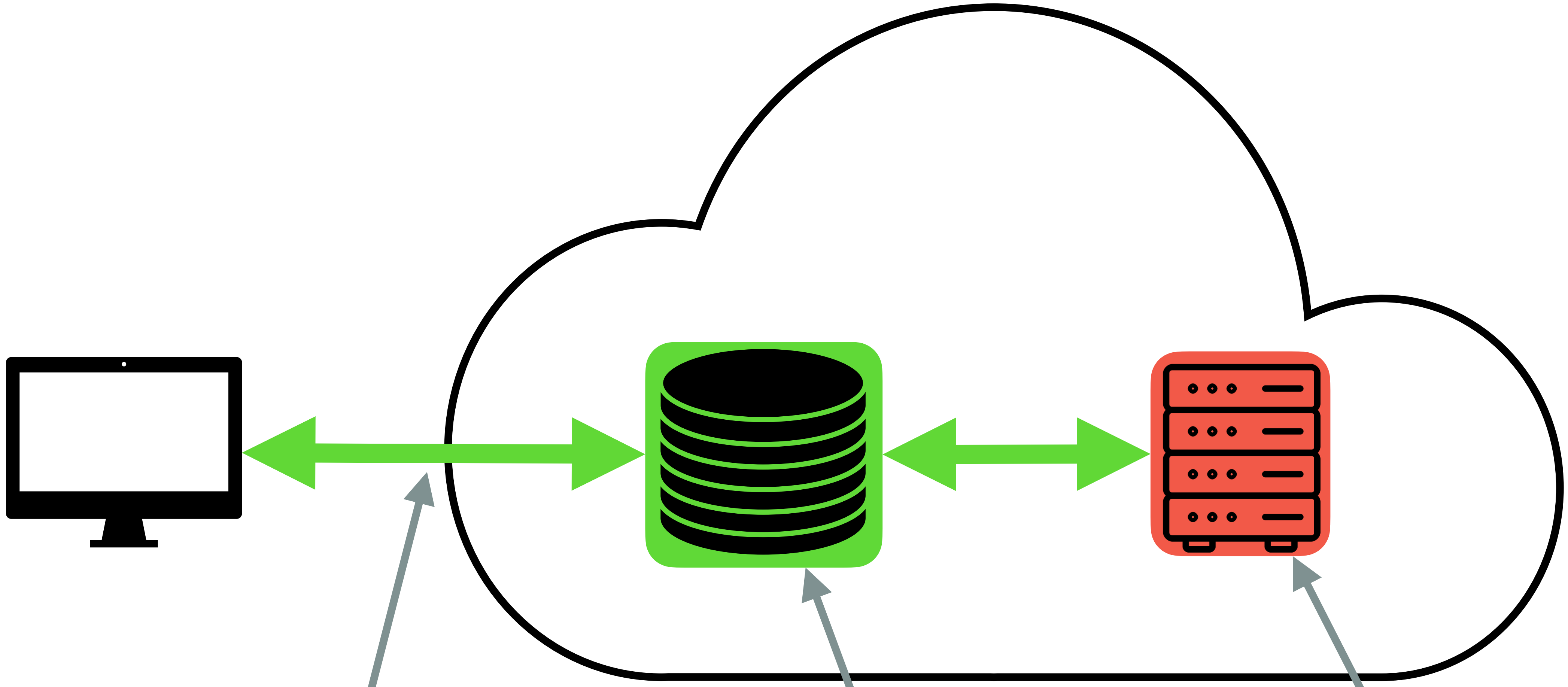
Cracking-Like Join for Trusted Execution Environments

Kajetan Maliszewski, Jorge-Arnulfo Quiané-Ruiz*, Volker Markl

VLDB 2023

* This work is dedicated to the memory of Jorge.

