p_partkey = l_partkey
and p_brand = '[BRAND]'
and p_container =
'[CONTAINER]'
and l_quantity < (
select
0.2 * avg(l_quantity)
from
lineitem
where
l_partkey = p_partkey
)
```

```
SELECT F1.PARCELID, F1.FDATE
FIRE1, F2.FDATE FIRE2,
F2.FDATE - F1.FDATE INTERVAL
FROM FIRES F1, FIRES F2
WHERE F1.PARCELID =
F2.PARCELID AND F2.FDATE >
F1.FDATE AND F2.FDATE <=
F1.FDATE + 300;
```

```
SELECT x
FROM table_A
WHERE
      y < 10;
```

```
select /* TPC_H Query 17 */
sum(l_extendedprice) / 7.0 as
avg_yearly
from
lineitem,
part
where
p_partkey = l_partkey
and p_brand = '[BRAND]'
and p_container =
'[CONTAINER]'
and l_quantity < (
select
0.2 * avg(l_quantity)
from
lineitem
where
l_partkey = p_partkey
)
```

```
SELECT F1.PARCELID, F1.FDATE
FIRE1, F2.FDATE FIRE2,
F2.FDATE - F1.FDATE INTERVAL
FROM FIRES F1, FIRES F2
WHERE F1.PARCELID =
F2.PARCELID AND F2.FDATE >
F1.FDATE AND F2.FDATE <=
F1.FDATE + 300;
```

```
select user_id, count(*) as how_many
from bboard
where not exists (select 1 from
                  bboard_authorized_maintainers bam
                  where bam.user_id = bboard.user_id)
and posting_time + 60 > sysdate
group by user_id
order by how_many desc;
```

```
SELECT x
FROM table_A
WHERE
      y < 10;
```

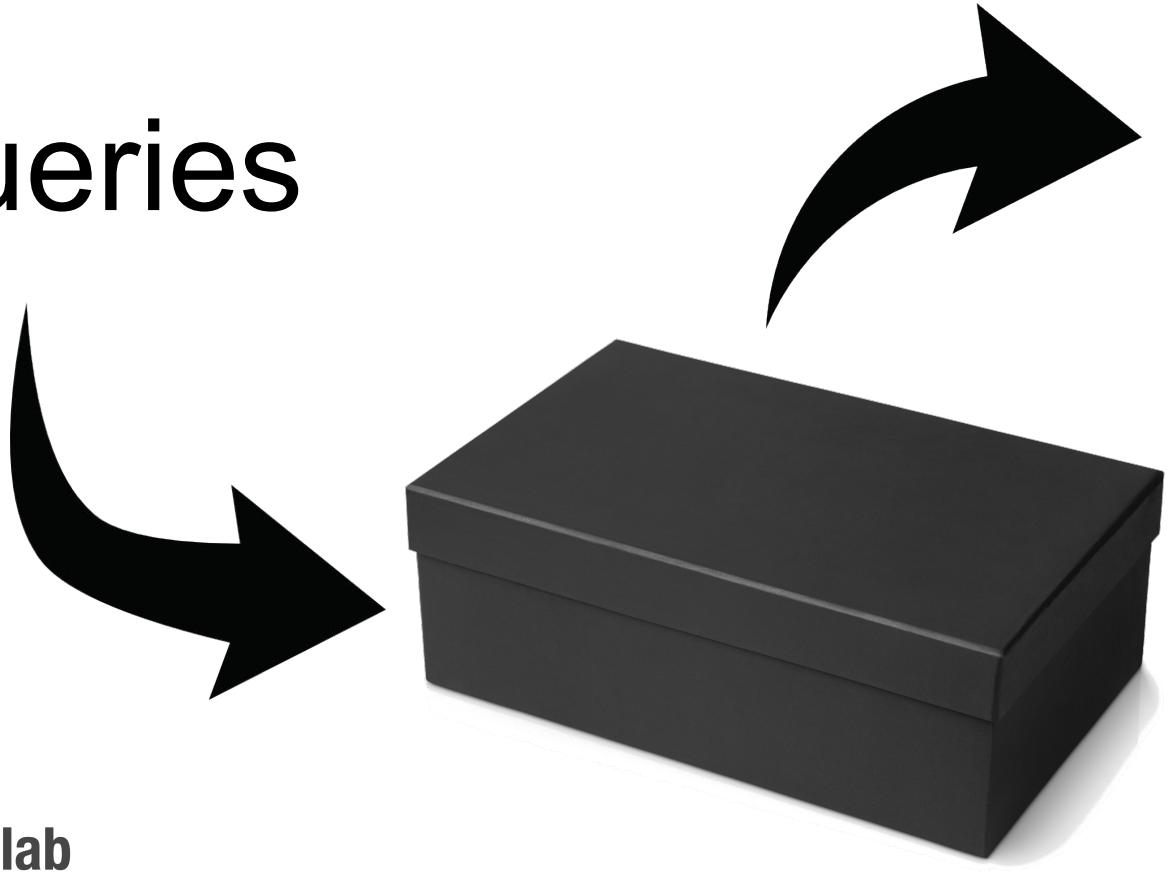
```
SELECT *
FROM exercise_logs
WHERE
(minutes > 10
   AND
calories > 50)
   OR
heart_rate > 110;
```

```
select /* TPC_H Query 17 */
sum(l_extendedprice) / 7.0 as
avg_yearly
from
lineitem,
part
where
p_partkey = l_partkey
and p_brand = '[BRAND]'
and p_container =
'[CONTAINER]'
and l_quantity < (
select
0.2 * avg(l_quantity)
from
lineitem
where
l_partkey = p_partkey
)
```

```
SELECT F1.PARCELID, F1.FDATE
FIRE1, F2.FDATE FIRE2,
F2.FDATE - F1.FDATE INTERVAL
FROM FIRES F1, FIRES F2
WHERE F1.PARCELID =
F2.PARCELID AND F2.FDATE >
F1.FDATE AND F2.FDATE <=
F1.FDATE + 300;
```

```
select user_id, count(*) as how_many
from bboard
where not exists (select 1 from
                  bboard_authorized_maintainers bam
                  where bam.user_id = bboard.user_id)
and posting_time + 60 > sysdate
group by user_id
order by how_many desc;
```

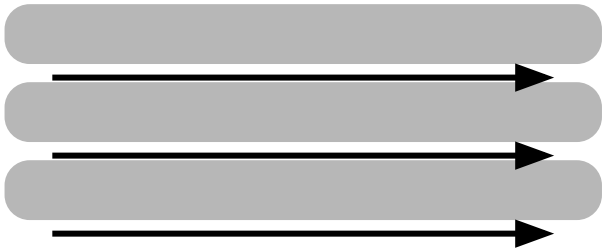
Queries



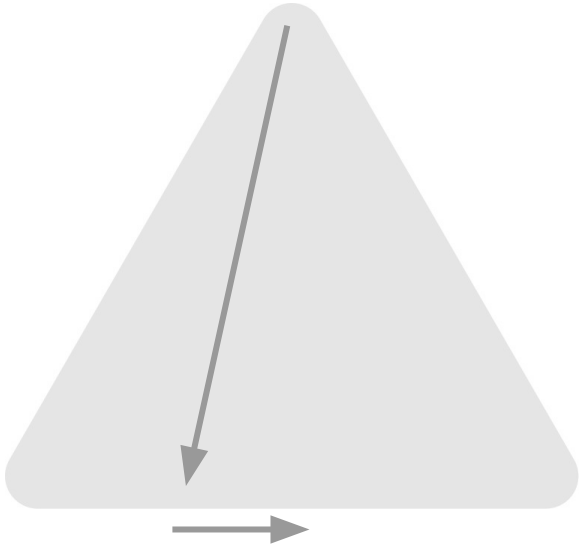
Results

DASlab @ Harvard SEAS

# Access Path Choices

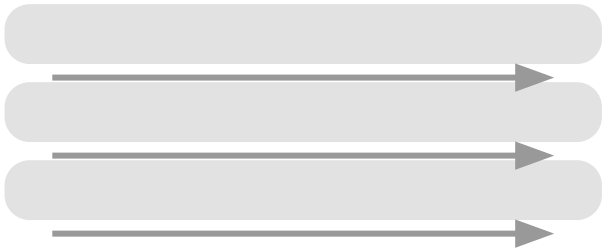


Full Base Data Scan

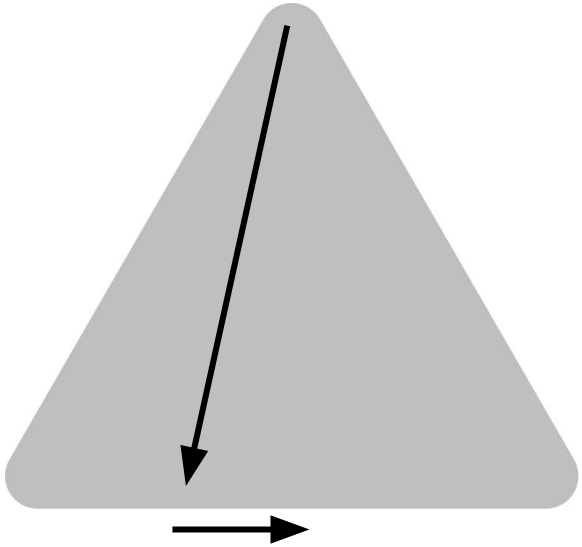


Secondary Index Scan
(Auxilliary copy of the data + structure)

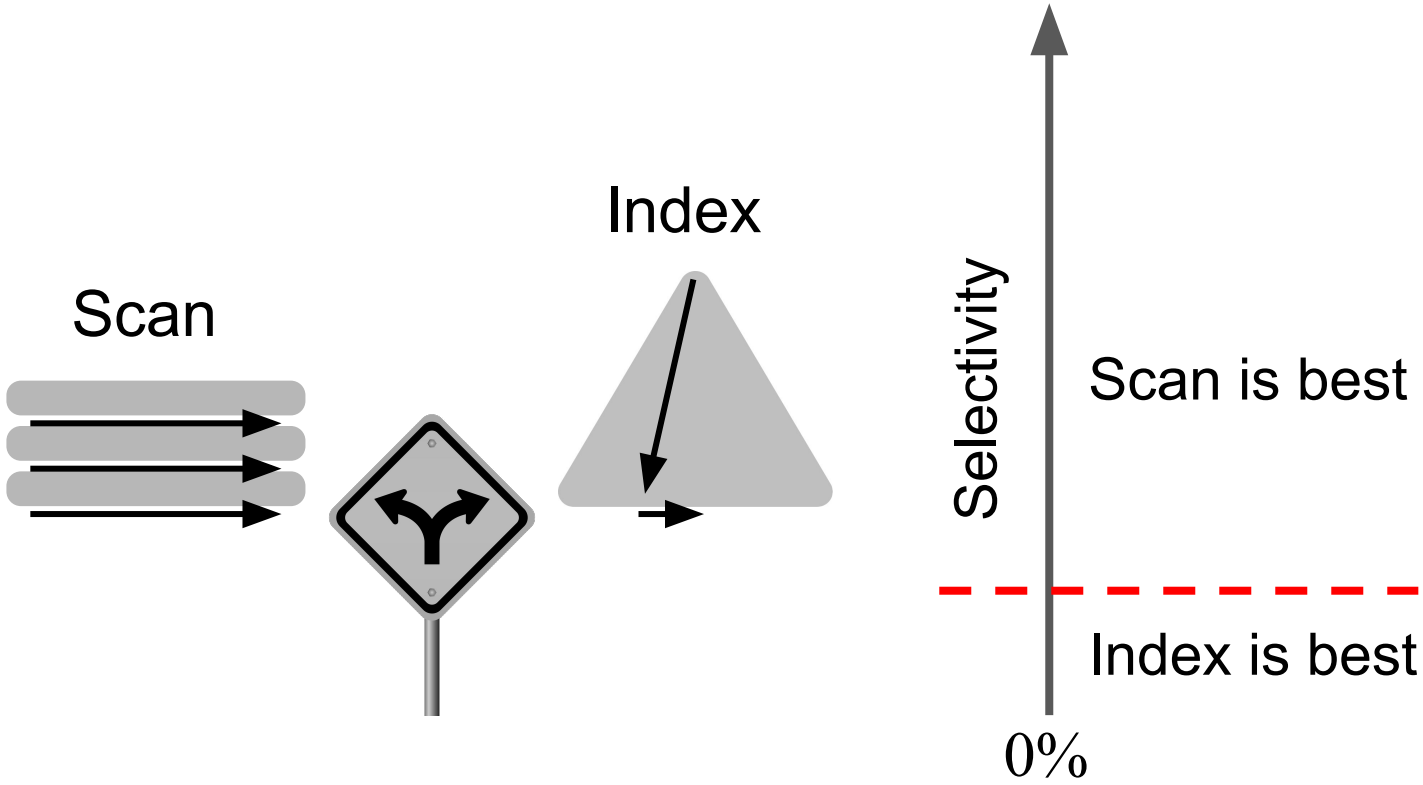
# Access Path Choices



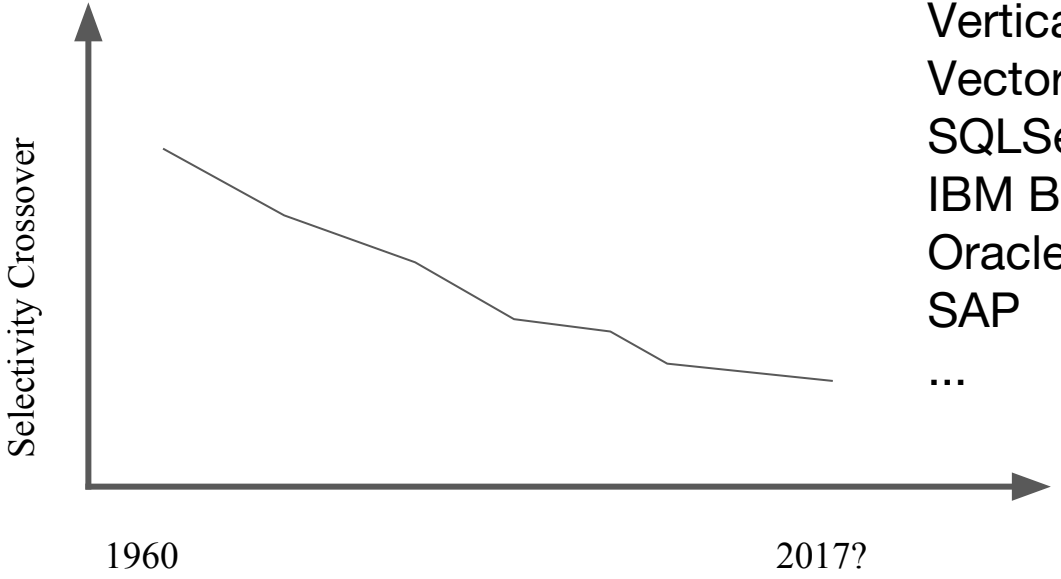
Full Base Data Scan



Secondary Index Scan
(Auxilliary copy of the data + structure)



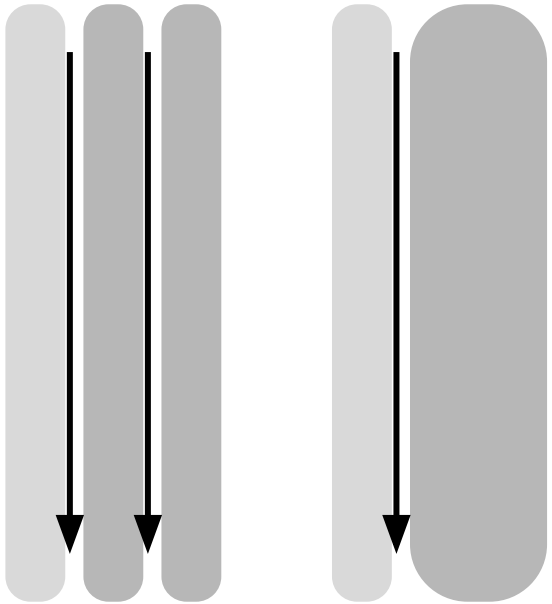

A fixed threshold



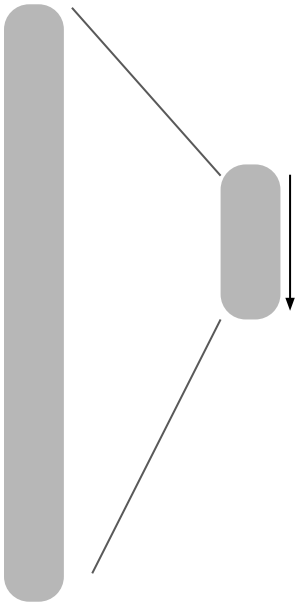

Vertica / C-store
Vectorwise / MonetDB
SQLServer
IBM BLU
Oracle
SAP
...

