

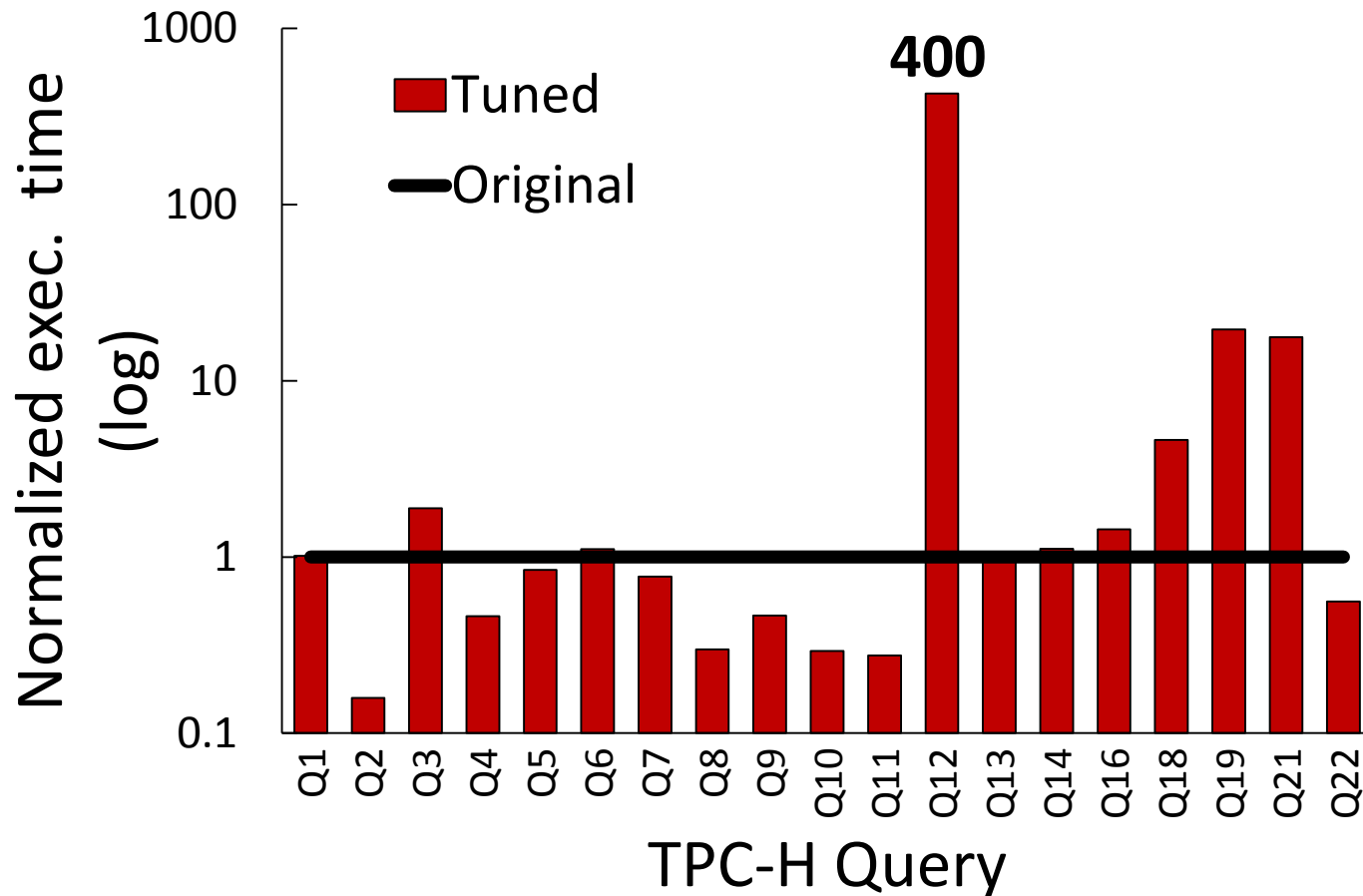
# Smooth Scan: Statistics-Oblivious Access Paths

Renata Borovica-Gajic   Stratos Idreos   Anastasia Ailamaki  
Marcin Zukowski   Campbell Fraser