Data in-use are currently not protected

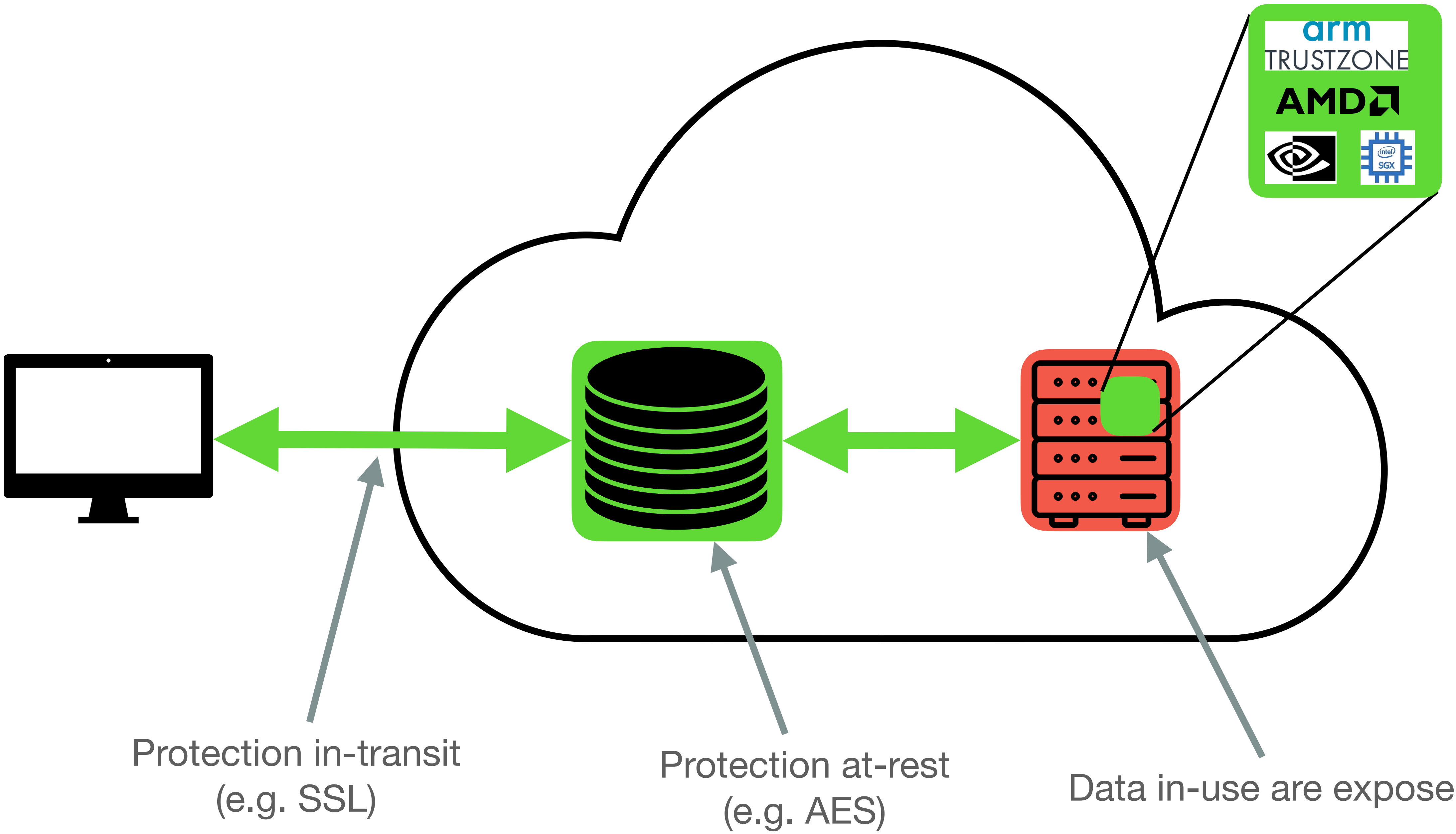


Protection in-transit
(e.g. SSL)

Protection at-rest
(e.g. AES)

Data in-use are exposed!

Data in-use are currently not protected



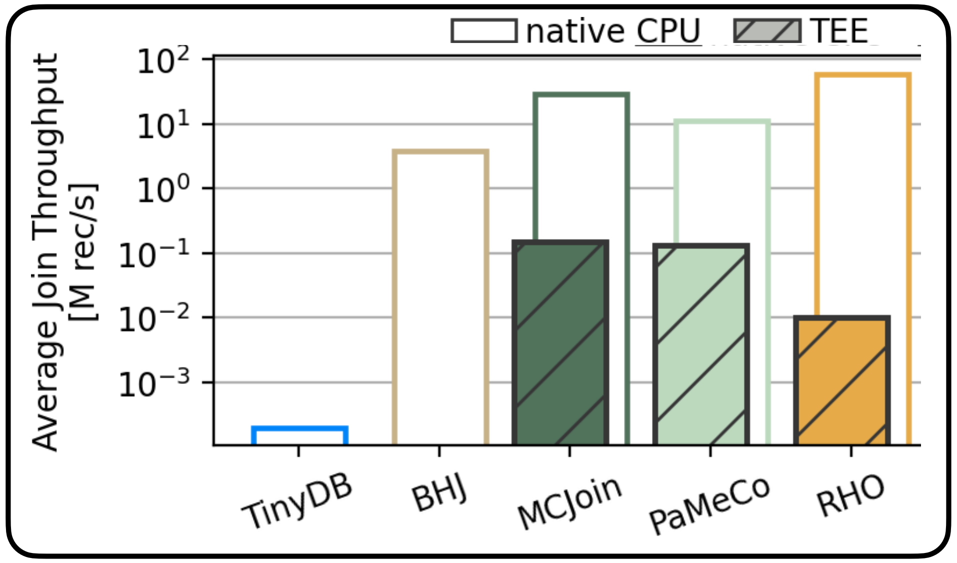
Protection in-transit
(e.g. SSL)

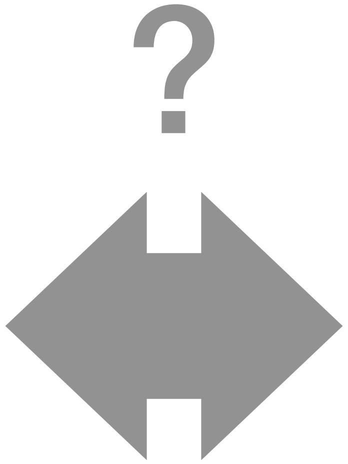
Protection at-rest
(e.g. AES)

Data in-use are exposed!