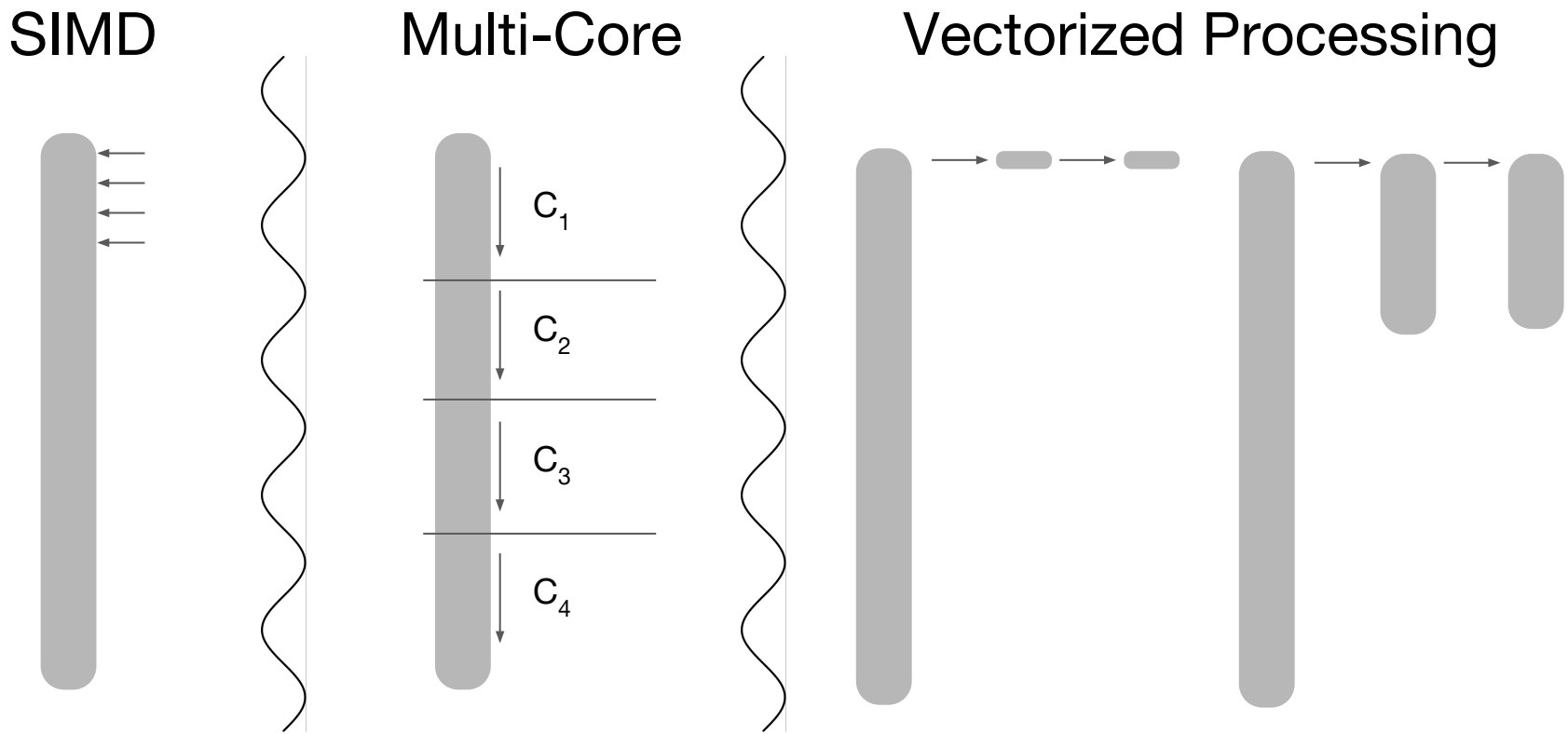
# Increasing pressure on indexes

# Column(-group) Orientation



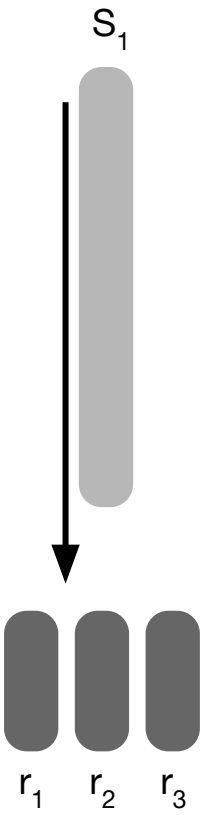
# Compression



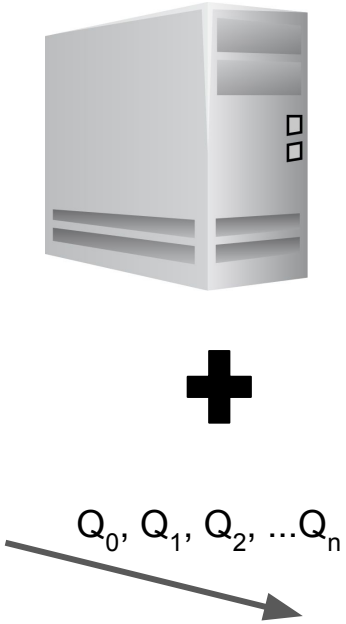

SIMD
Multi-Core
Vectorized Processing
C1
C2
C3
C4

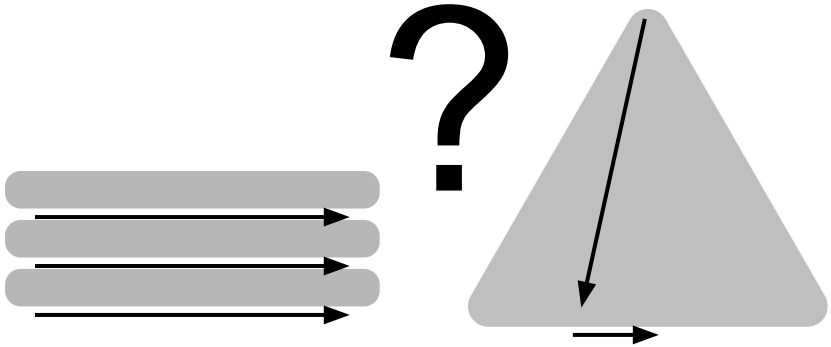
## 1:1 Scan to Result

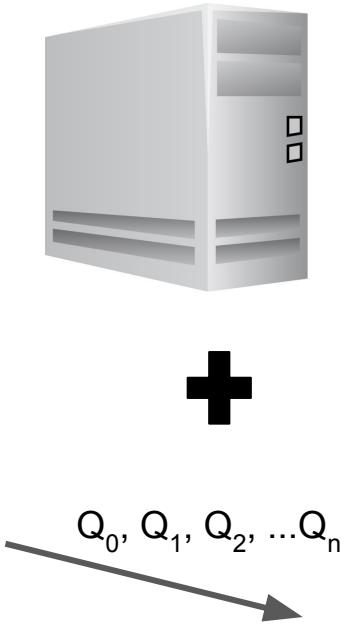

## Shared Scans



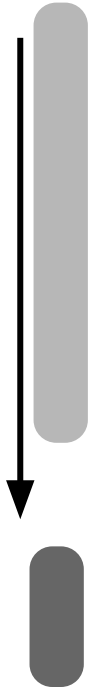

DASlab @ Harvard SEAS


$Q_0, Q_1, Q_2, ...Q_n$

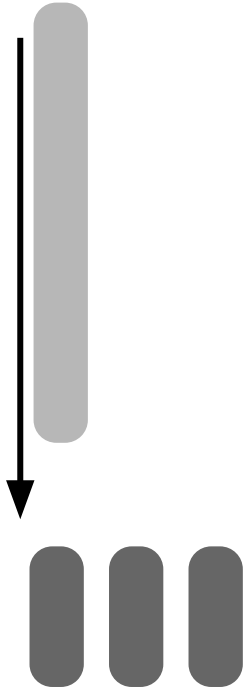

?



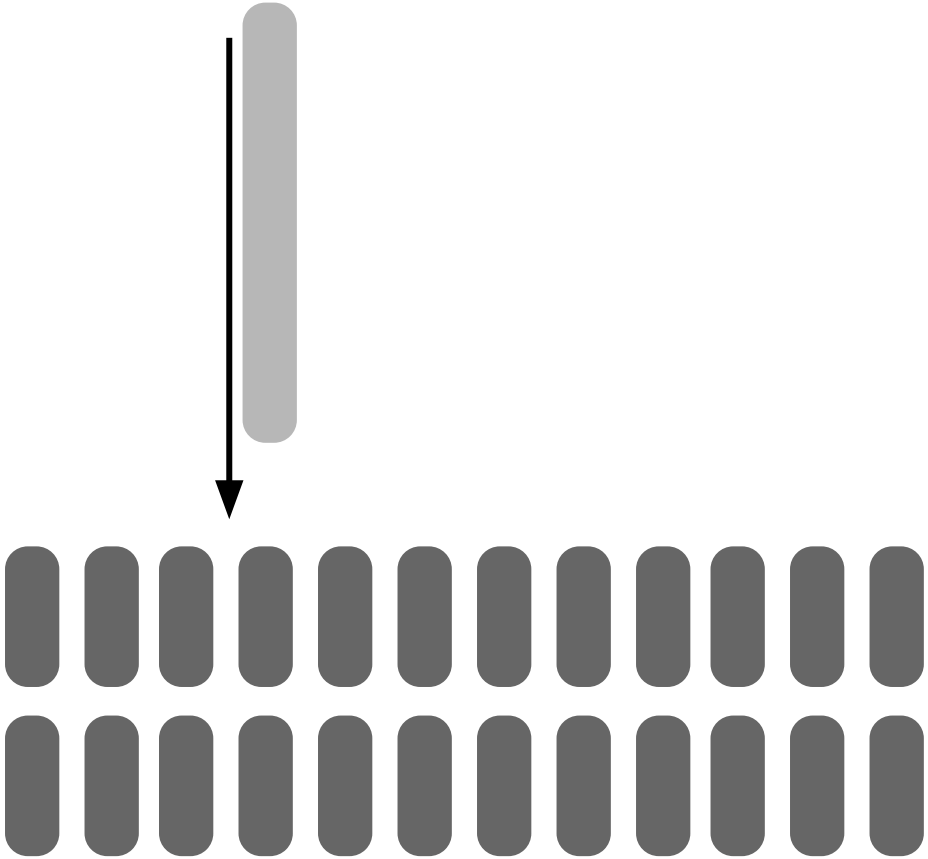

# Are indexes ever needed?



# Are indexes ever needed?

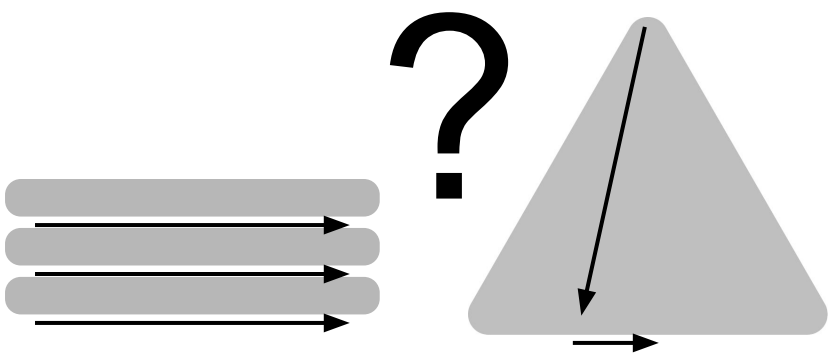


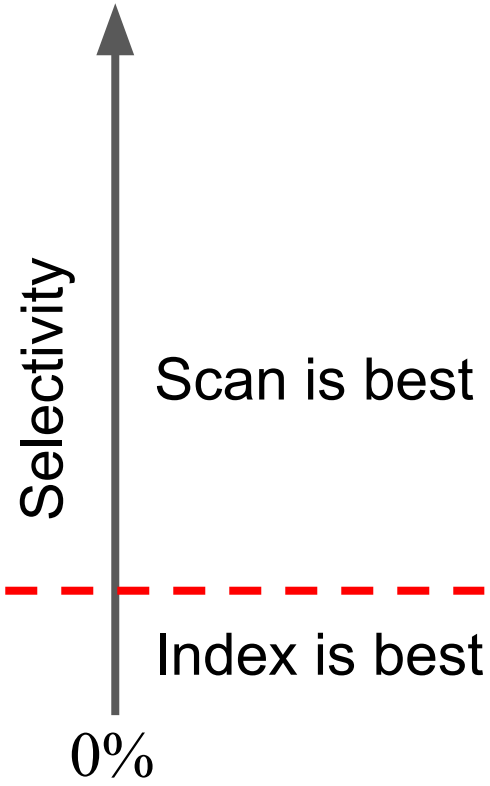
# Are indexes ever needed?



# Are indexes ever needed?

If so, how should the optimizer choose an access path?

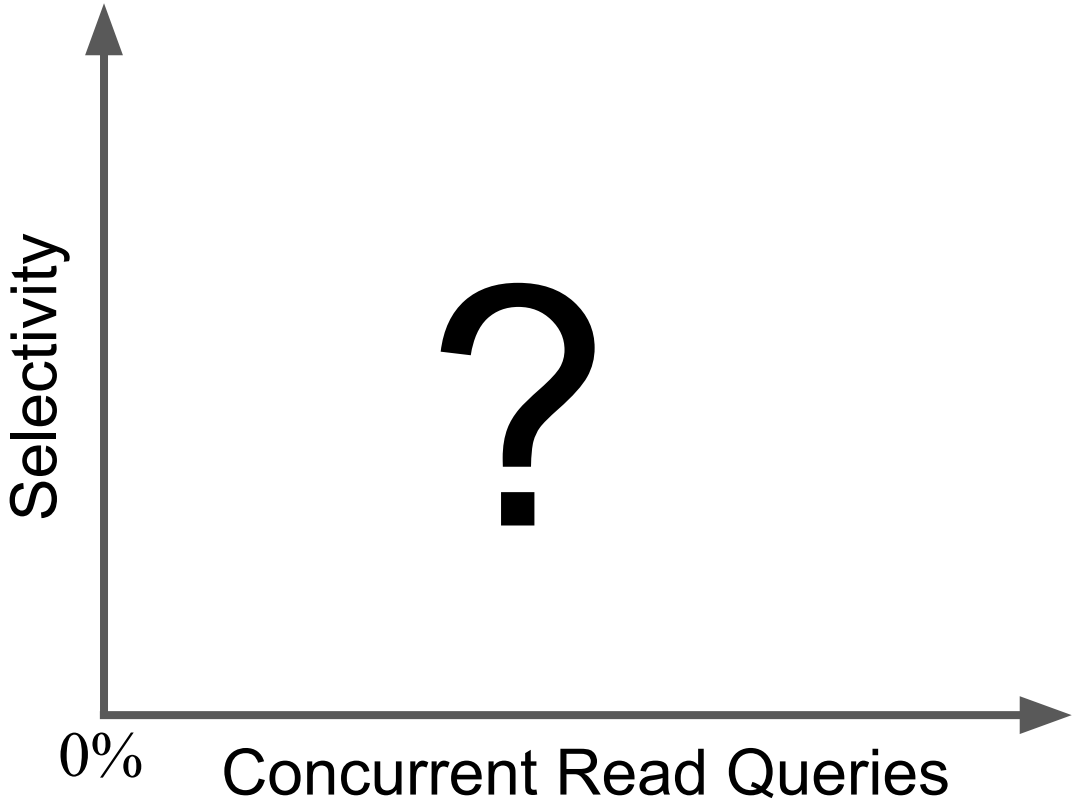




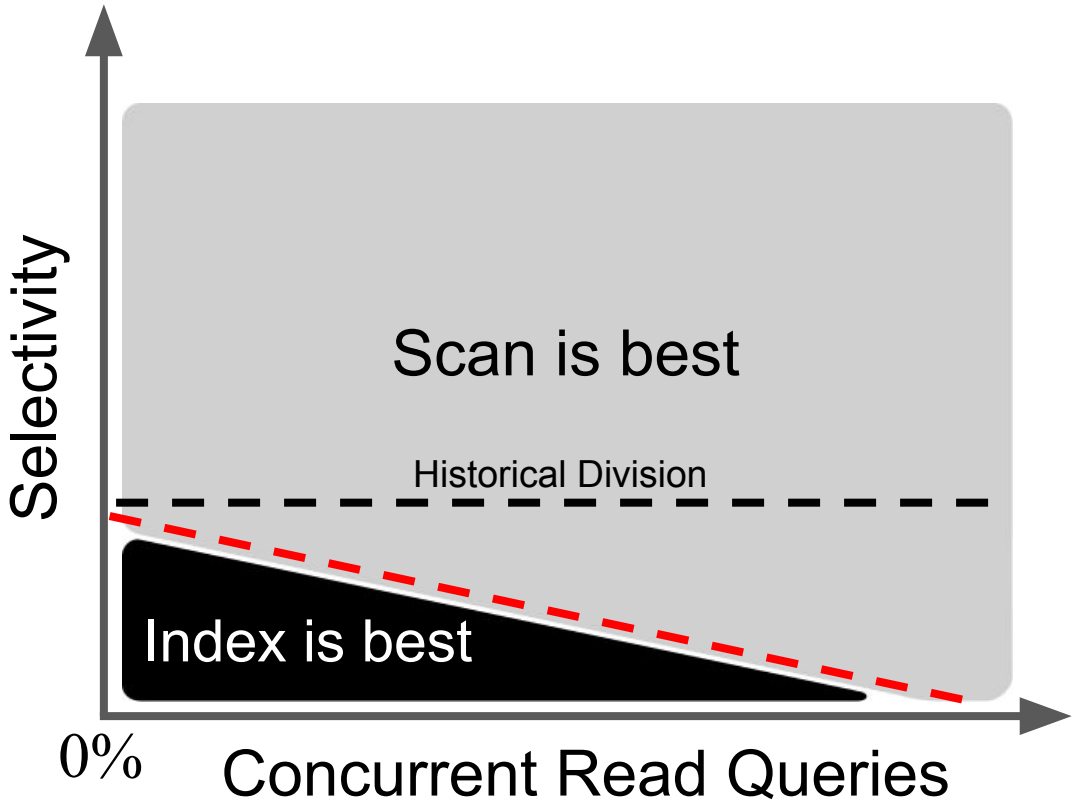

Access Path Selection
in a Relational Database Management System

P. Griffiths Selinger
M. M. Astrahan
D. D. Chamberlin
R. A. Lorie
T. G. Price

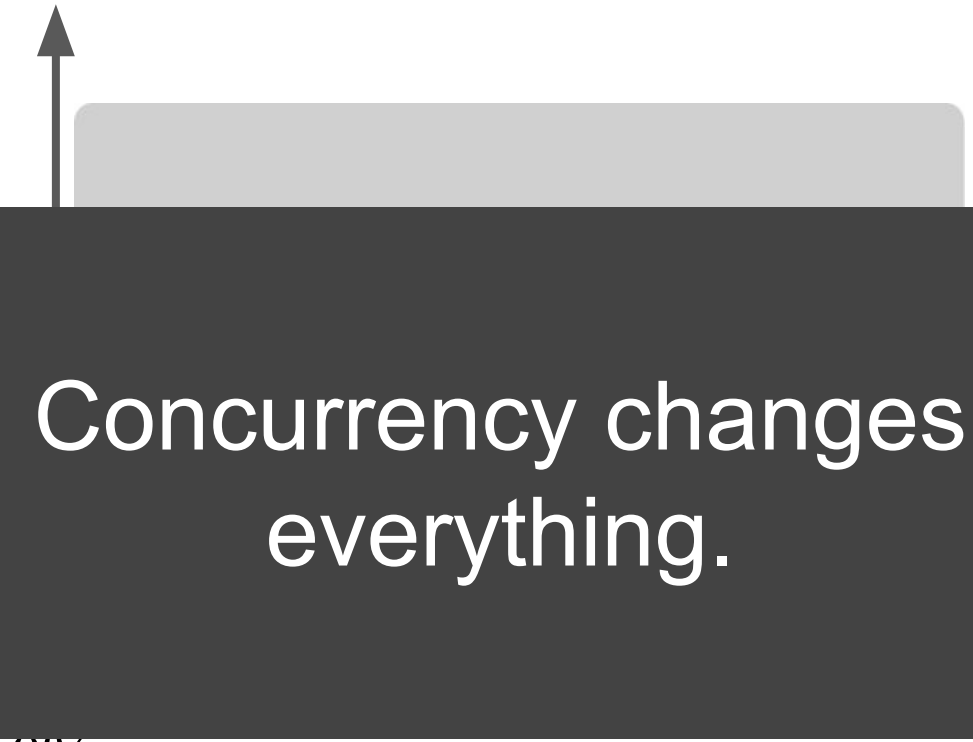
IBM Research Division, San Jose, Californ




DASlab @ Harvard SEAS



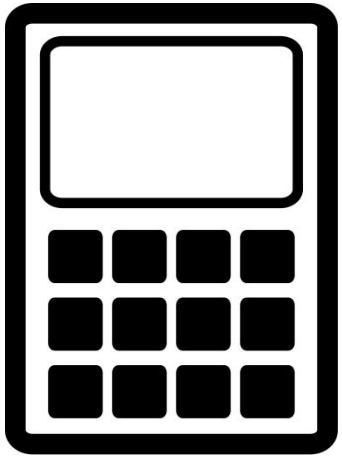

DASlab @ Harvard SEAS


Concurrency changes everything.