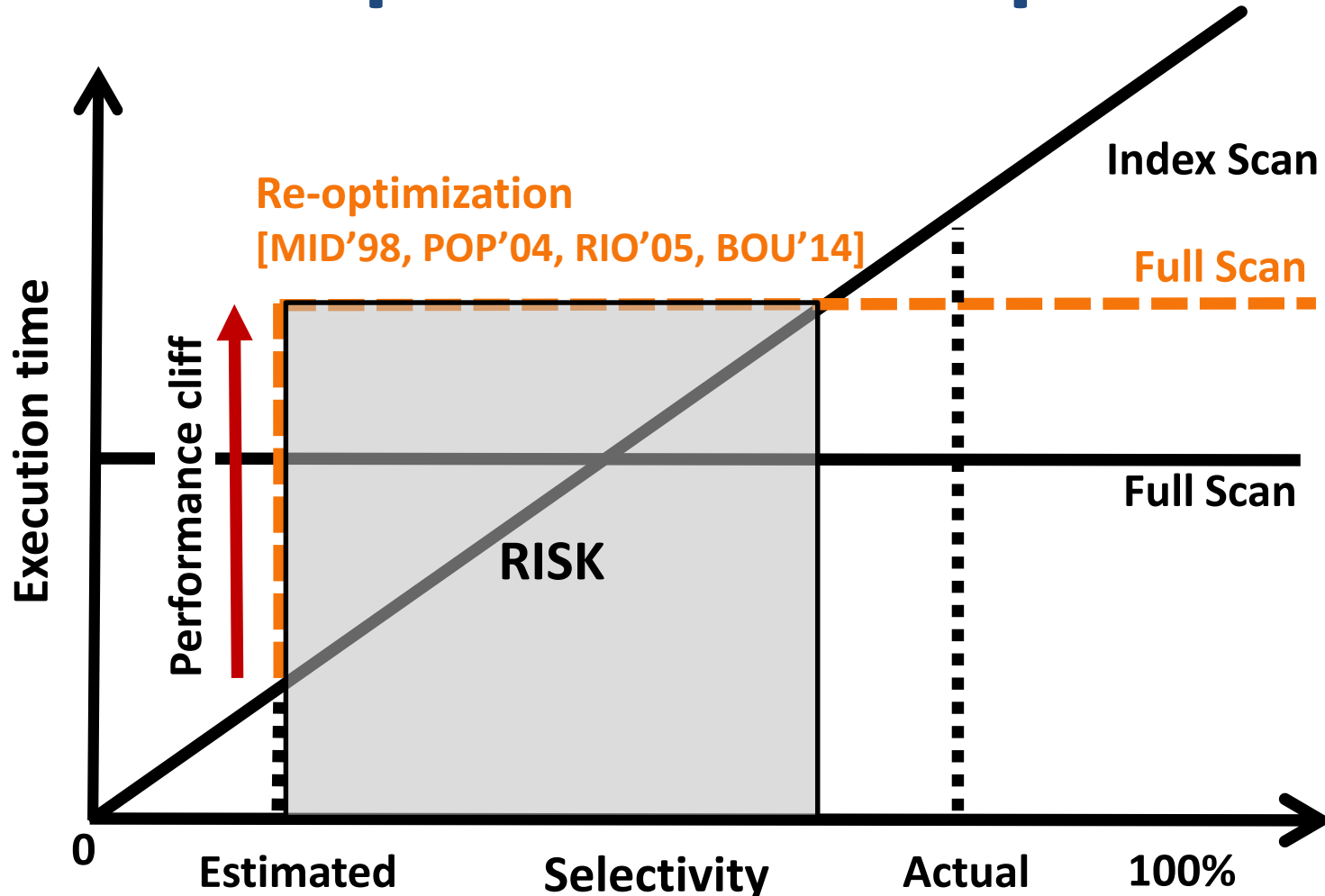
# Optimizers' sensitivity to statistics

Setting: TPC-H, SF10, DBMS-X, Tuning tool 5GB space



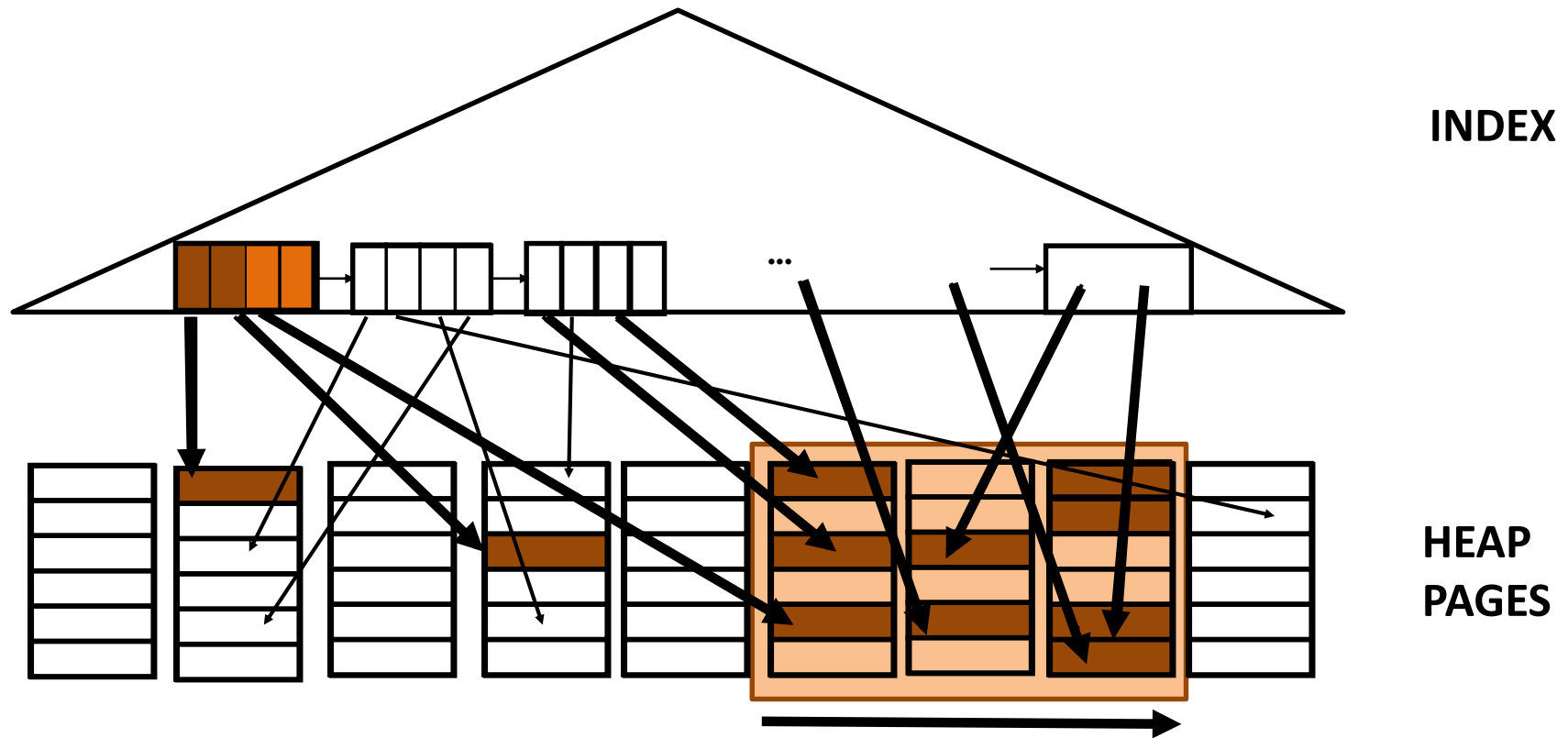
Degradation due to sub-optimal access paths