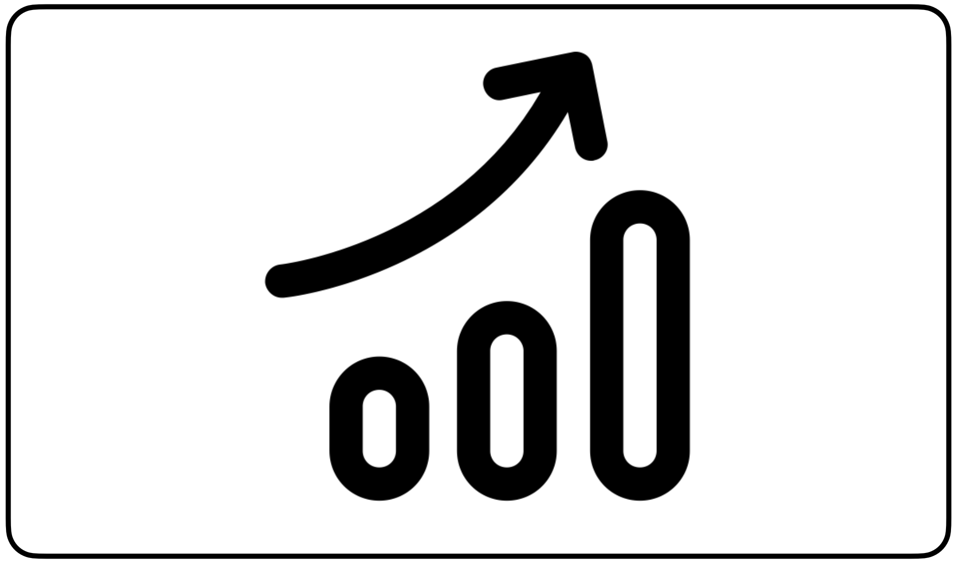
Performance of TEEs is an open challenge

CPU vs. TEE

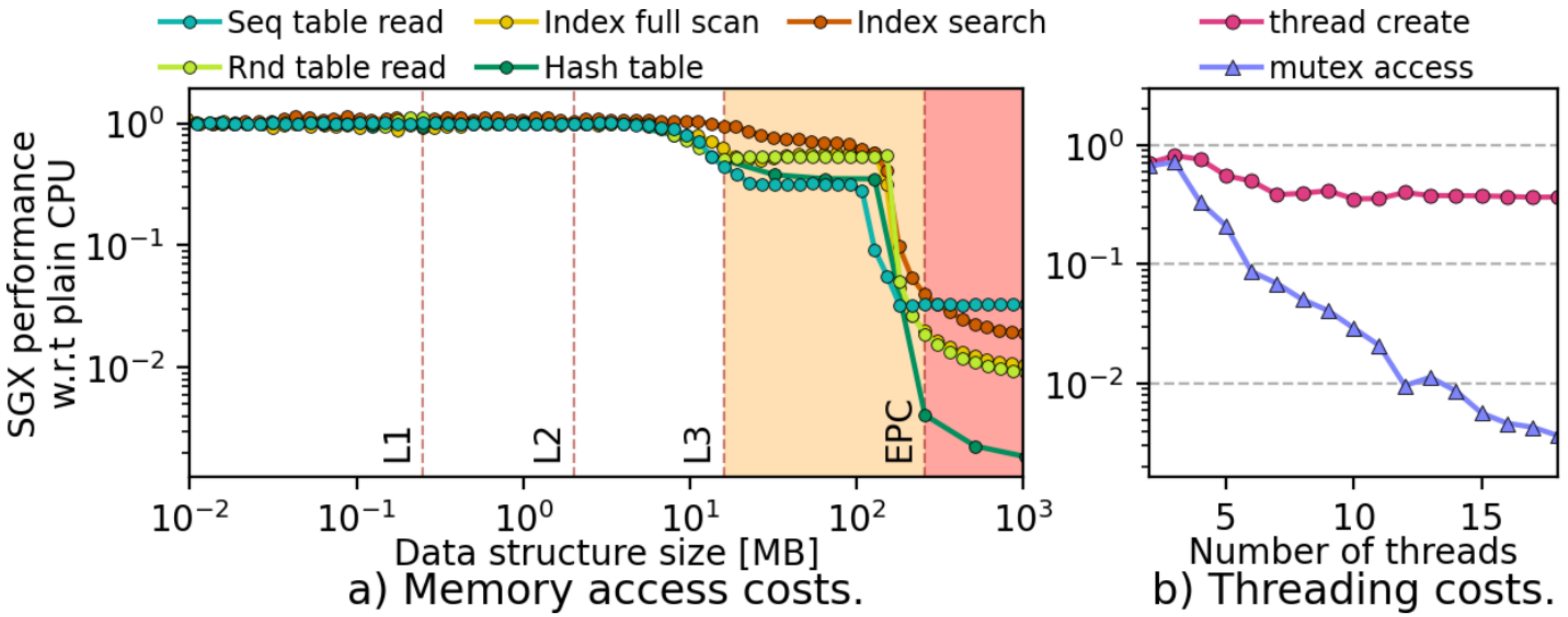




High Performance



TEEs need new primitives



a) Memory access costs.

b) Threading costs.