0%
Concurrent Read Queries

?

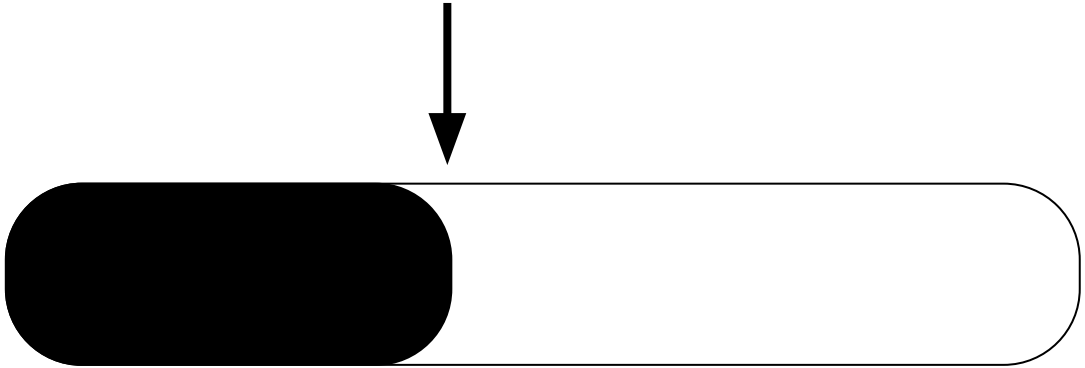
$$\text{APS} = \frac{\text{Index scan}}{\text{Sequential scan}}$$

$$\text{APS} = \frac{\text{Index scan}}{\text{[redacted]} + ? + ?}$$

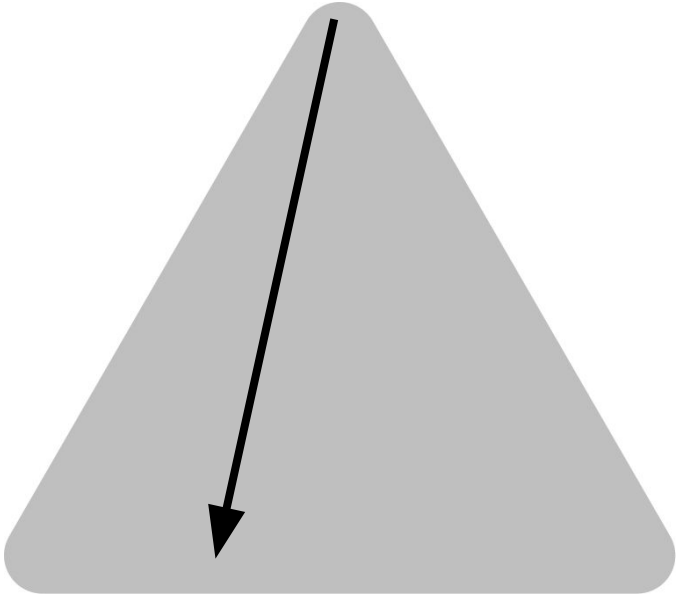


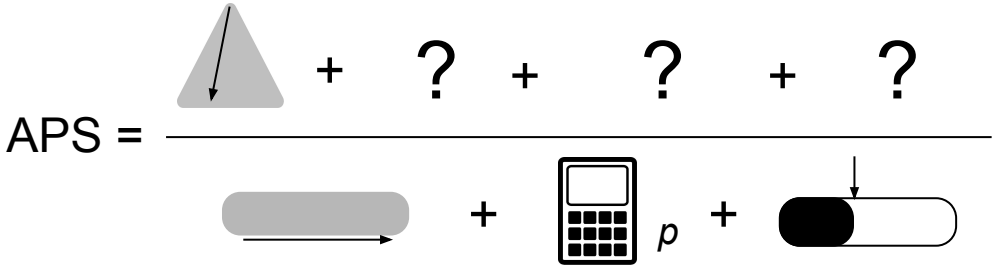
*p*

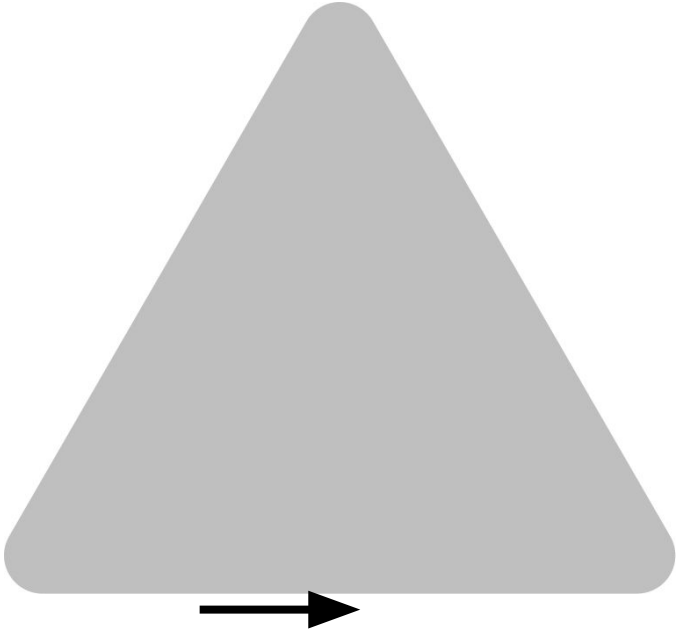
$$\text{APS} = \frac{\text{Index scan}}{\text{[redacted]} + \text{[redacted]}\ p + ?}$$

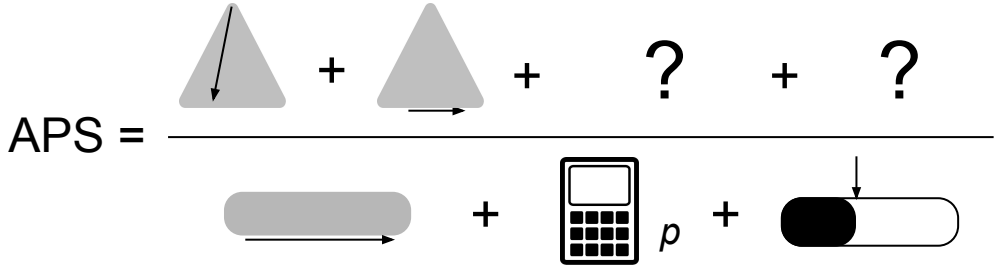


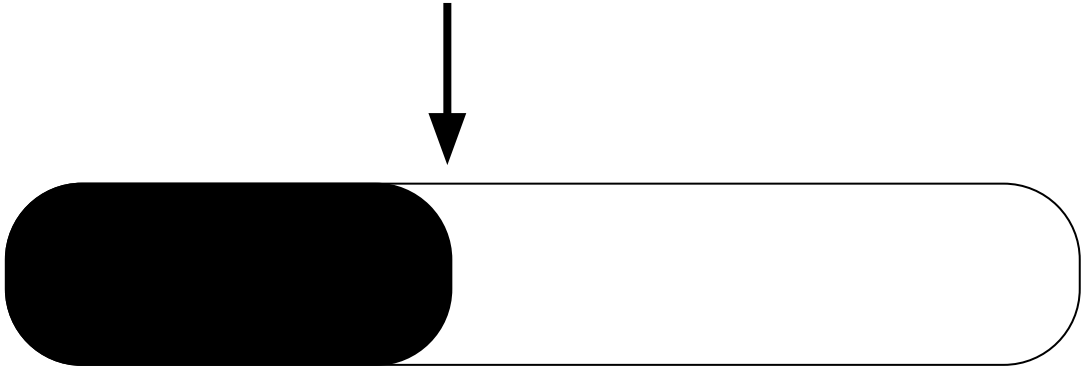
$$\text{APS} = \frac{\text{Index scan}}{\text{[arrow]} + \text{[calculator]}\ p + \text{[switch]}}$$

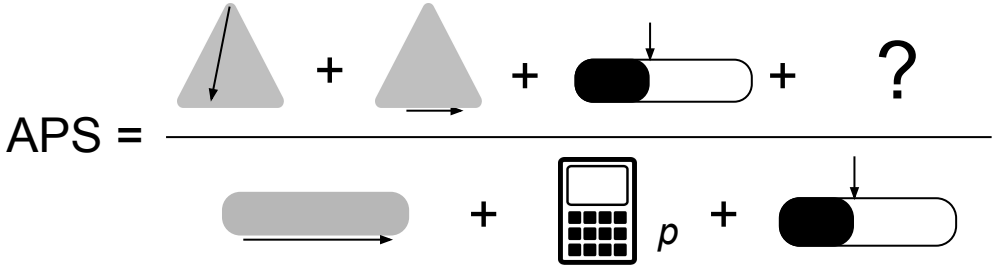


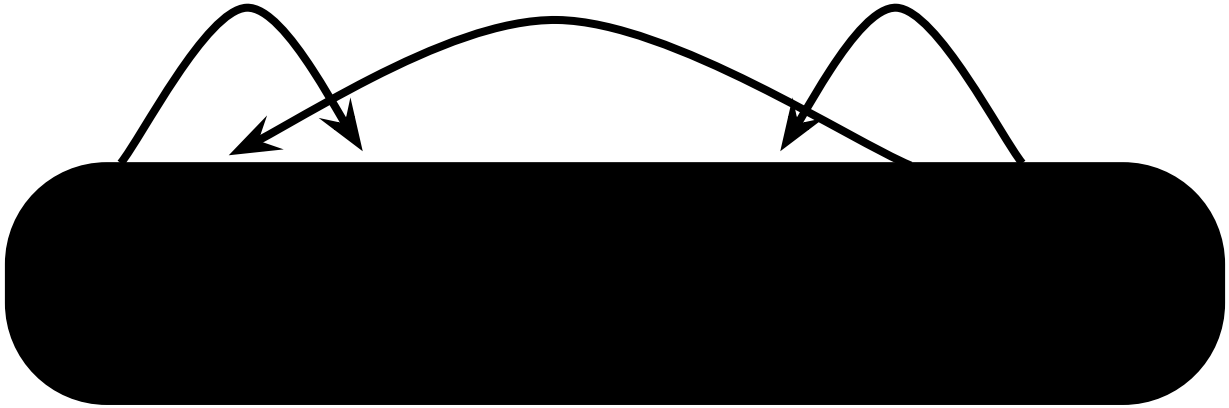

APS =
+ ? + ? + ?
+ p +



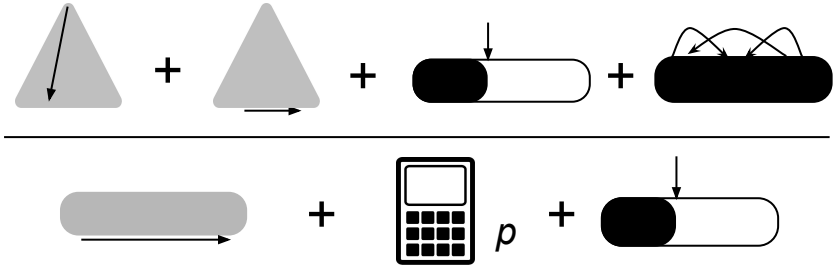

APS =
+
+
?
+
?
+
p
+



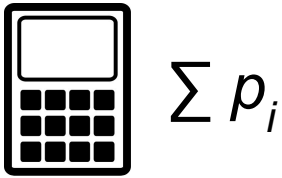

APS =
+
+
+
?