# Access path selection problem



**Statistics: unreliable advisor**  
**Re-optimization: risky**

# Access paths under looking glass



## Index Access

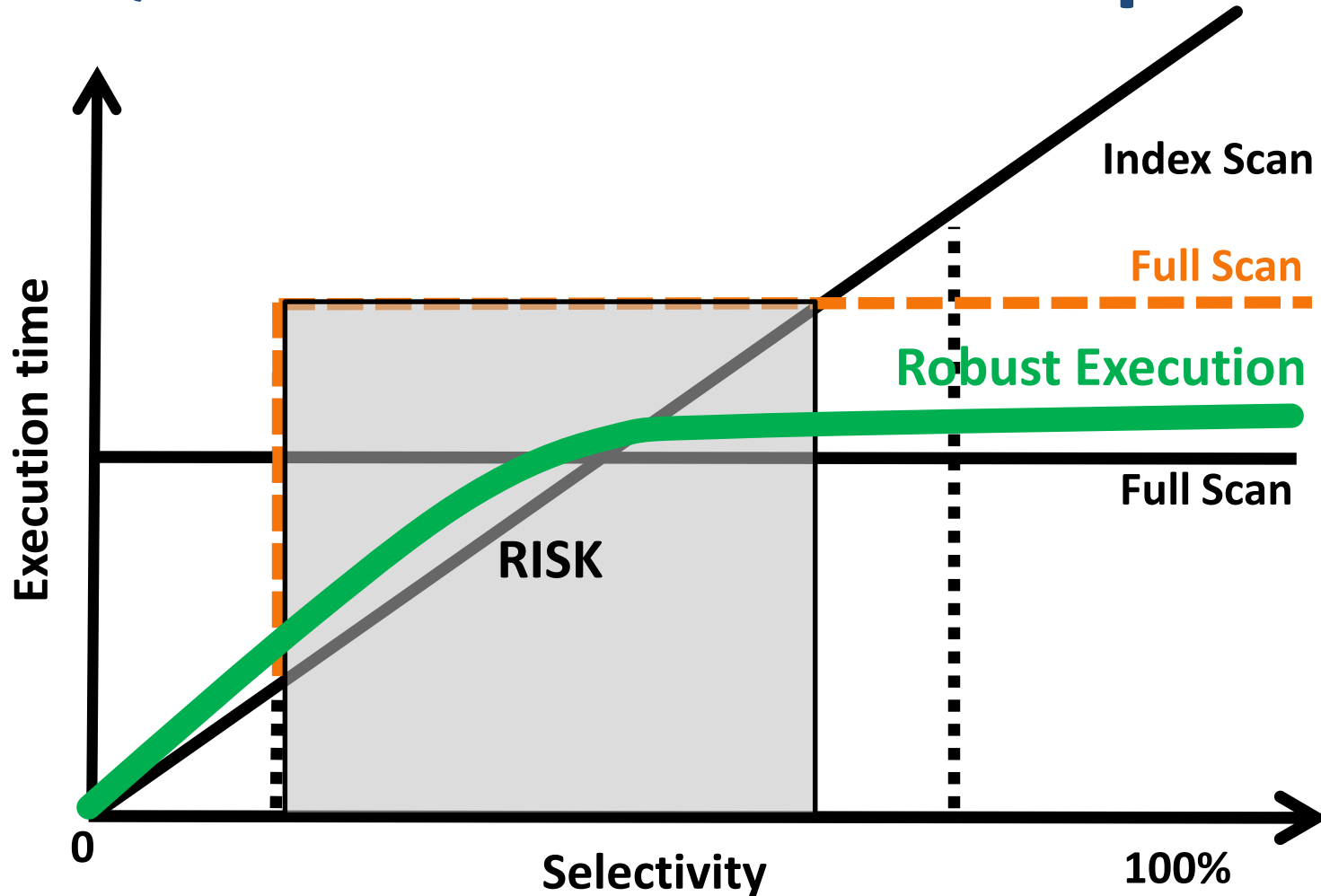
- + read what you need
- random (& repeated) I/O