Desiderata for TEE-native processing:

D1. Access patterns
D2. Low memory consumption
D3. Wait-free algorithms

Cracking-Like Philosophy

Rules

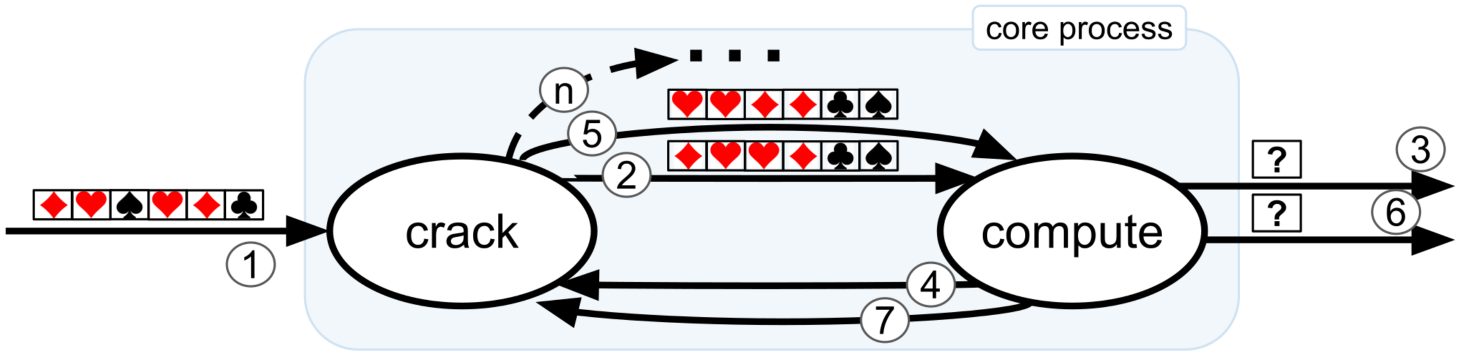
partition the data

perform only sequential scans

consume little memory

design barrier-free data structures

Process



b) Cracking-Like Philosophy (CLP)