+
p
+



Multiquery APS =

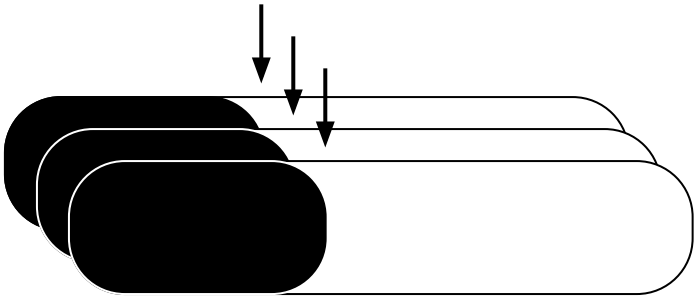



Predicate evaluation

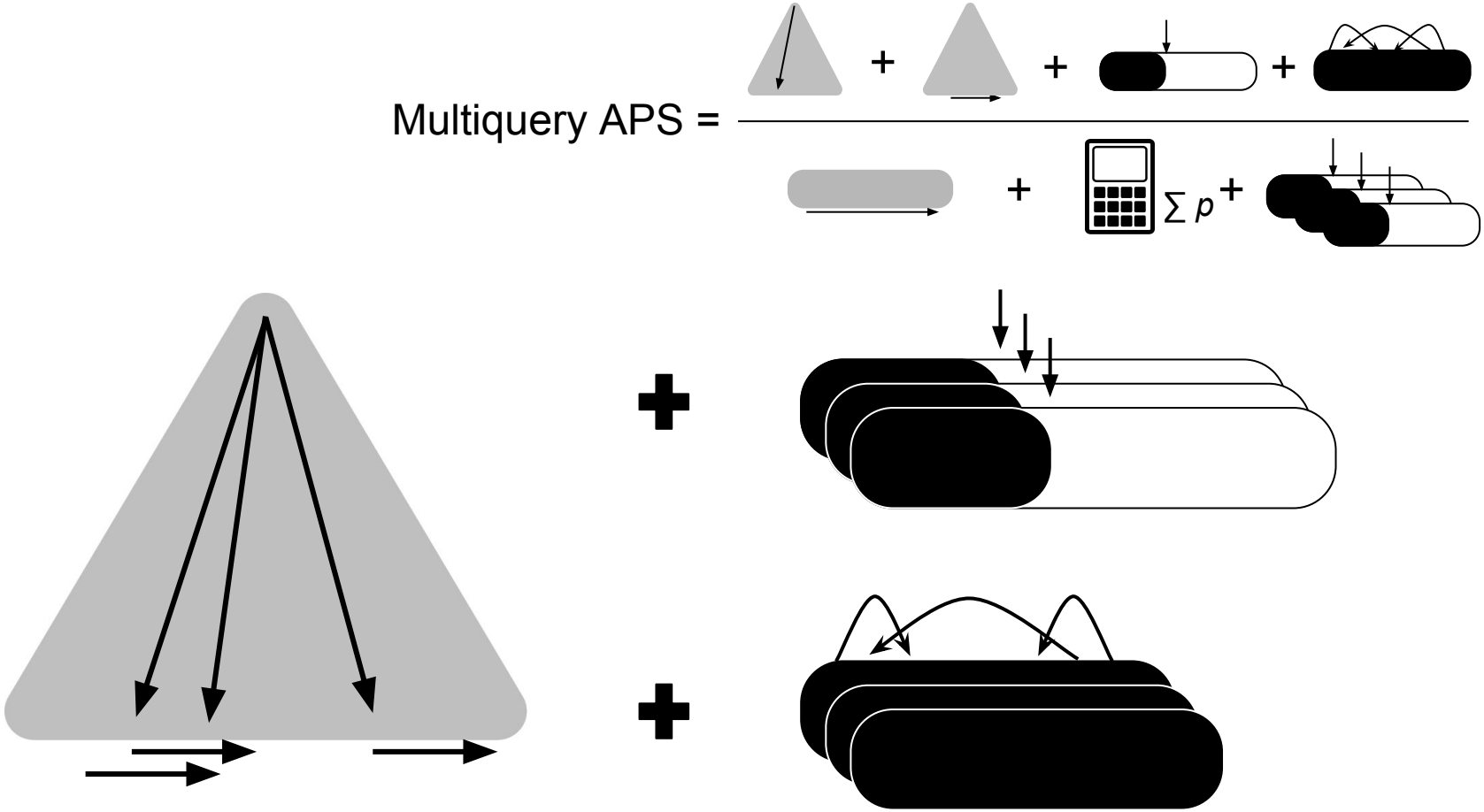


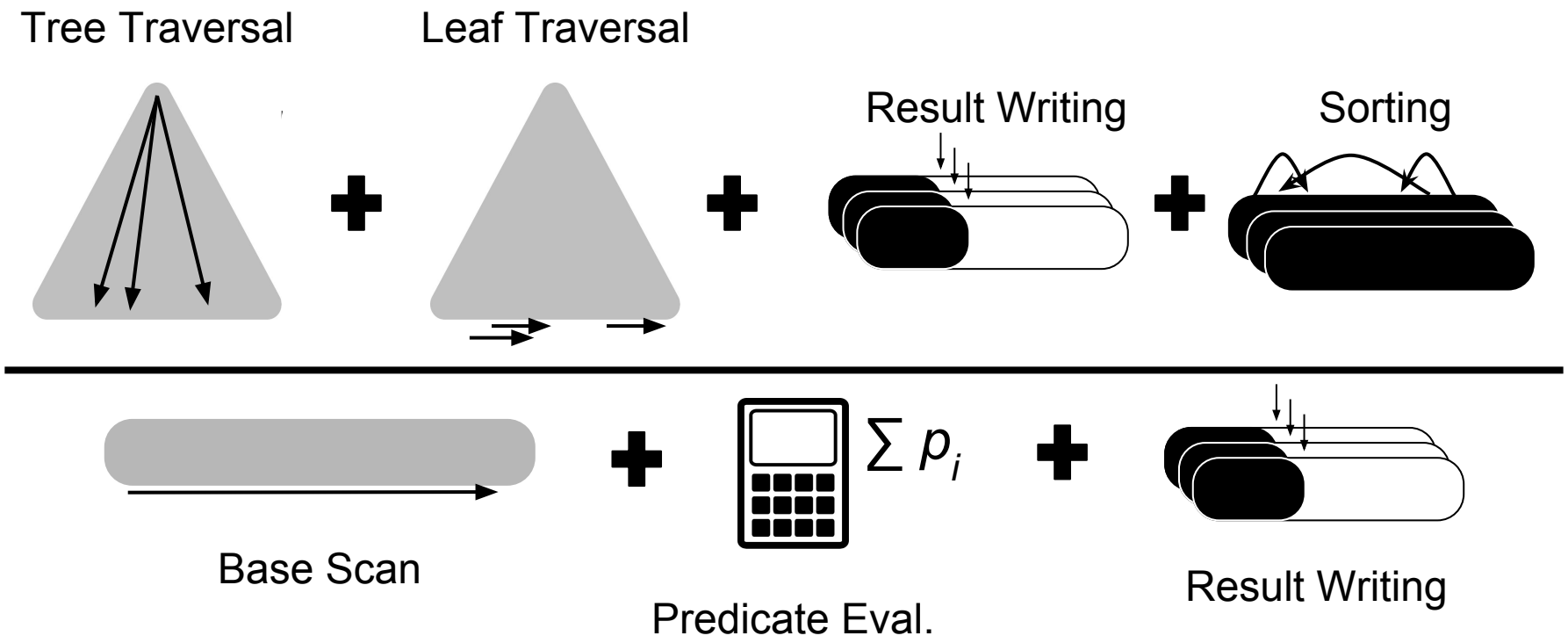
$\sum p_i$

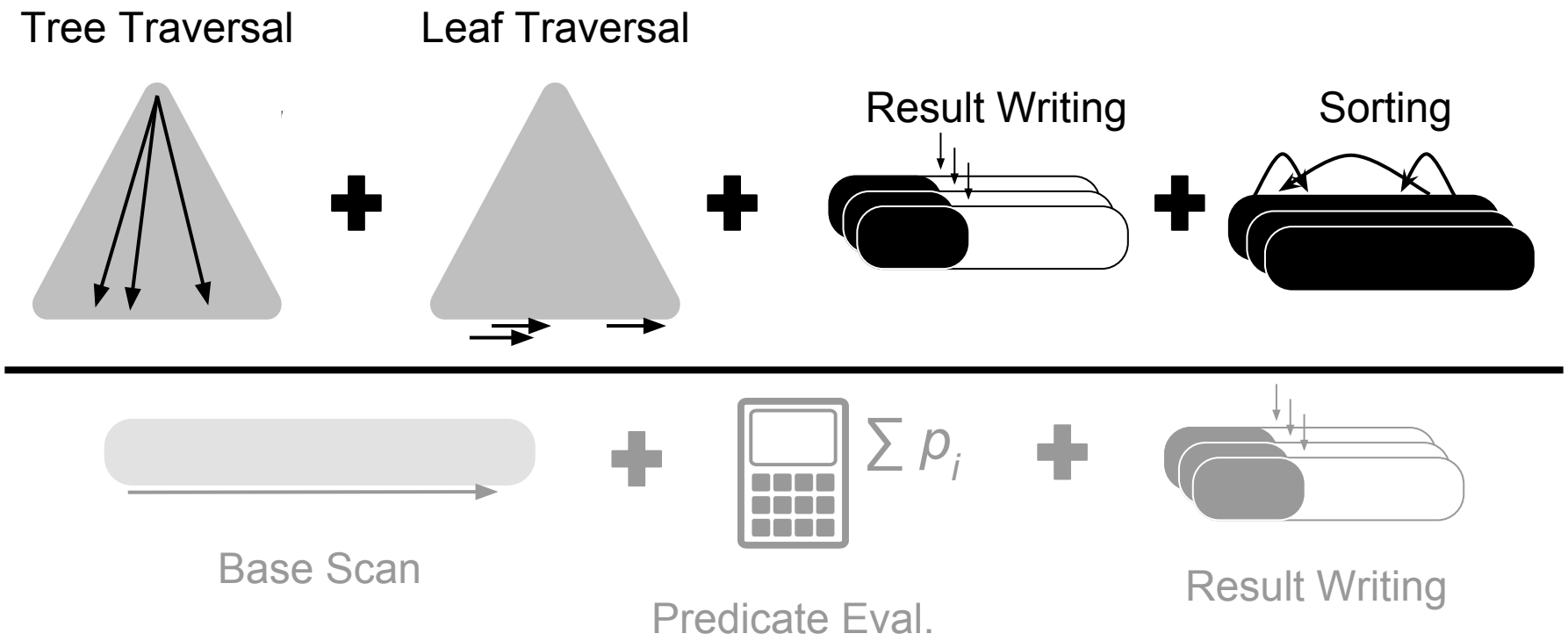
Result set writing

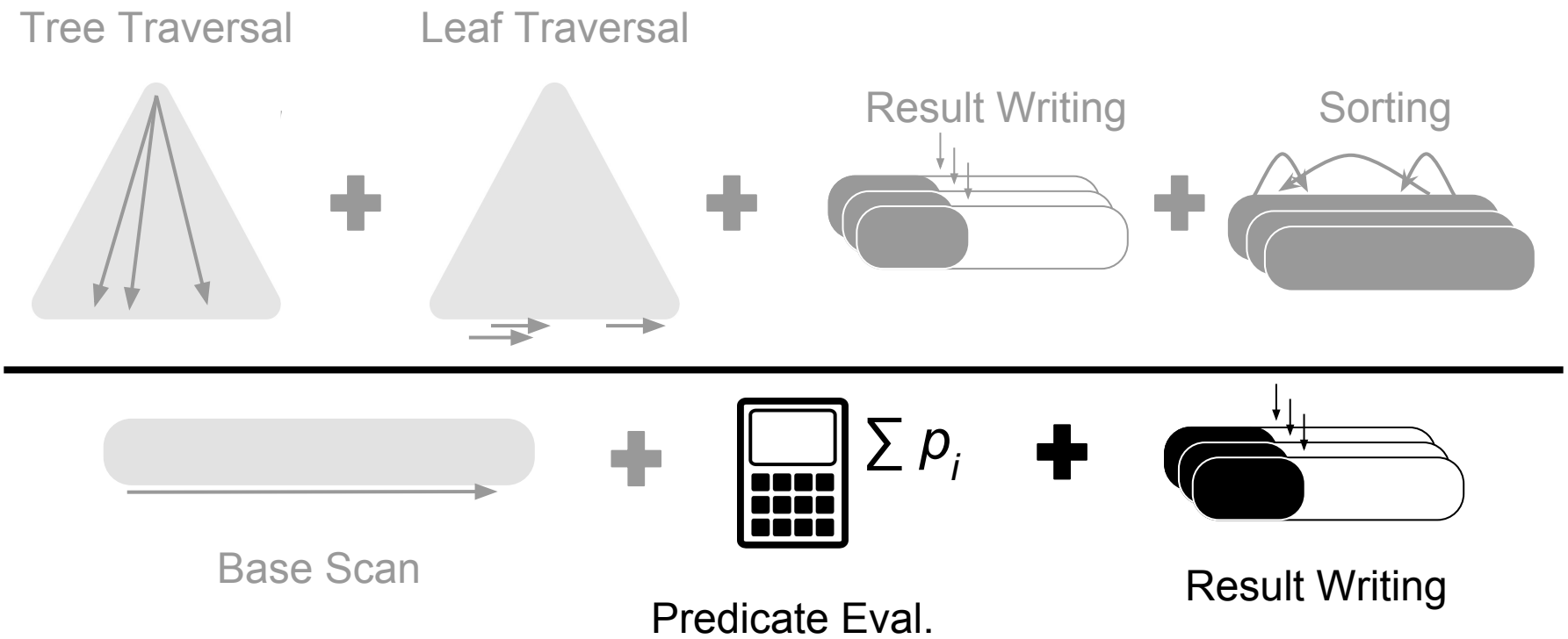


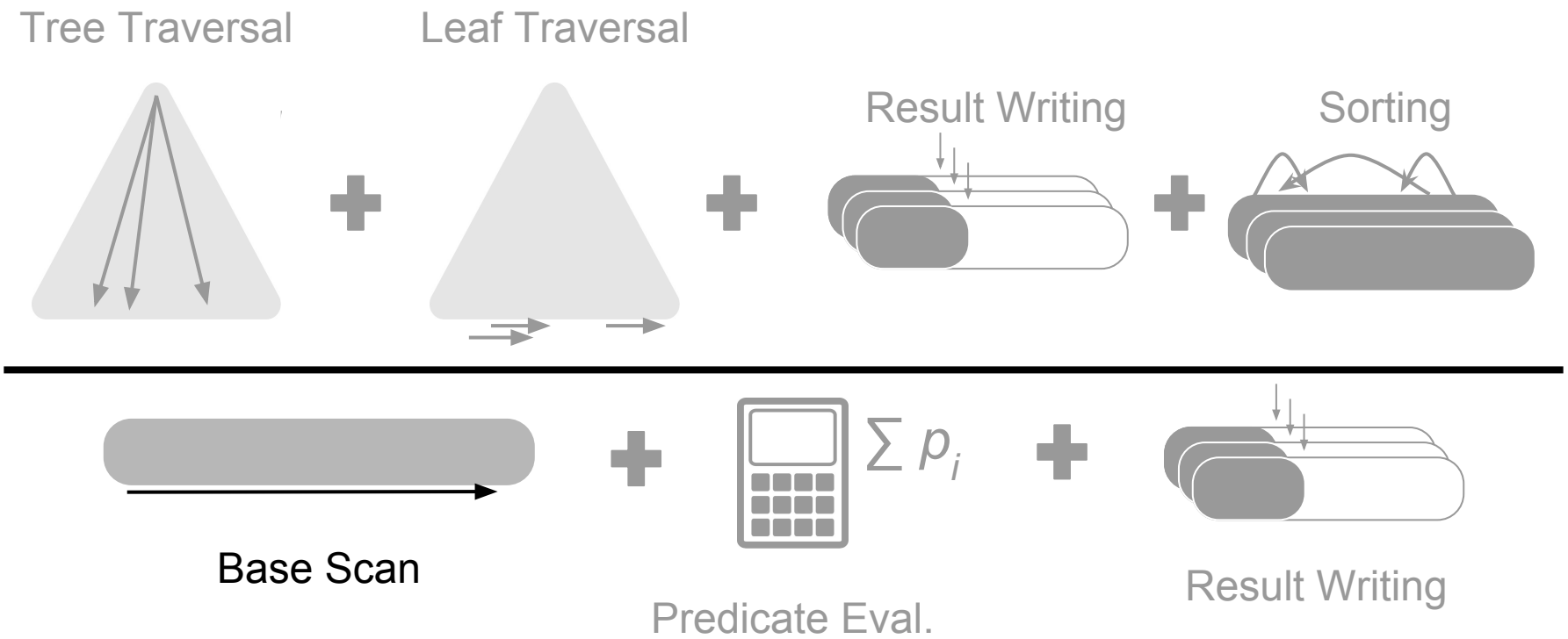
Multiquery APS =

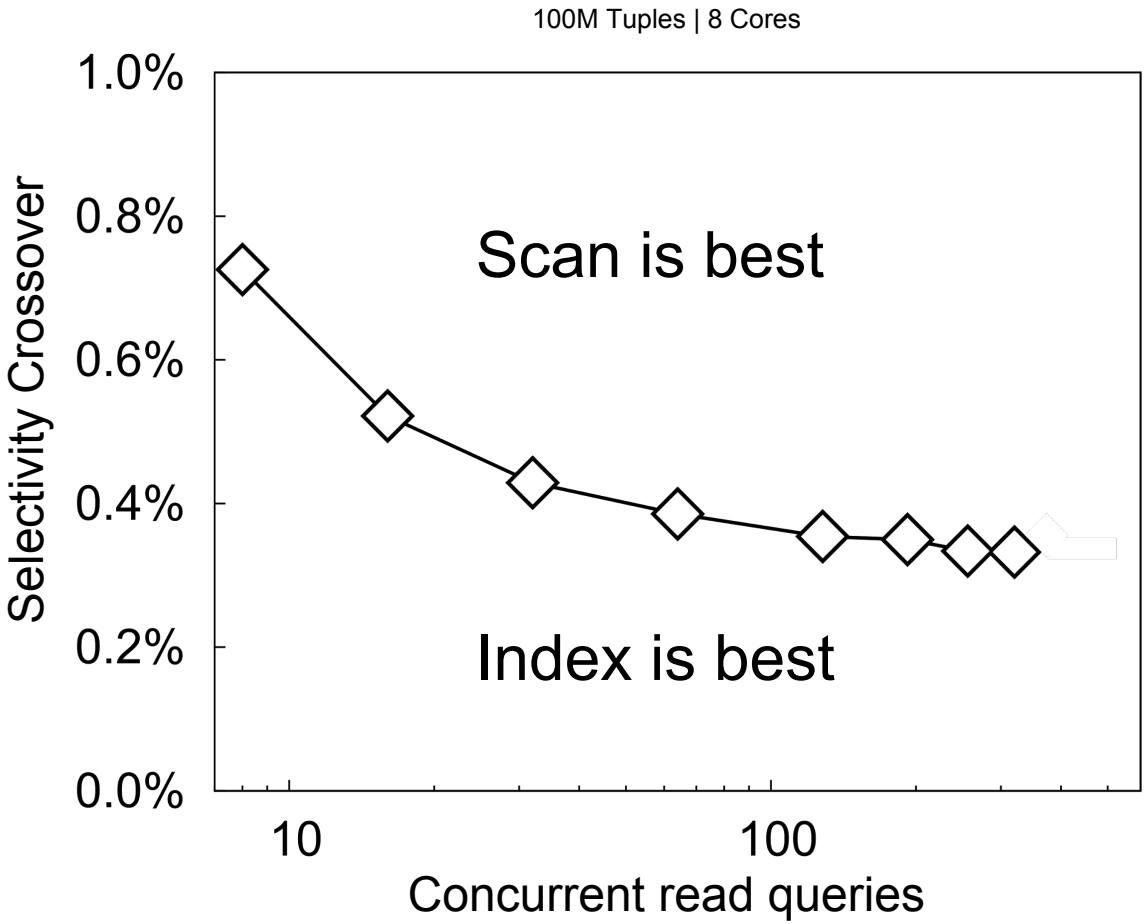



Tree Traversal
Leaf Traversal
Result Writing
Sorting
$\sum p_i$
Base Scan
Predicate Eval.