## Sequential Access

- + (fast) sequential I/O
- read everything

**No single path is optimal**

# Quest for robust access paths



Near-optimal throughout entire selectivity range

# Smooth Scan in a nutshell

- Statistics-oblivious access path
- Learn result distribution at run-time
- **Adapt** as you go

## DESIGN GOALS

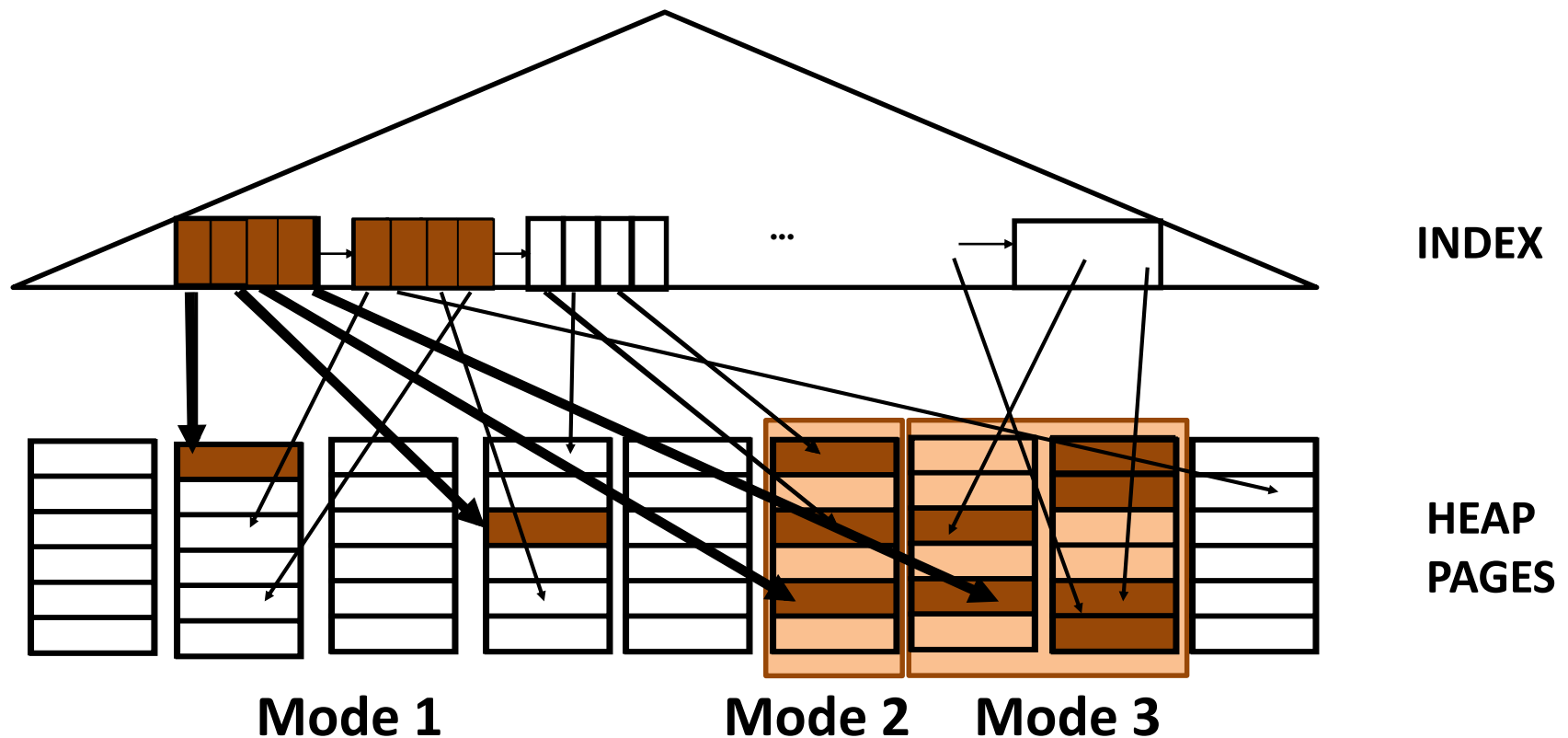
- Avoid **performance cliffs & risk**
- Continuous, gradual and **smooth adaptation**