Cracking-Like Philosophy

Rules

partition the data

perform only sequential scans

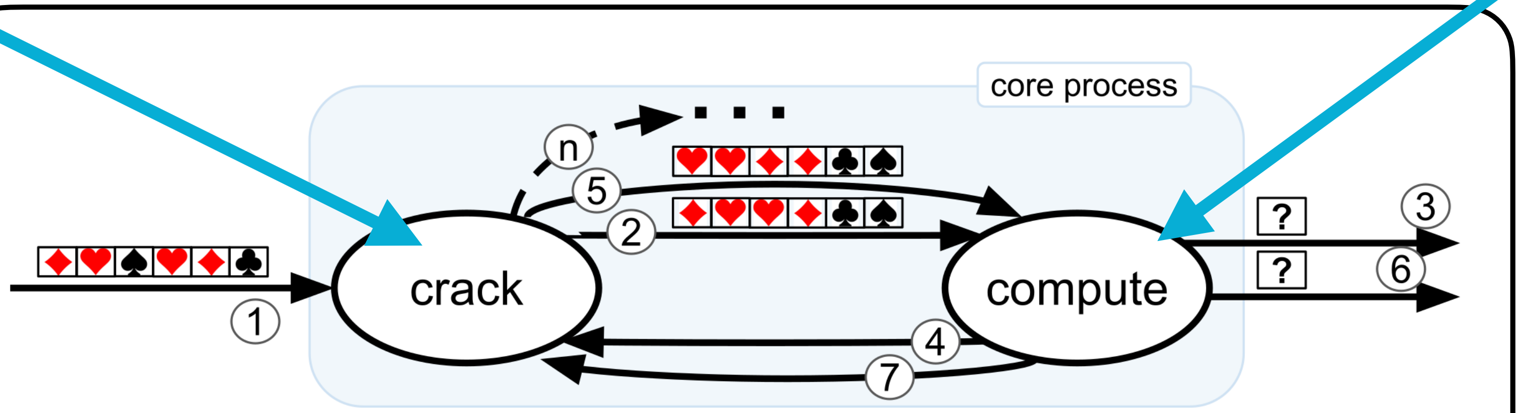
consume little memory

design barrier-free data structures

"in-place radix sort" bit by bit

arbitrary computation

Process



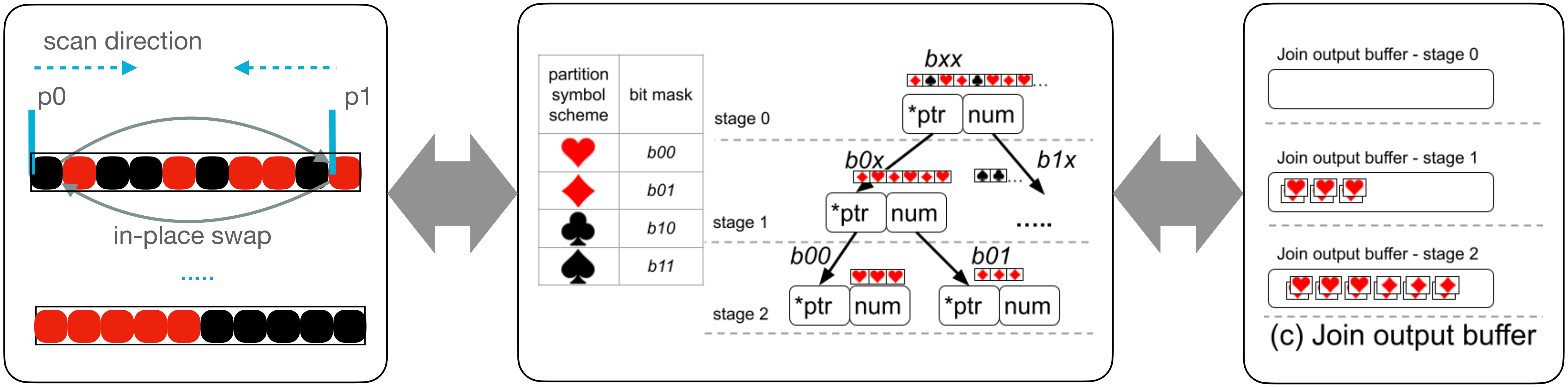
b) Cracking-Like Philosophy (CLP)