Result Writing

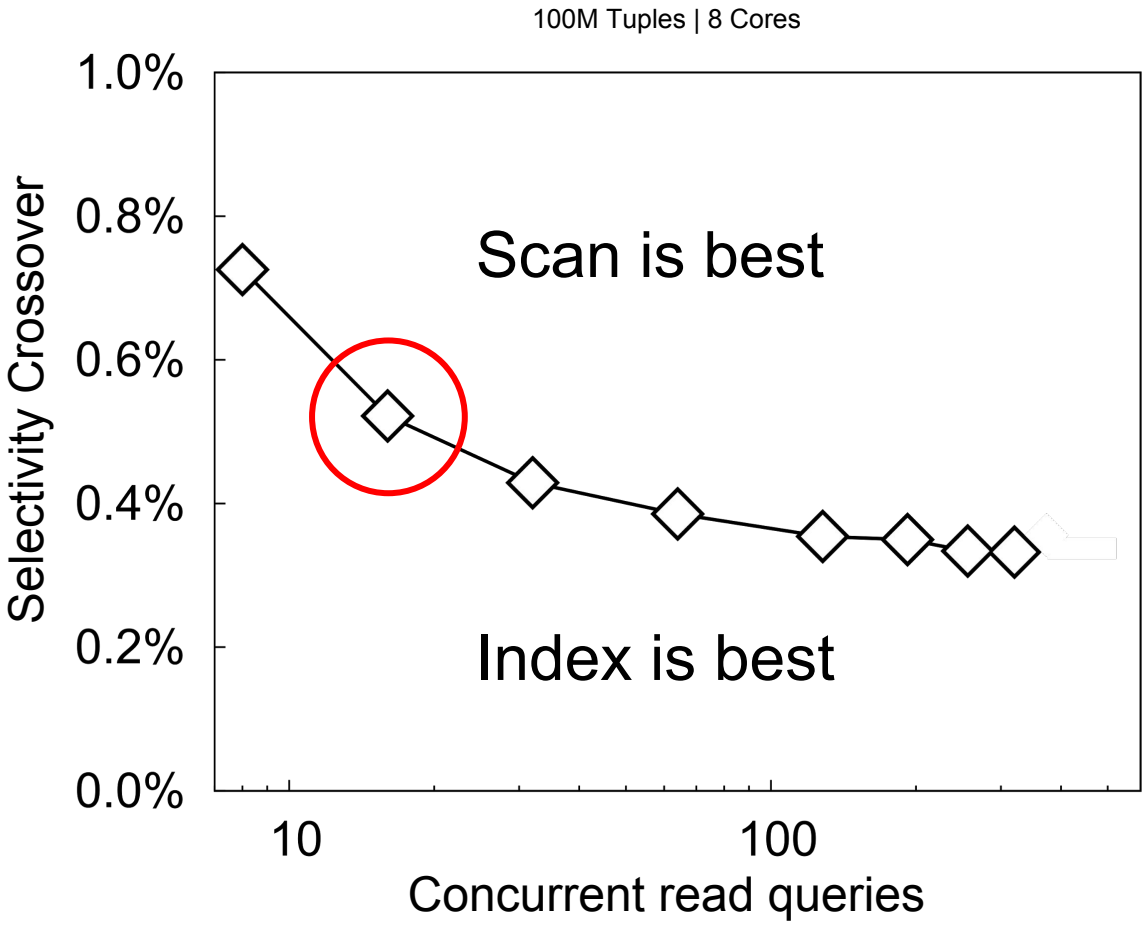

Tree Traversal
Leaf Traversal
Result Writing
Sorting
$\sum p_i$
Base Scan
Predicate Eval.
Result Writing


Tree Traversal
Leaf Traversal
Result Writing
Sorting
Base Scan
$\sum p_i$
Predicate Eval.
Result Writing

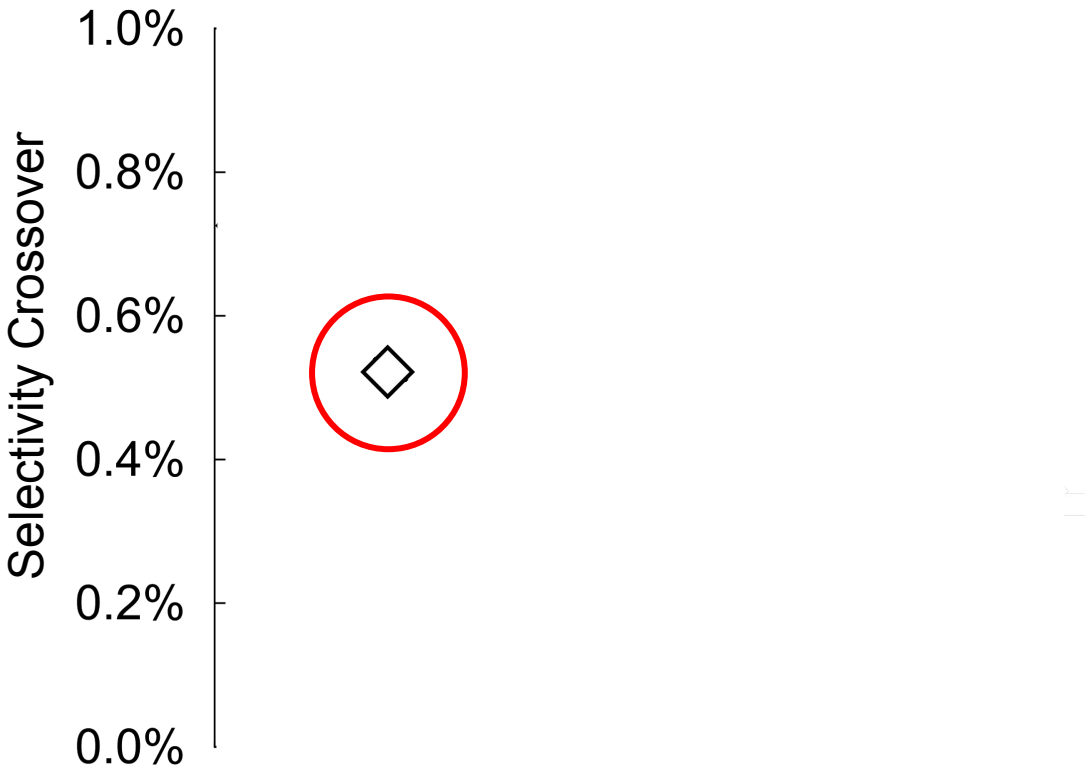

Tree Traversal
Leaf Traversal
Result Writing
Sorting
$\sum p_i$
Base Scan
Predicate Eval.
Result Writing

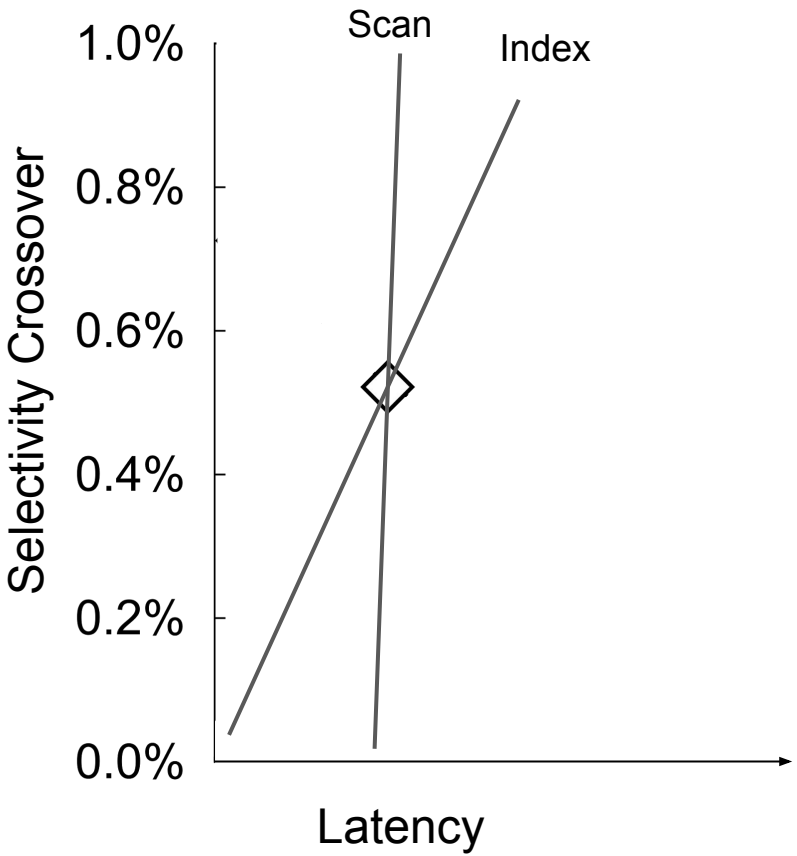



DASlab @ Harvard SEAS

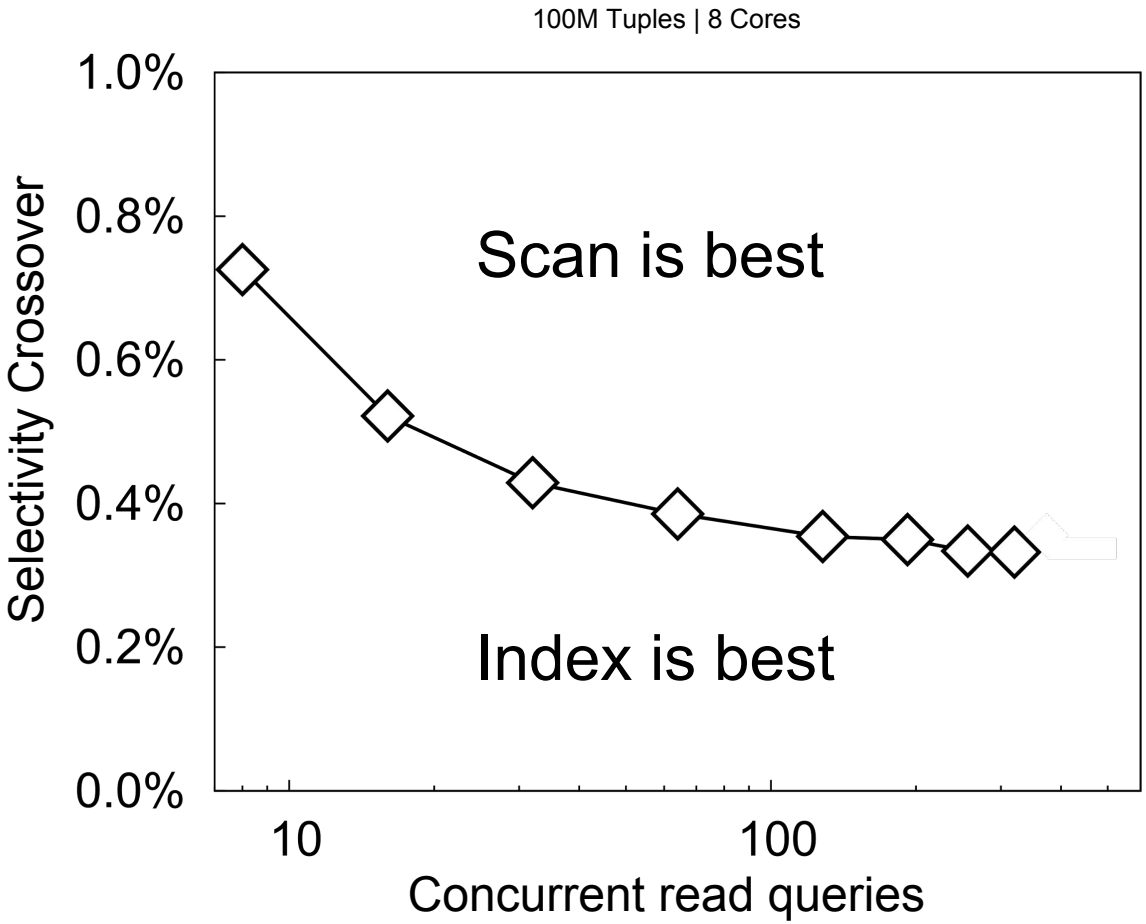



DASlab @ Harvard SEAS


Selectivity Crossover
1.0%
0.8%
0.6%
0.4%
0.2%
0.0%

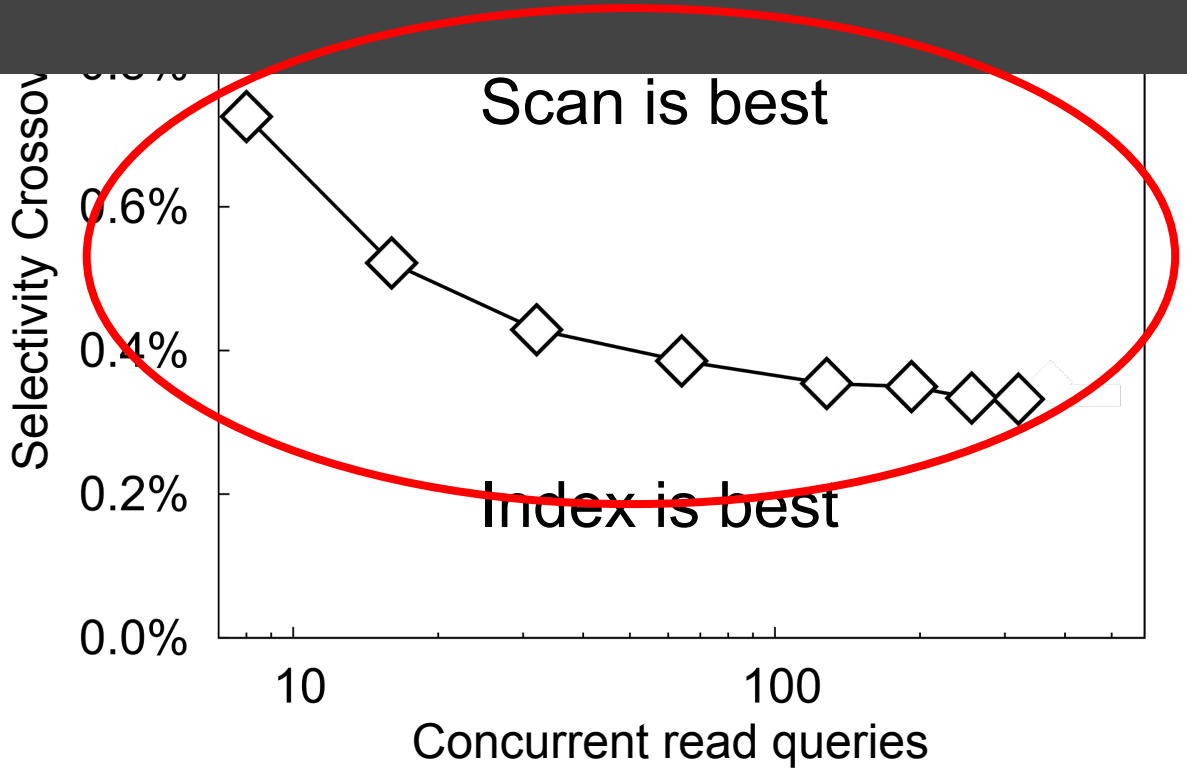



DASlab @ Harvard SEAS

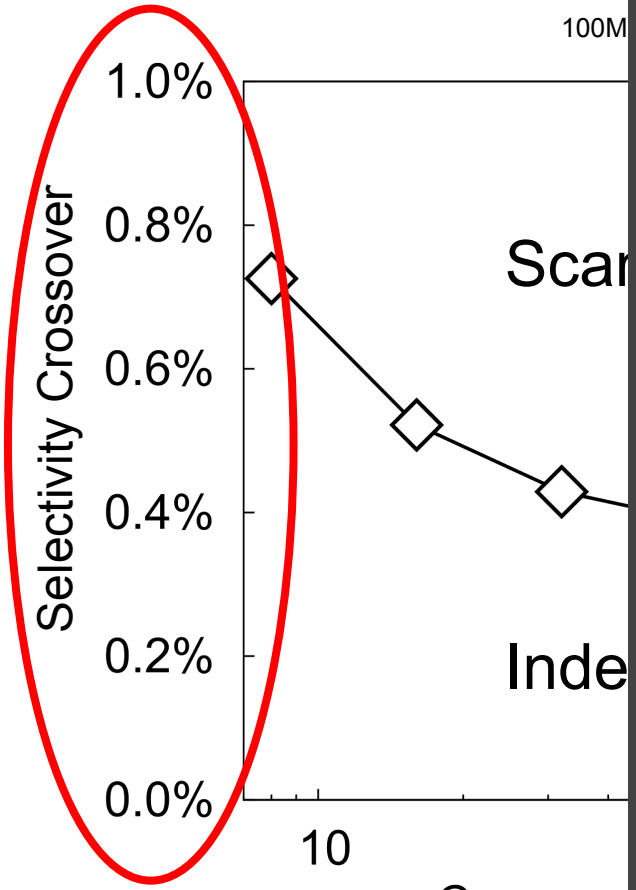



DASlab @ Harvard SEAS

# Concurrency matters

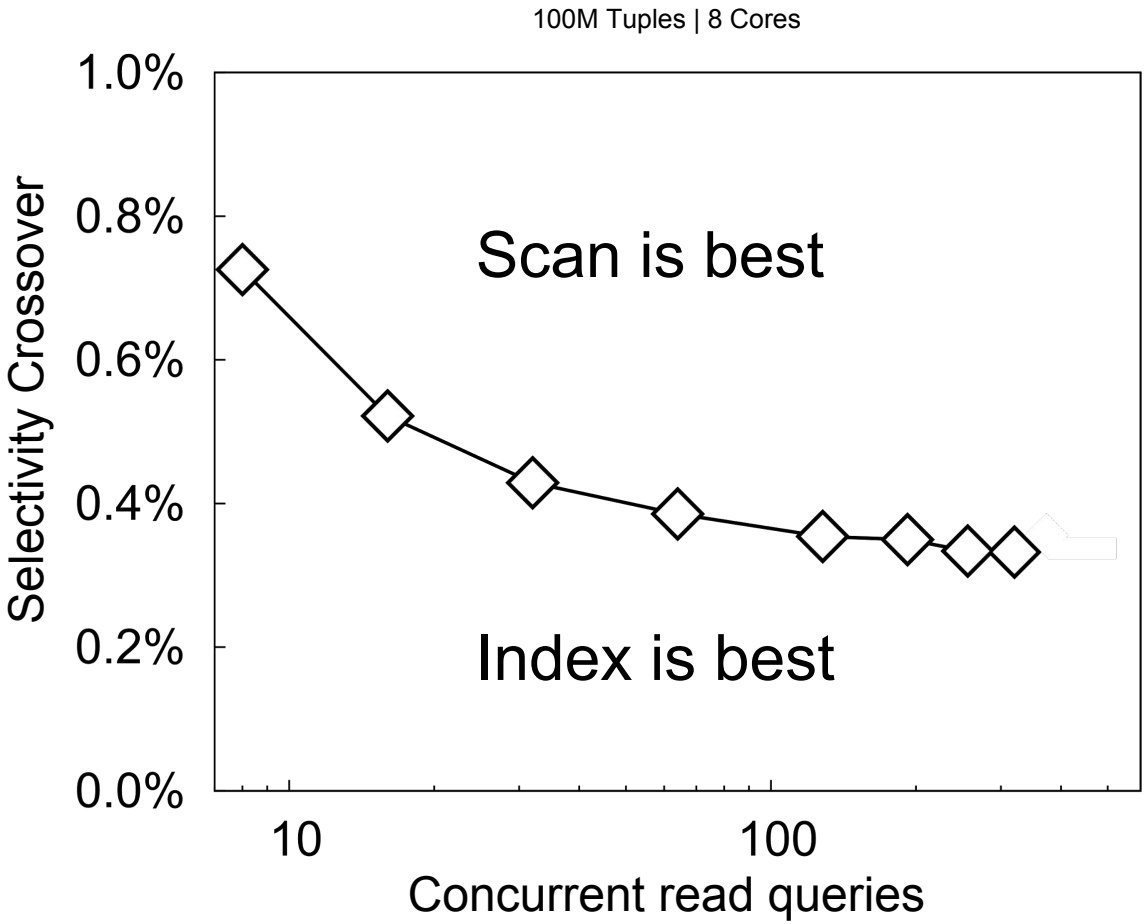



DASlab @ Harvard SEAS