# Adaptivity with Smooth Scan

**Morph** between Index and Sequential Scan

# Morphing mechanism

- Modes:
  1. **Index Access:** Traditional index access
  2. **Entire Page Probe:** Index access probes entire page
  3. **Gradual Flattening Access:** Probe adjacent region(s)





# Morphing policies

- Policies:
  - Greedy
  - Selectivity Increase Driven
  - Elastic

Selectivity increase -> Mode Increase

$$SEL_{region} > SEL_{global}$$

Selectivity decrease -> Mode Decrease

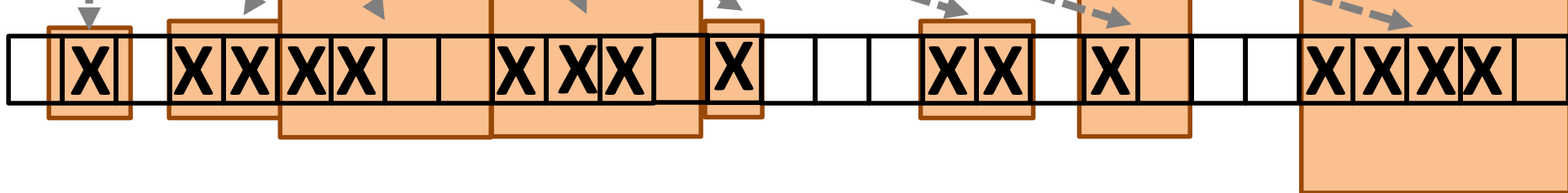
$$SEL_{region} < SEL_{global}$$

**INDEX**



















**X:** Page with result  
**SR:** Region selectivity  
**SG:** Global selectivity

**HEAP PAGES**



**Region snooping = Selectivity driven adaptation**

# Smooth Scan benefits

	Index Scan	Full Scan	Sort Scan	Smooth Scan
Avoid repeated accesses				
Fast sequential I/O				
Avoid full table read				
Tuples pipelining				

# Experimental setup

## Hardware:

2 Intel Xeon 6-core CPU @2.8 GHz, 48GB RAM

HDD: I/O transfer rate 120 MB/s, Random vs. Sequential ratio = 10

## Software:

PostgreSQL 9.2.1: Index Scan, Full Scan, Sort (Bitmap) Scan, Smooth Scan

## Workload:

TPC-H: SF 10

Micro-benchmark: 400M tuples, 10 columns random ( $1 - 10^5$ ), 25GB

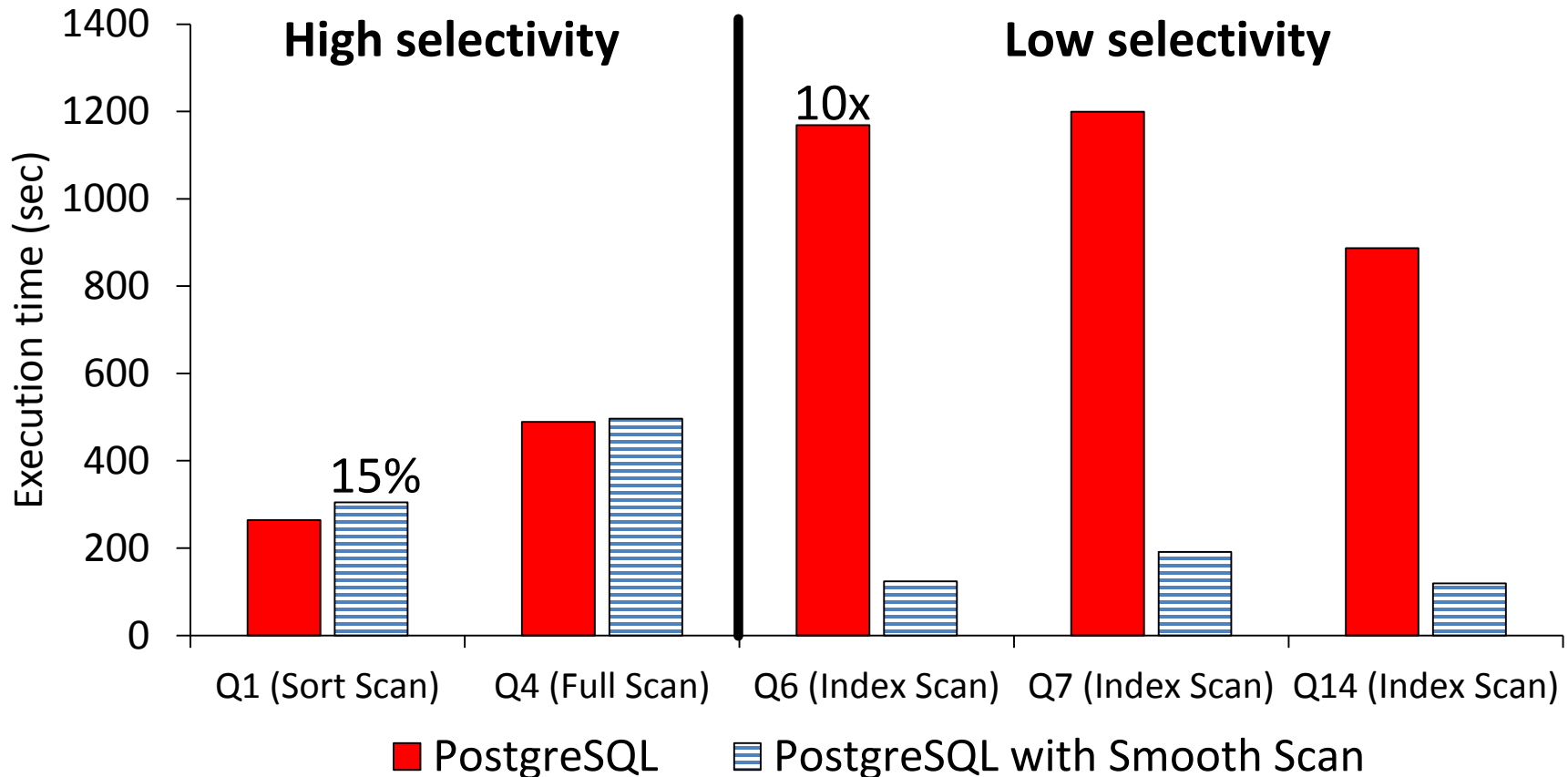
Q1: select \* from relation where  $c2 \geq 0$  and  $c2 < X\%$  [order by c2];