Two new primitives perform a Cracking-Like Join

Stage is a computation primitive

Cracking Tree is a storage primitive

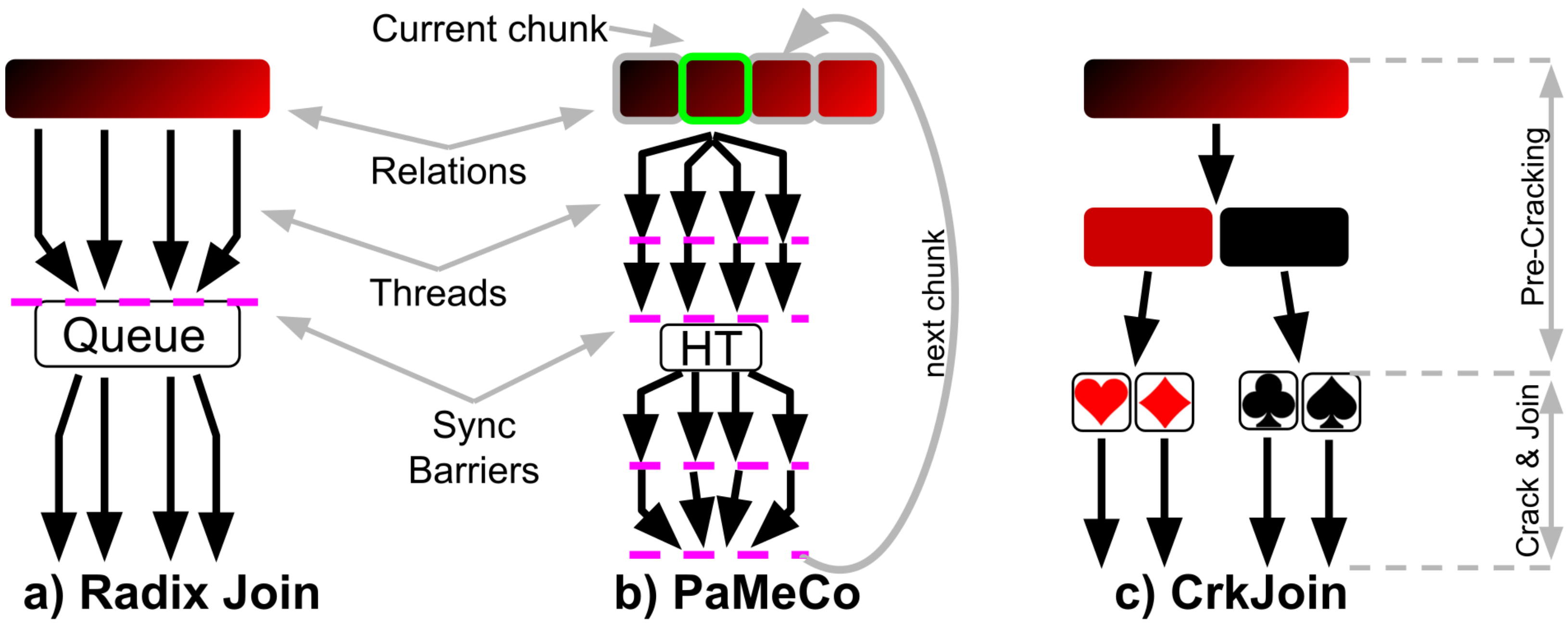
We join tuples
after every stage



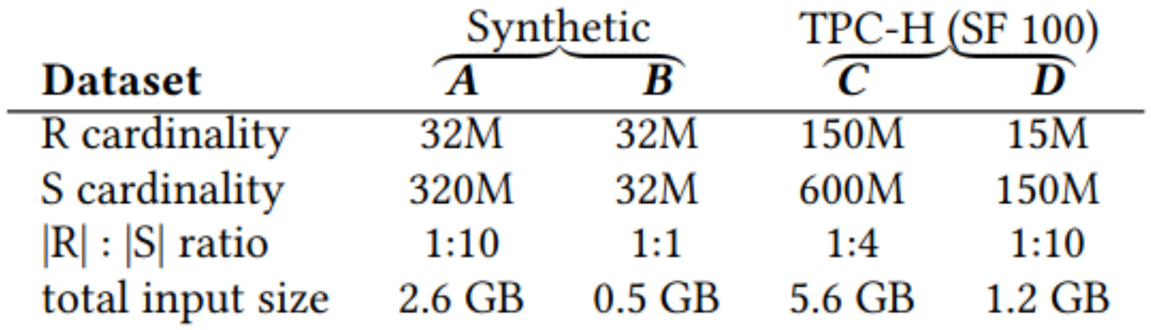
partition symbol scheme	bit mask
♥	*b00*
♦	*b01*
♣	*b10*
♠	*b11*

(c) Join output buffer

Partially partitioned relations form independent chunks



Experimental setup

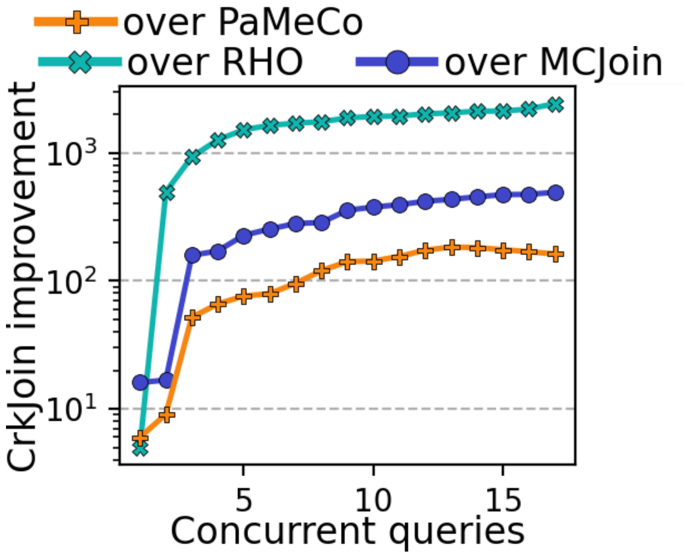


	Synthetic		TPC-H (SF 100)	
Dataset	***A***	***B***	***C***	***D***
R cardinality	32M	32M	150M	15M
S cardinality	320M	32M	600M	150M
|R| : |S| ratio	1:10	1:1	1:4	1:10
total input size	2.6 GB	0.5 GB	5.6 GB	1.2 GB

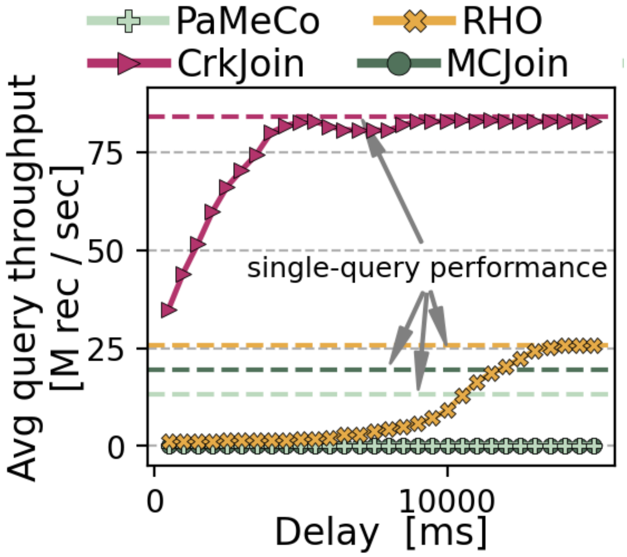
- Intel Xeon E-2278G CPU
- 125 GB RAM, 256 MB EPC
- Ubuntu 20 OS, SGX driver v2.15