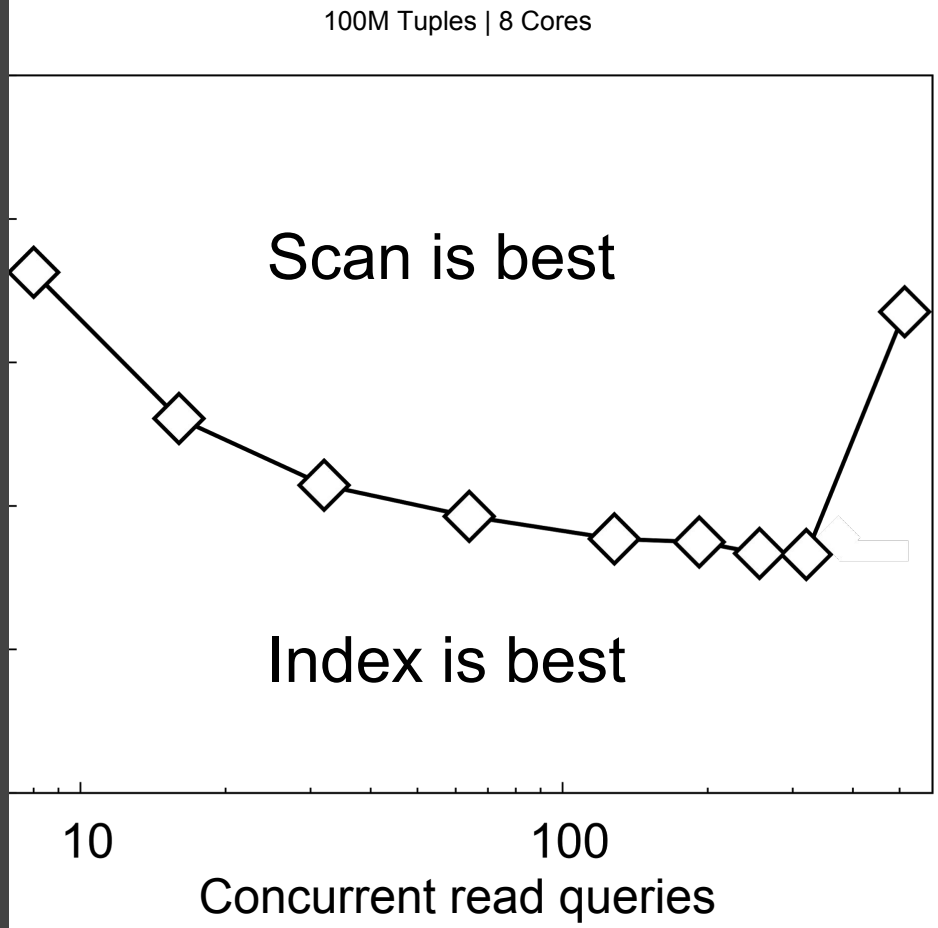

# Large result count

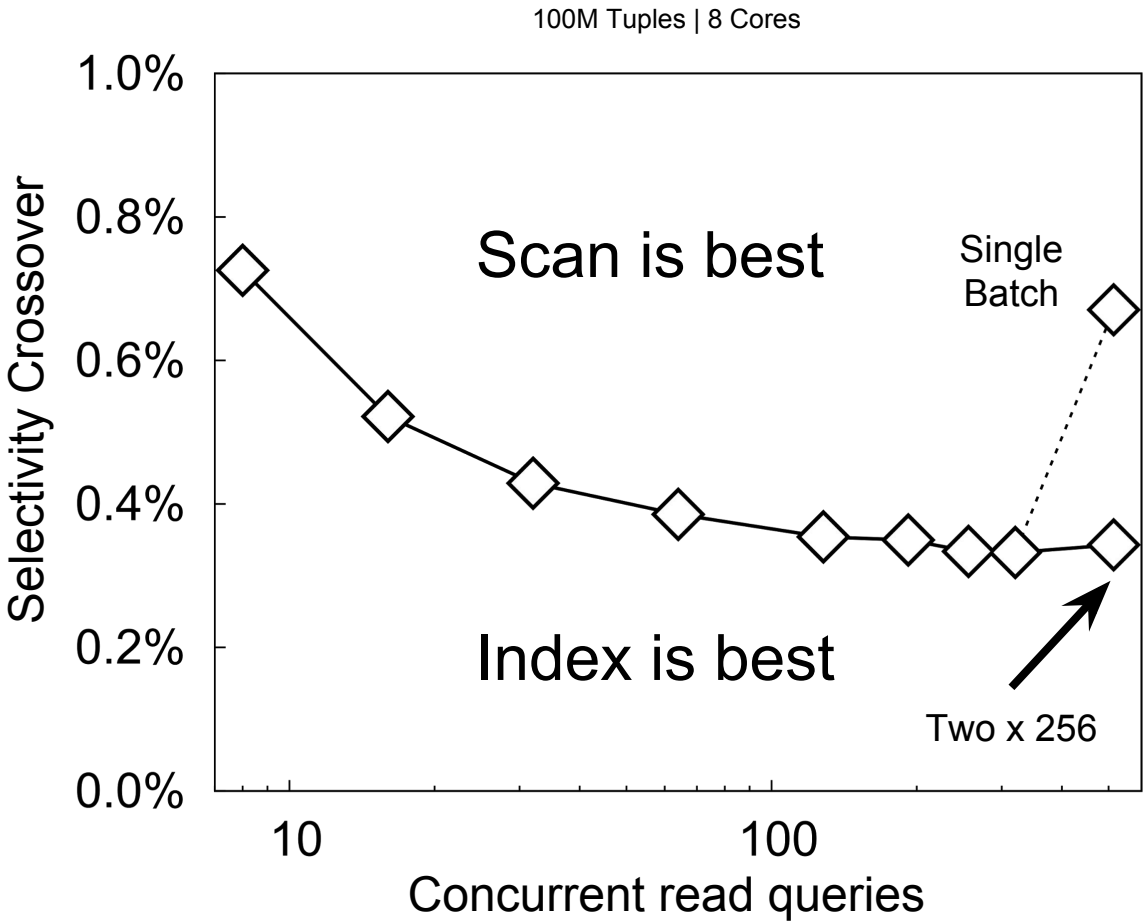
DASlab @ Harvard SEAS




DASlab @ Harvard SEAS

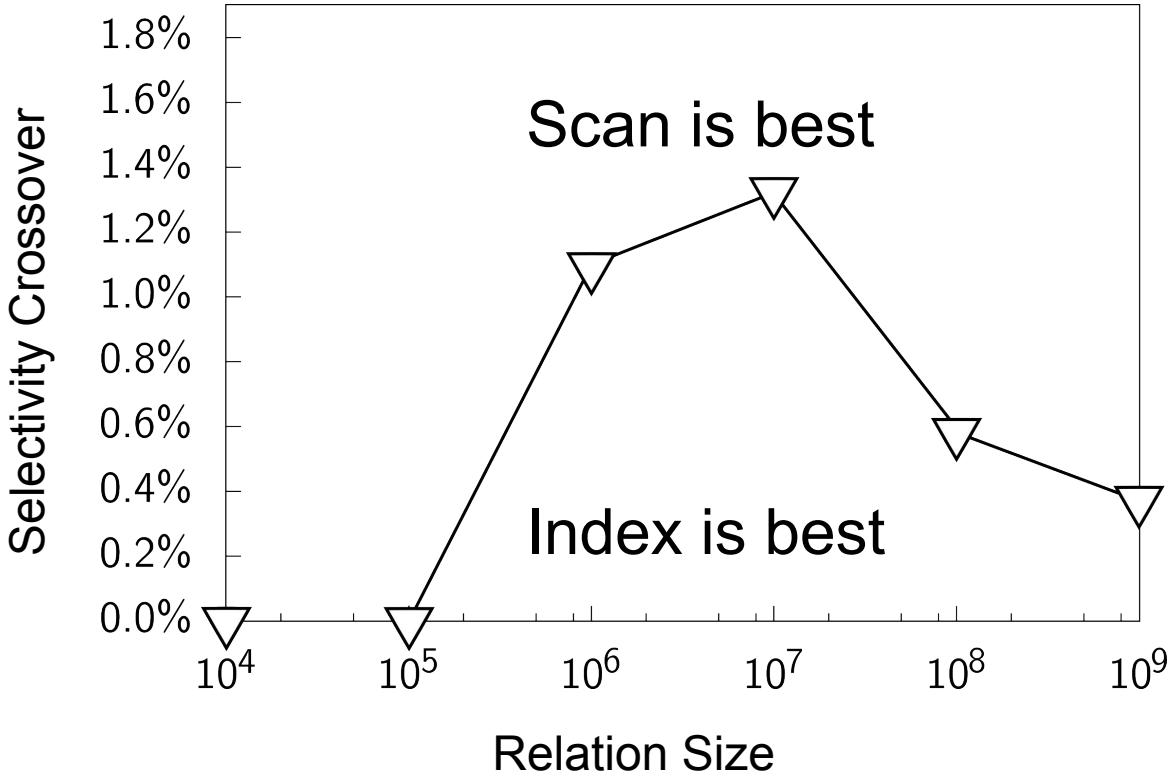
# Shared scans help… to a point





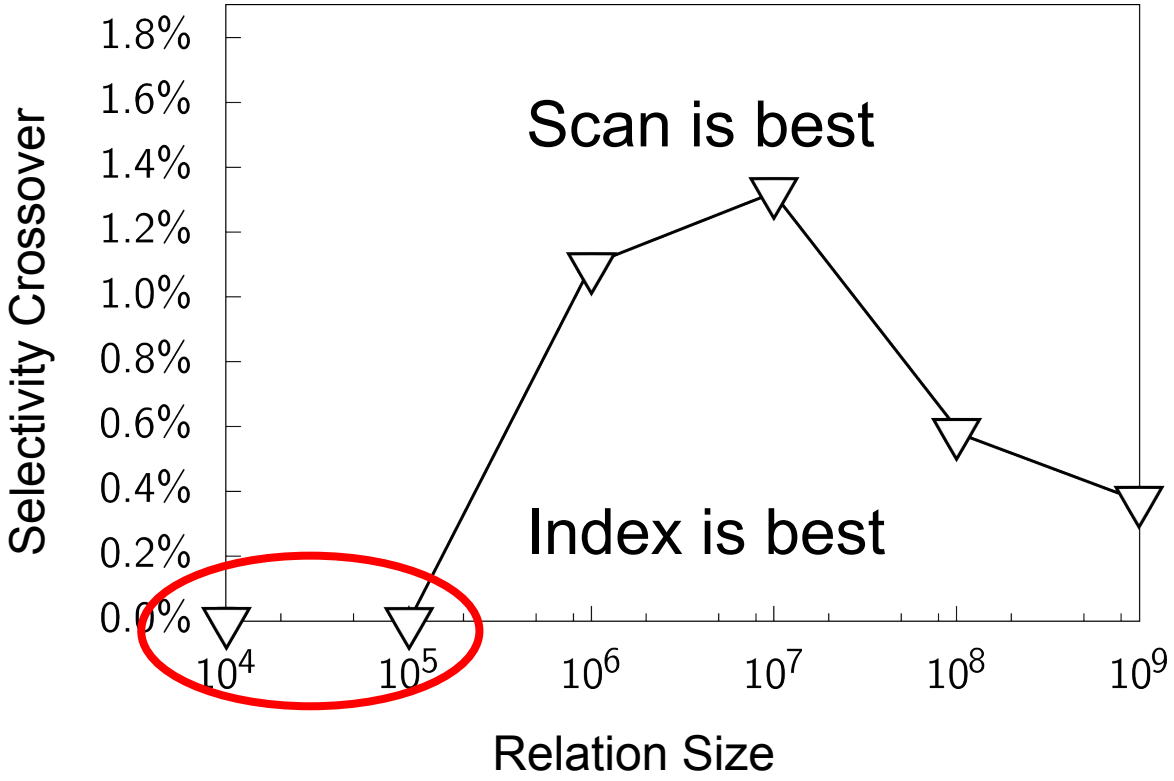

DASlab @ Harvard SEAS

8 Queries | 8 Cores | Mean of 10 trials




DASlab @ Harvard SEAS

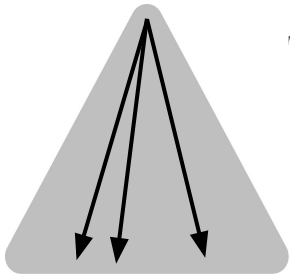
8 Queries | 8 Cores | Mean of 10 trials

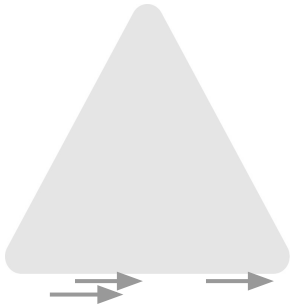



DASlab @ Harvard SEAS

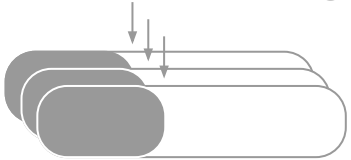
Tree Traversal

Leaf Traversal

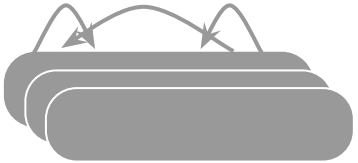




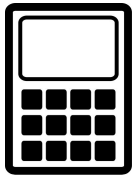
Result Writing



Sorting

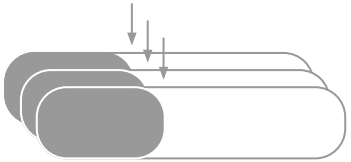


Base Scan



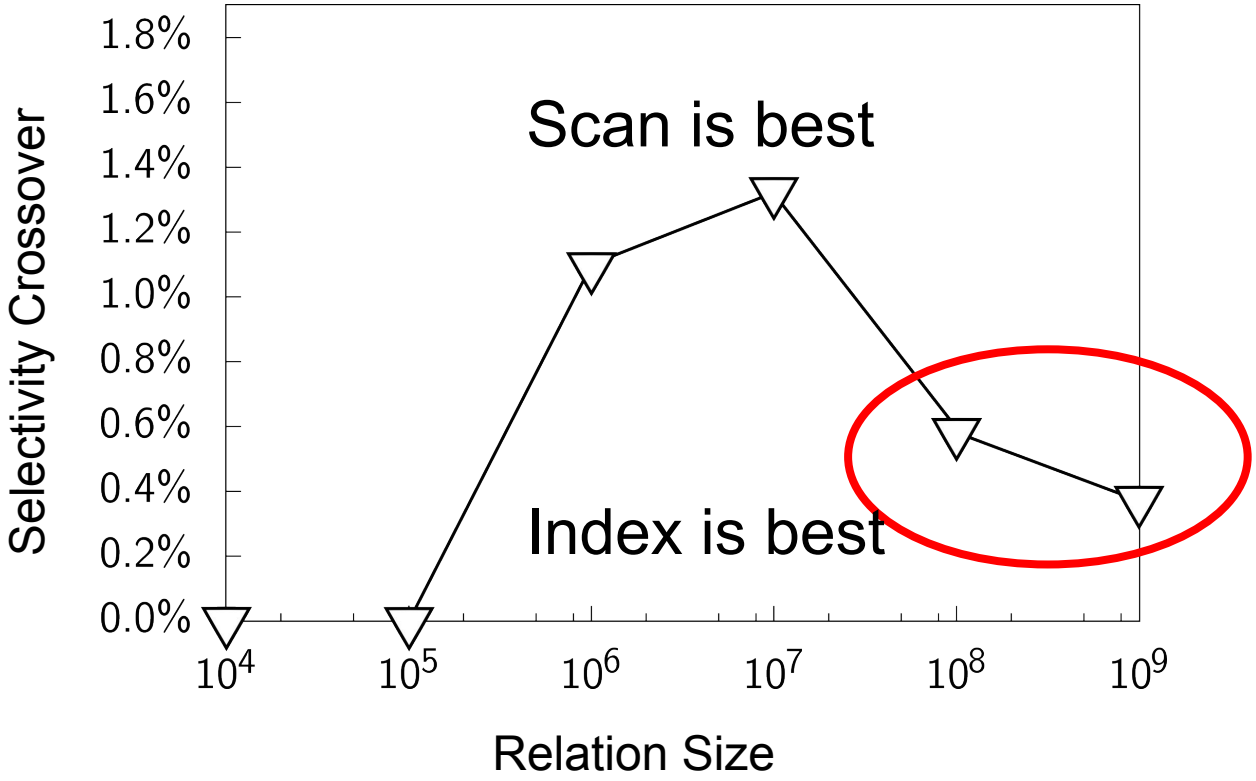
$\sum p_i$

Predicate Eval.