## Experimental Condition:

Cold file system cache

# TPC-H with Smooth Scan

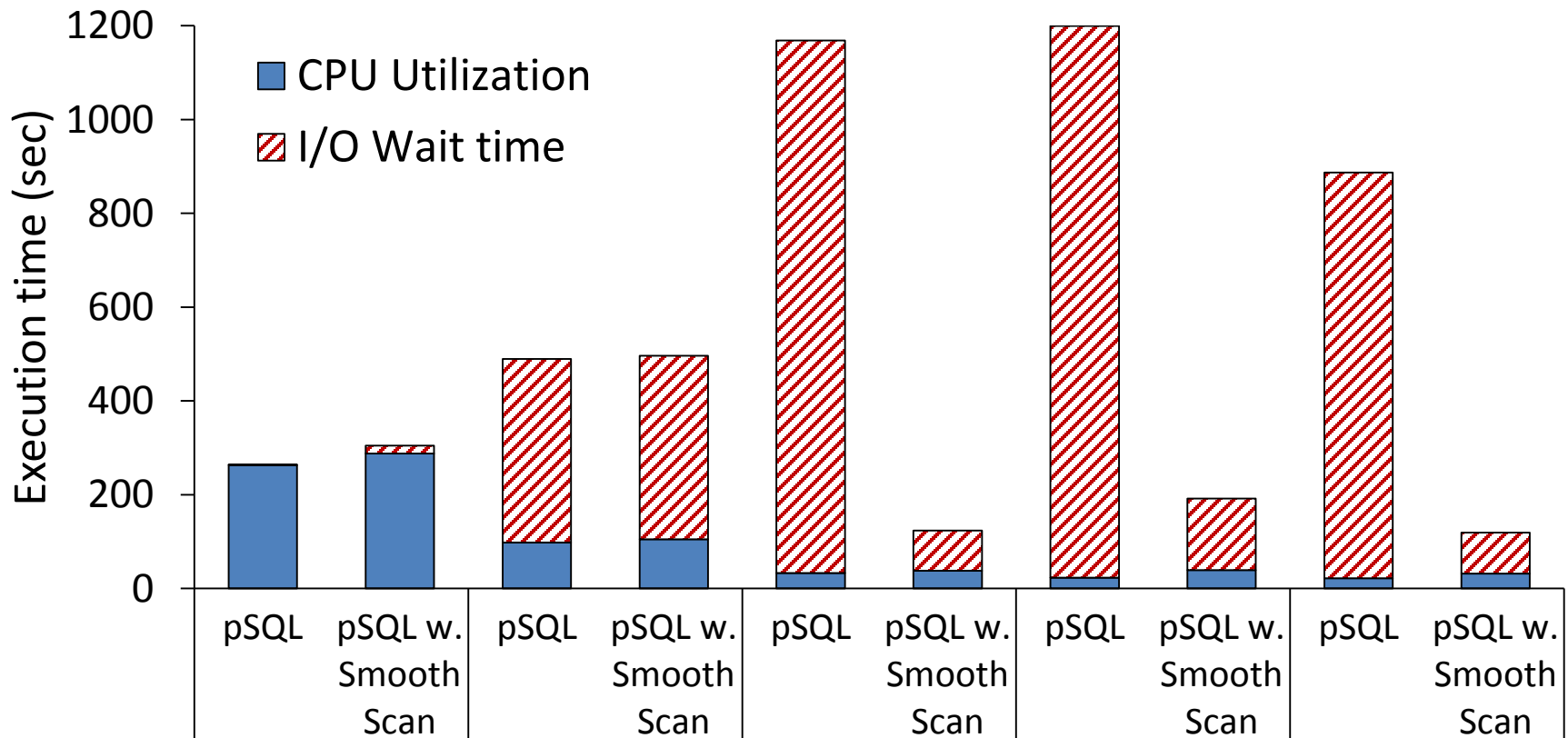
**Setting:** TPC-H, SF10, PostgreSQL with Smooth Scan