- Radix Join (RHO)
- MCJoin
- PaMeCo

CrkJoin outperforms the baselines in a multi-tenant scenario

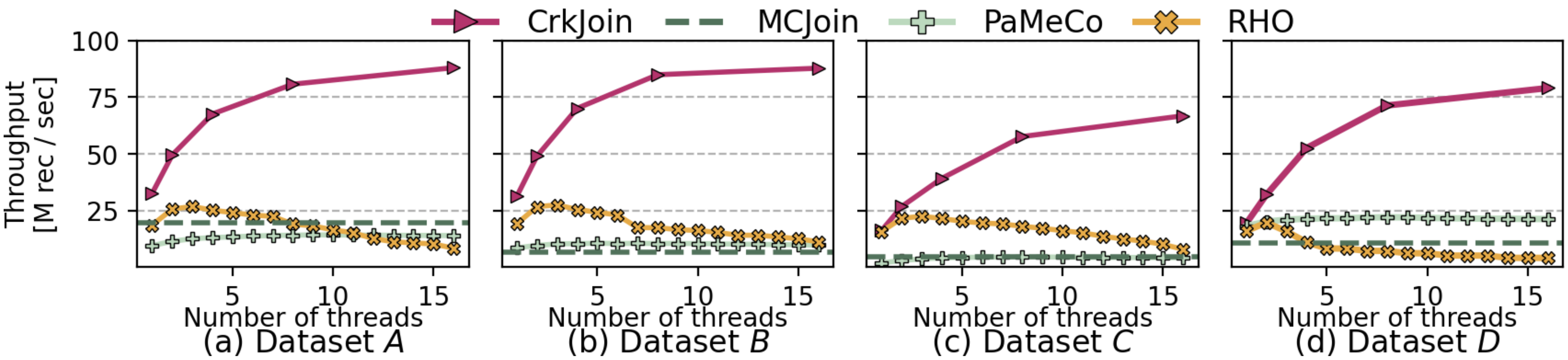


(a) Dataset *A*

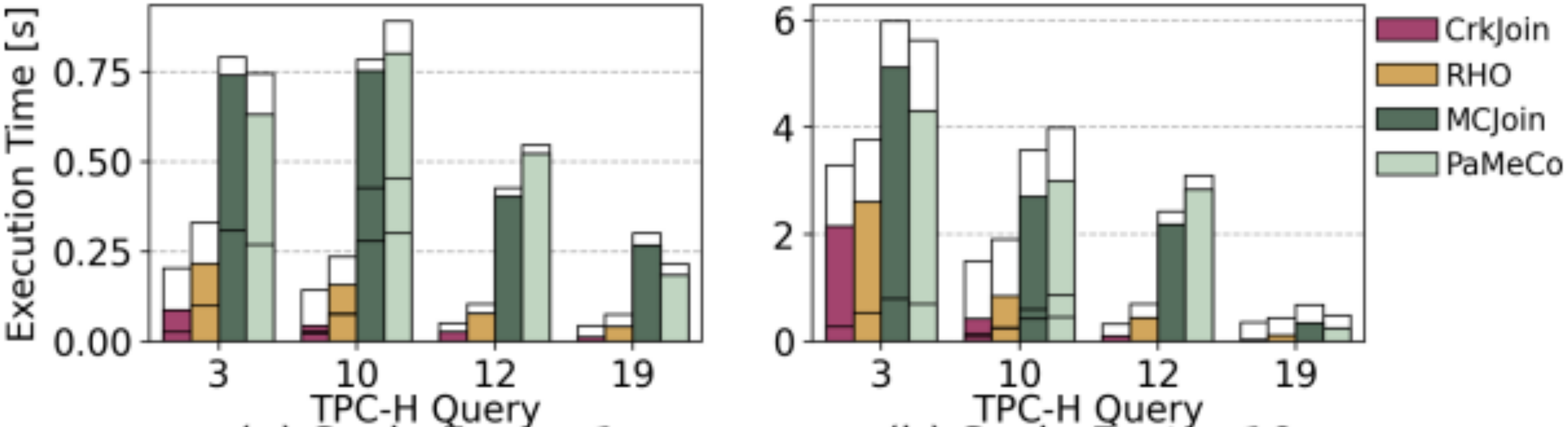


(c) Dataset *A*

CrkJoin scales to multi-core architectures



CrkJoin speeds up TPC-H queries by up to 11x



(a) Scale Factor 1

(b) Scale Factor 10

Contributions

New primitives for TEEs

Cracking-Like Philosophy addresses the desiderata

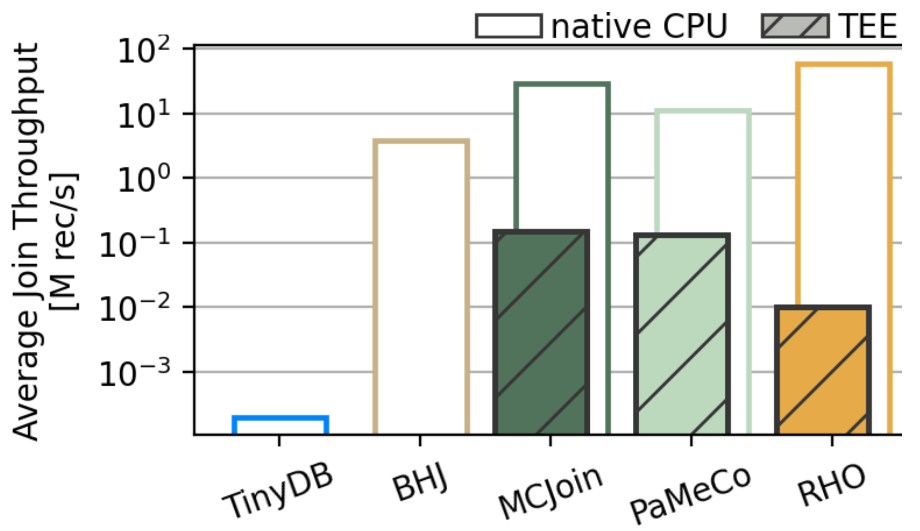
CrkJoin achieves superior performance in TEEs

Cracking-Like Join for Trusted Execution Environments