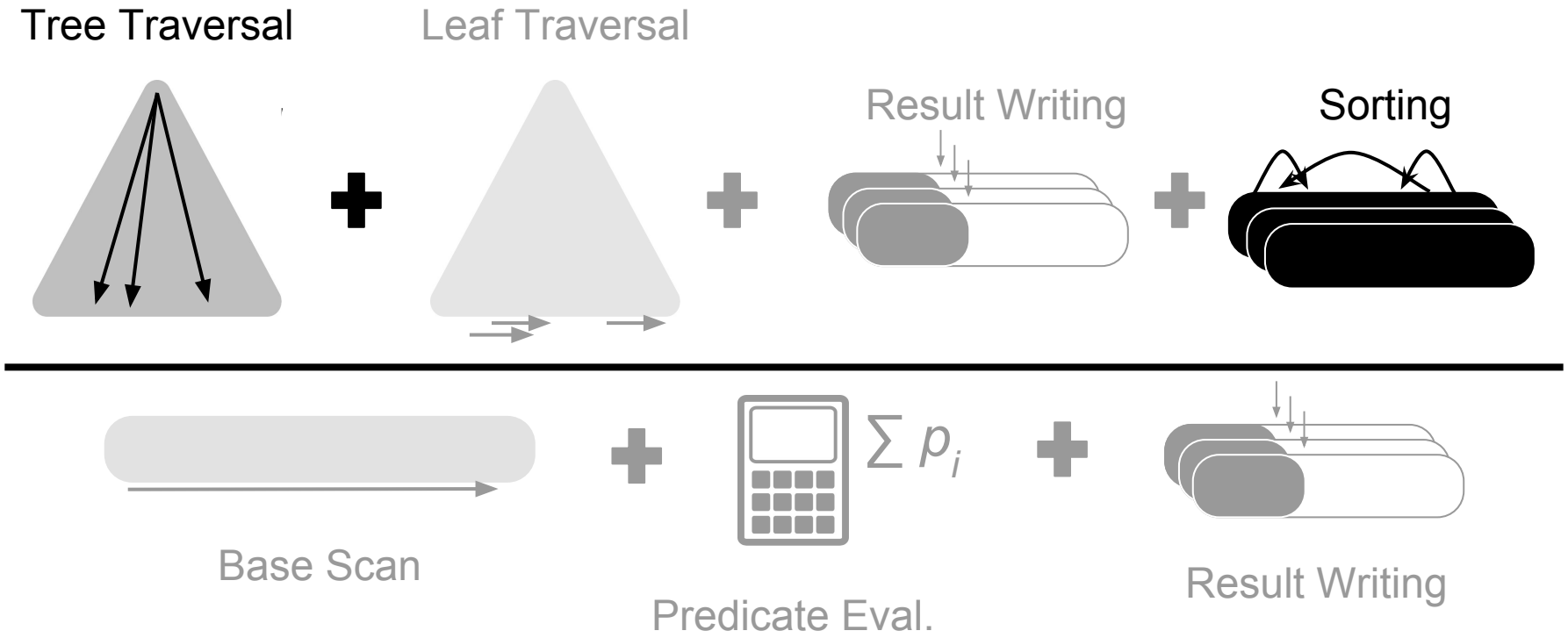


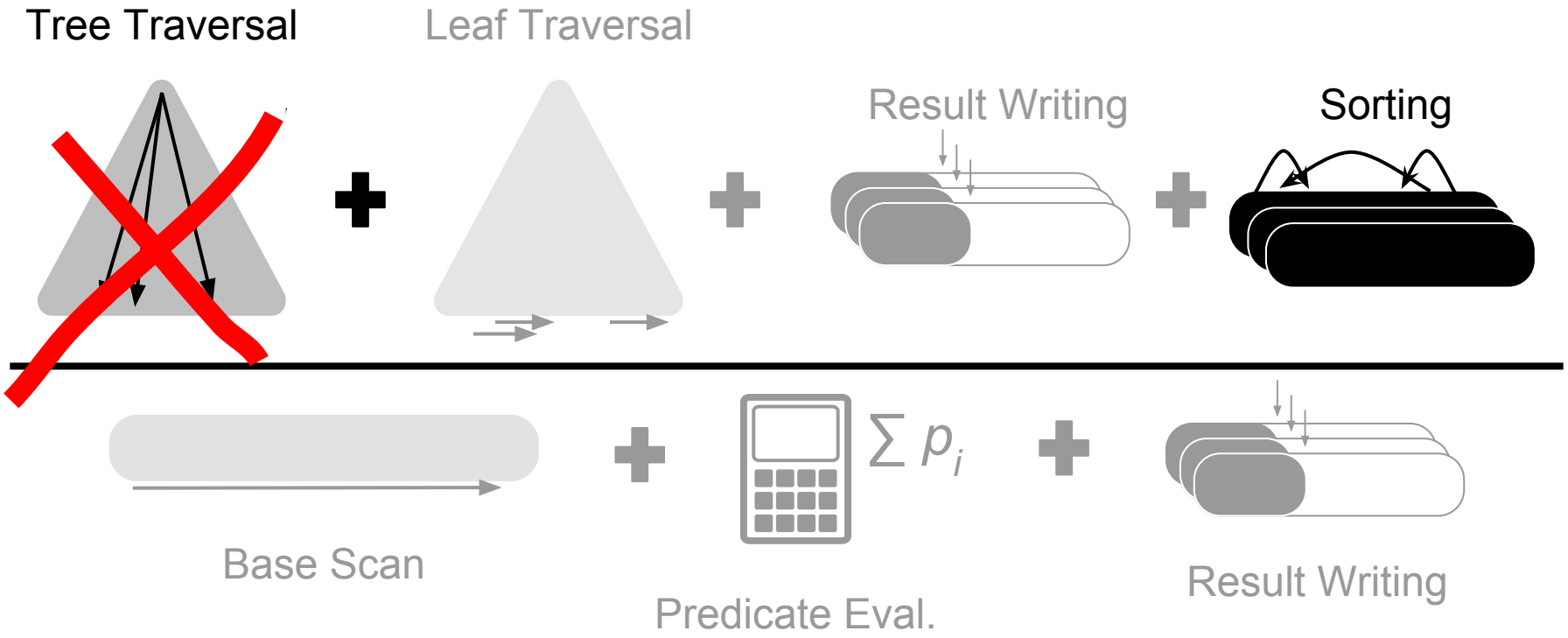
Result Writing

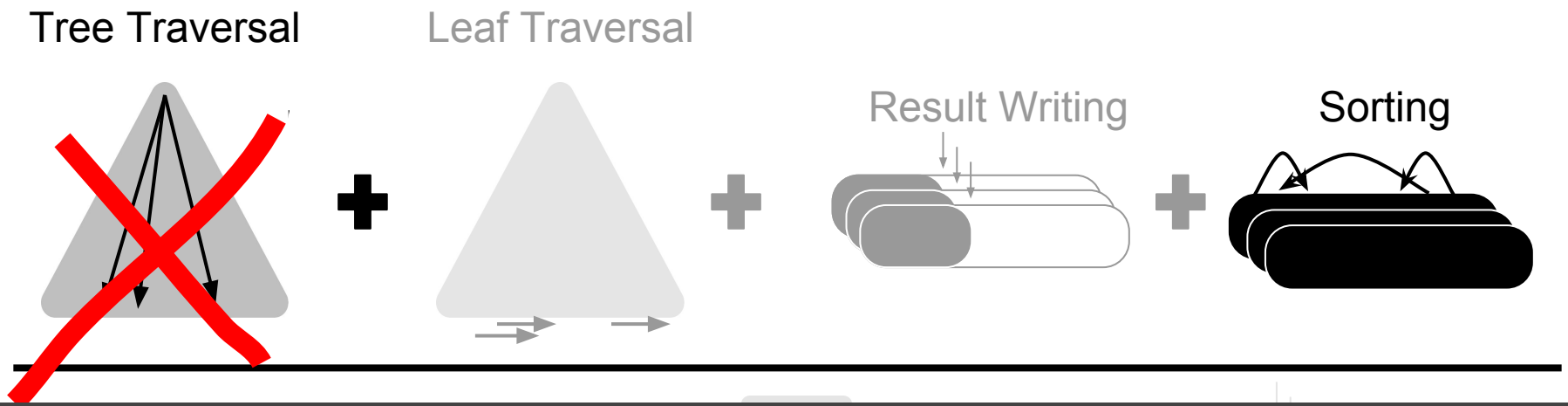
8 Queries | 8 Cores | Mean of 10 trials



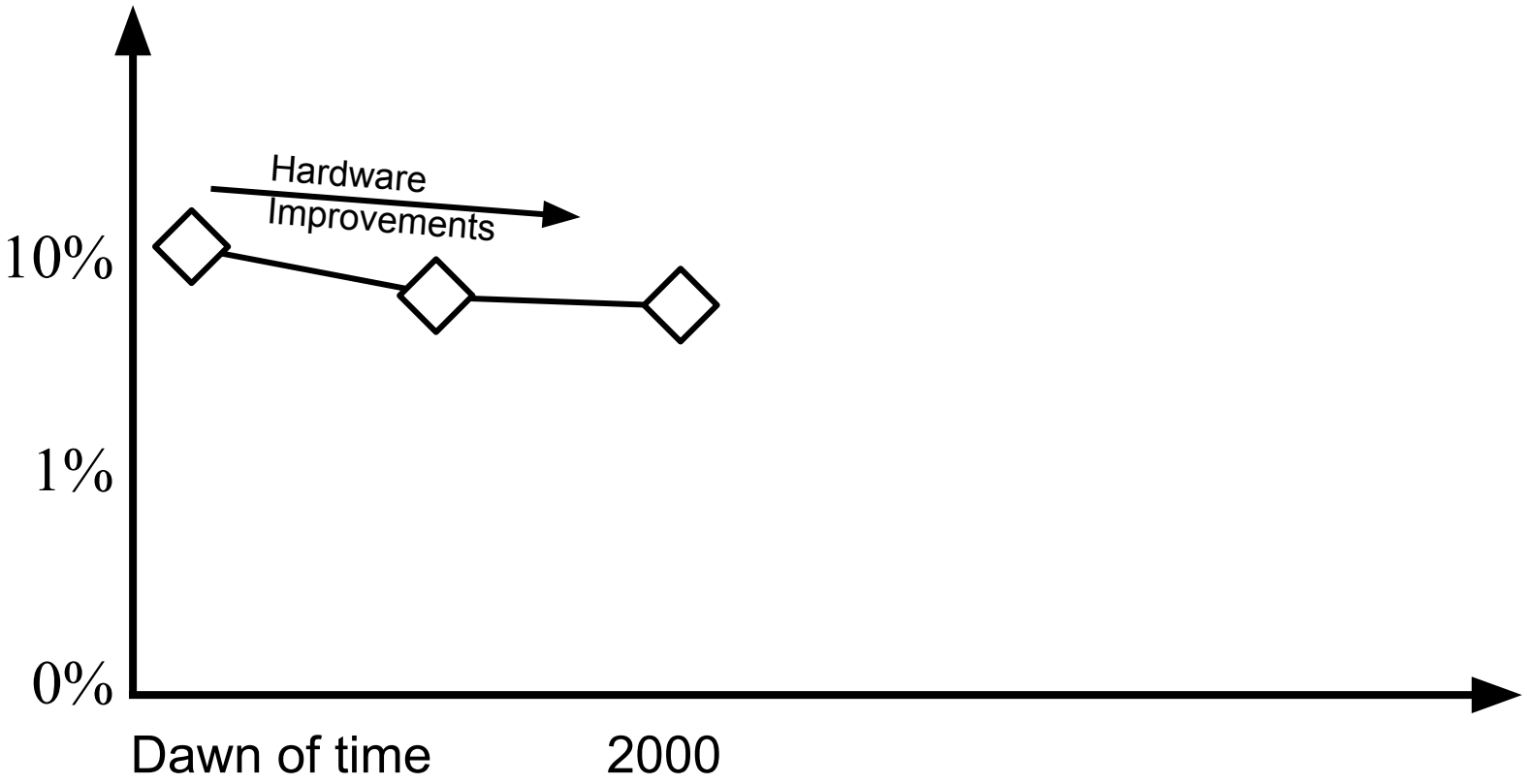

DASlab @ Harvard SEAS

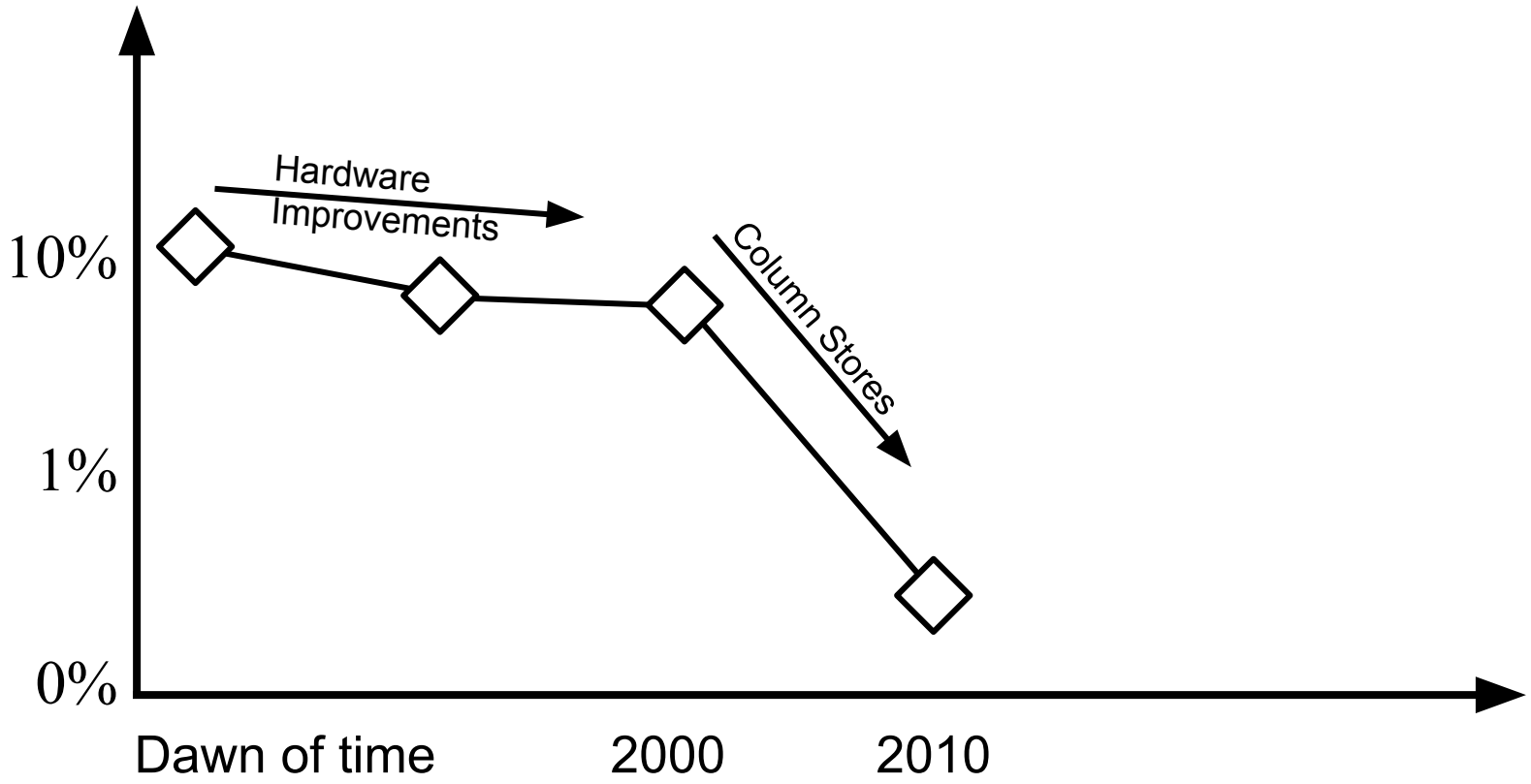

Tree Traversal
Leaf Traversal
Result Writing
Sorting
$\sum p_i$
Base Scan
Predicate Eval.