**Robust execution for all queries**

# TPC-H breakdown

	Q1		Q4		Q6		Q7		Q14	
	pSQL	Smooth S.	pSQL	Smooth S.	pSQL	Smooth S.	pSQL	Smooth S.	pSQL	Smooth S.
# I/O Requests (K)	70	77	224	235	566	95	745	124	416	87

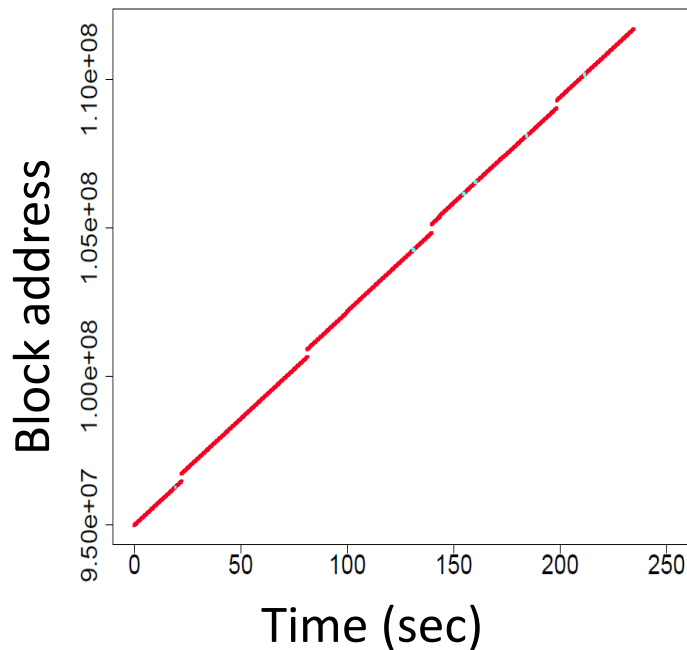


**Smooth Scan significantly decreases I/O wait time**

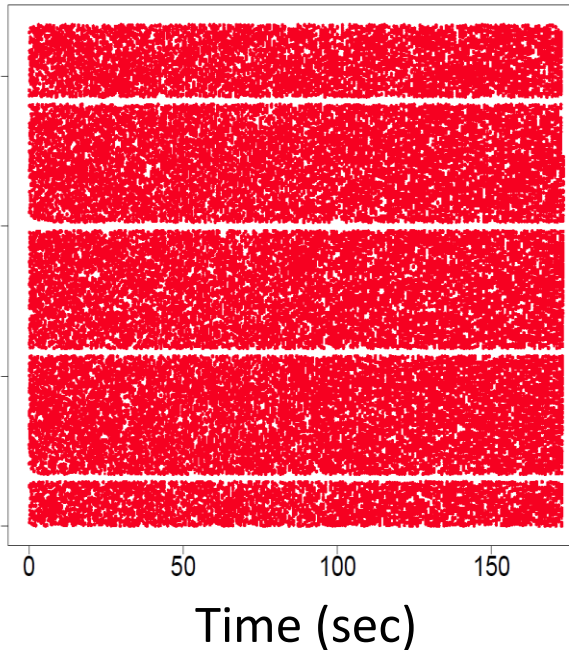
# Snooping I/O access

Setting: TPC-H, Q1, Lineitem table, iosnoop tool

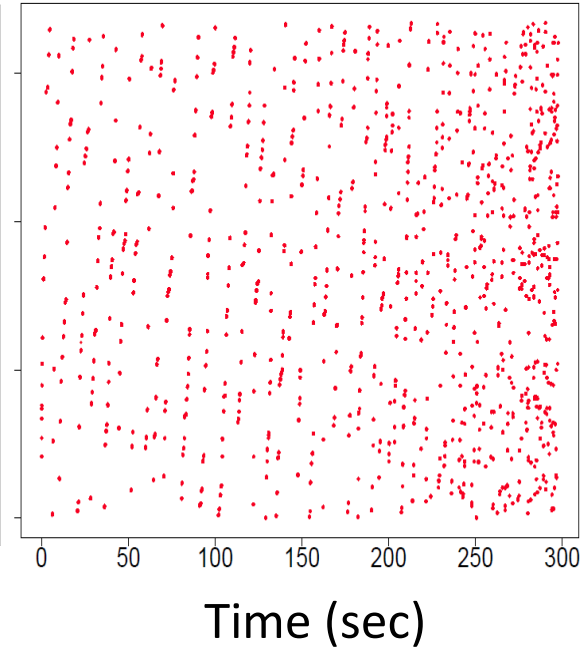
## Sequential Scan



## Index Scan



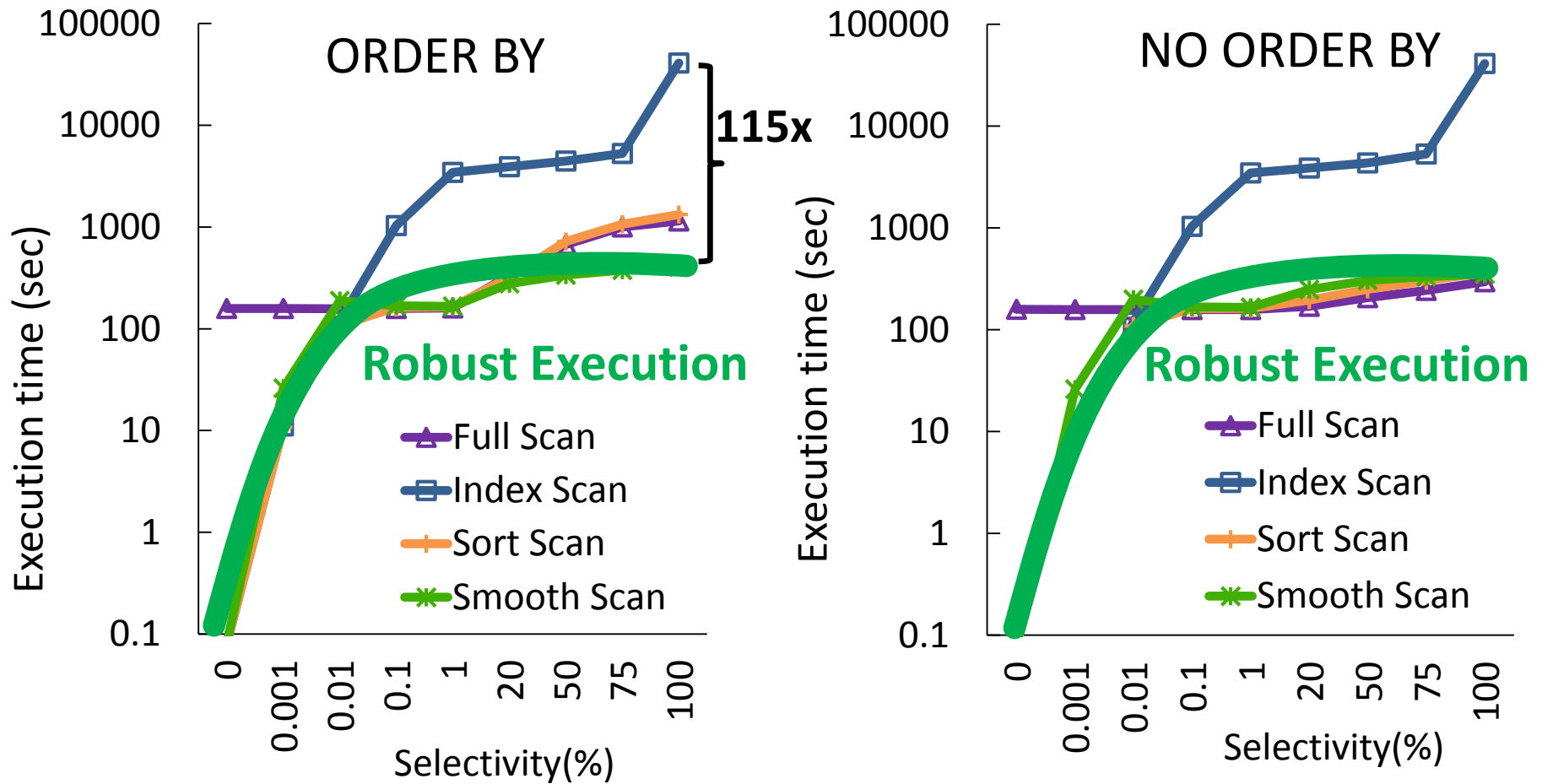
## Smooth Scan



**Smooth Scan reduces random I/O requests**

# Adaptivity over selectivity range

**Setting:** Micro-benchmark, Q1 (w. and w/o. order), Selectivity 0-100%



**Near-optimal performance throughout entire range**

# Conclusions

## SMOOTH SCAN

- **Statistics-oblivious** access path
- Uses region snooping to **morph** between alternatives
- **Near-optimal performance** for all selectivities

## IMPACT

- Removes access path selection decision
- Robust execution for all query inputs



# Q & A



[renata.borovica@epfl.ch](mailto:renata.borovica@epfl.ch)



[stratos@seas.harvard.edu](mailto:stratos@seas.harvard.edu)



[natassa@epfl.ch](mailto:natassa@epfl.ch)



[marcin@snowflake.net](mailto:marcin@snowflake.net)



[campbellf@google.com](mailto:campbellf@google.com)

# Thank you!