Result Writing

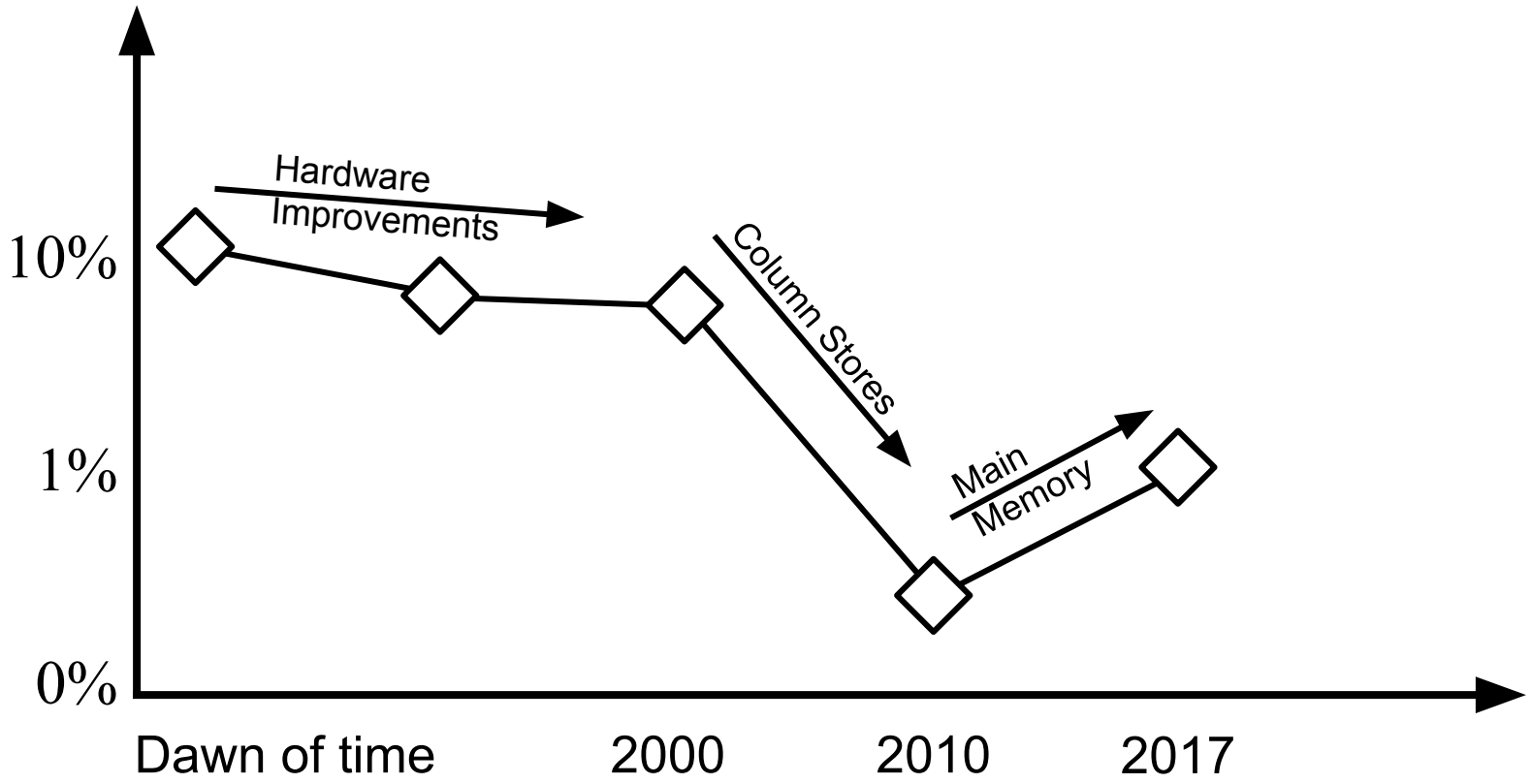

Tree Traversal
Leaf Traversal
Result Writing
Sorting
$\sum p_i$
Base Scan
Predicate Eval.
Result Writing




# Concurrency + data set size

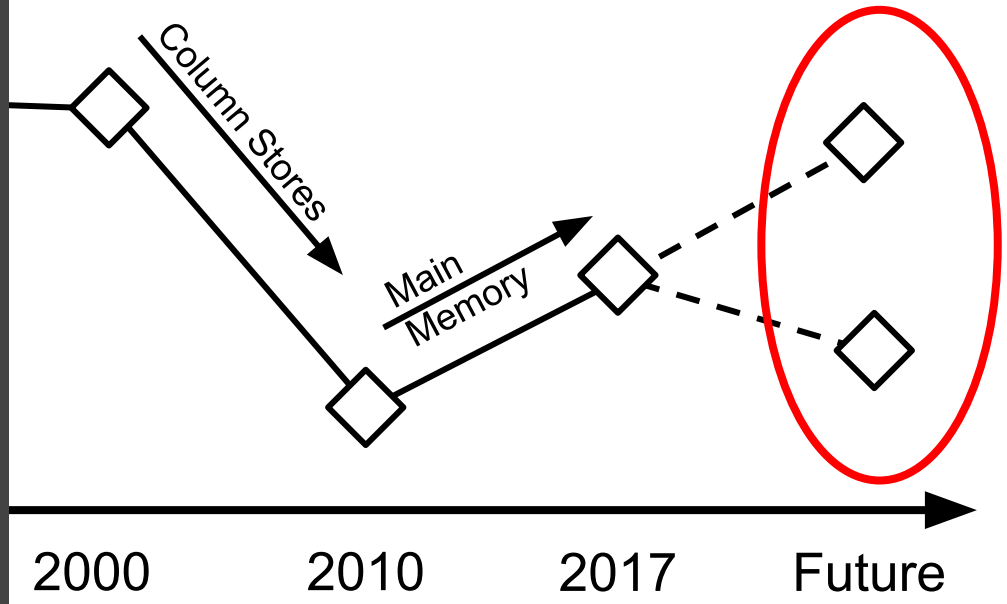

Hardware
Improvements
10%
1%
0%
Dawn of time
2000

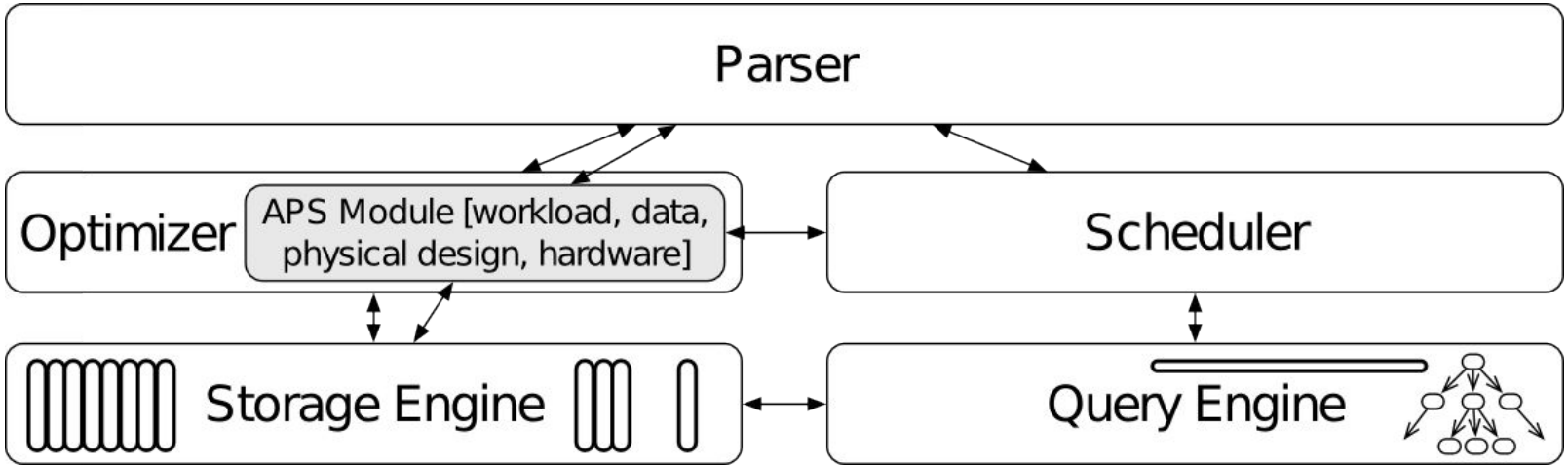

Hardware
Improvements
Column Stores
10%
1%
0%
Dawn of time
2000
2010



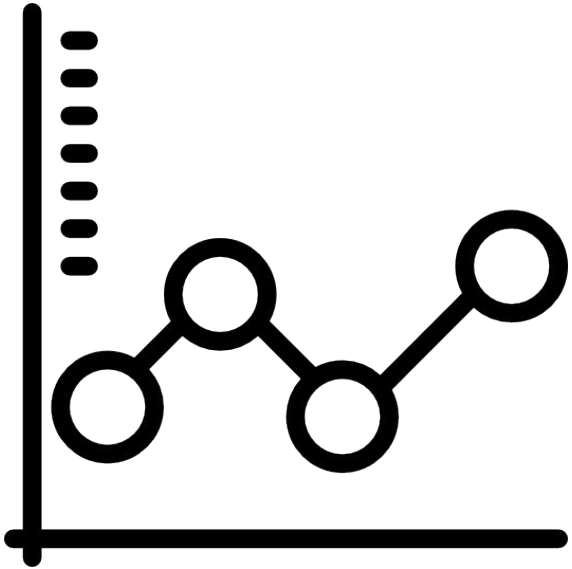

DASlab
@ Harvard SEAS

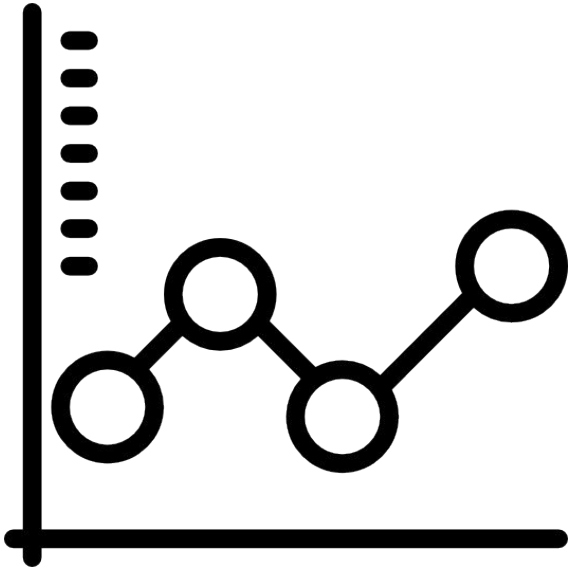
# “What-if” questions

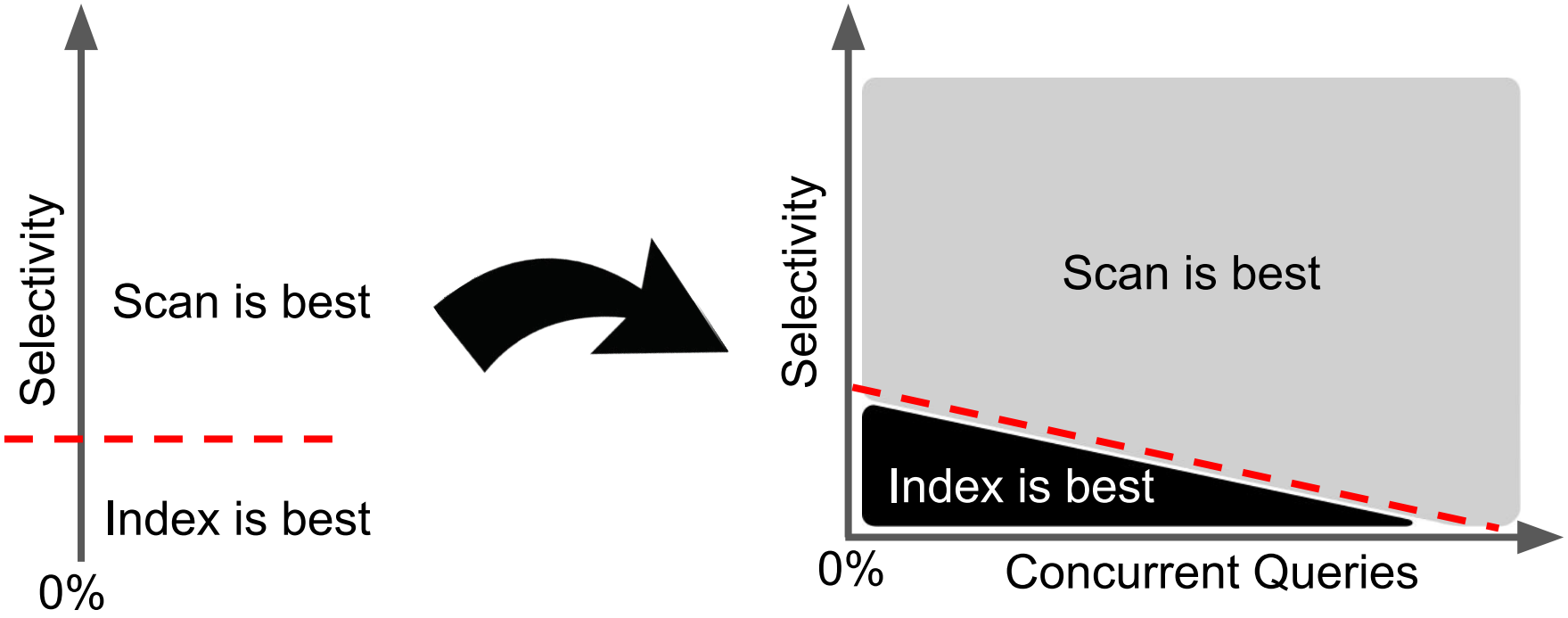


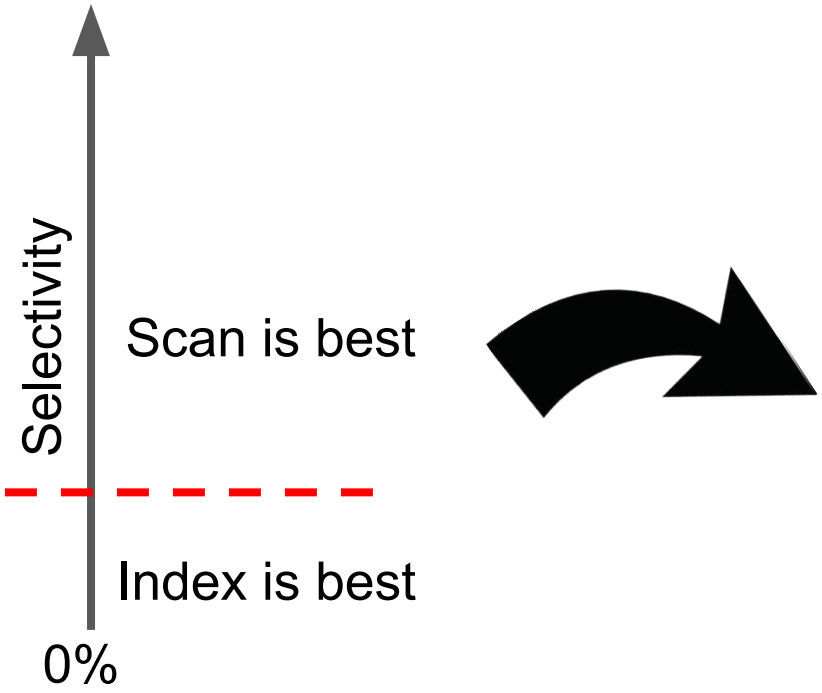

Parser
Optimizer
APS Module [workload, data, physical design, hardware]
Scheduler
Storage Engine
Query Engine

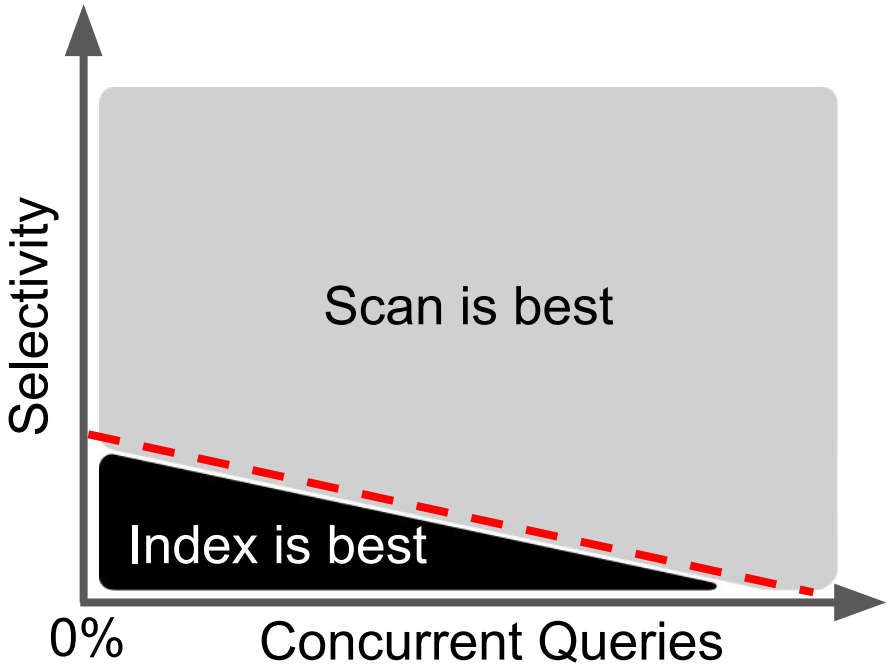
# Some problems you may be thinking.






Selectivity
Scan is best
Index is best
0%
Selectivity
Scan is best
Index is best
0%
Concurrent Queries




Thank you

daslab.seas.harvard.edu/evosys