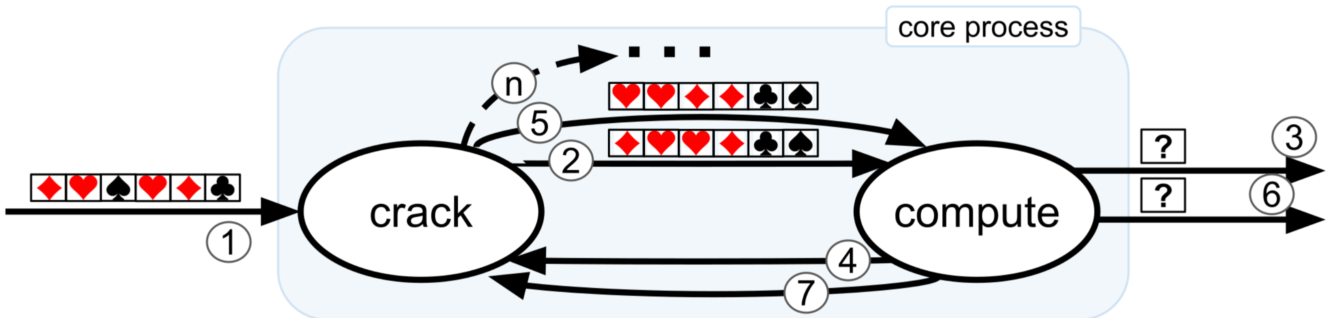
Kajetan Maliszewski, Jorge-Arnulfo Quiané-Ruiz, Volker Markl

Proceedings of the VLDB Endowment 16.9 (2023): 2330-2343.

Existing join algorithms underperform in TEEs

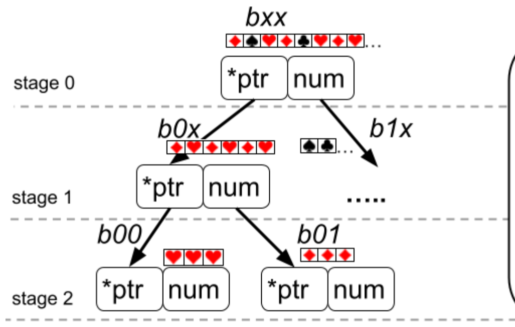


We propose Cracking-Like Philosophy (CLP), a novel processing method crafted for TEEs

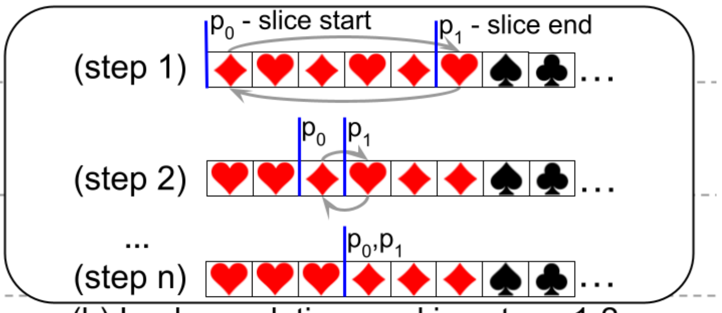


b) Cracking-Like Philosophy (CLP)

We built a relational join algorithm based on CLP. For that, we propose two new primitives for computation and storage.

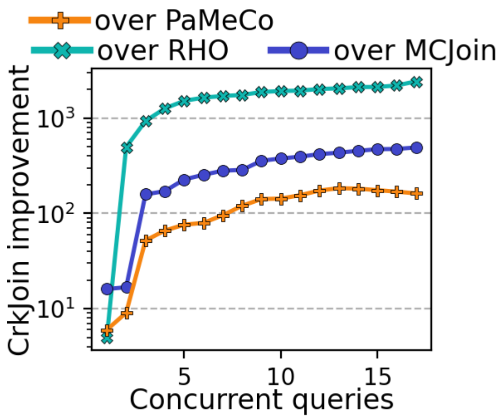


(a) Cracking Tree

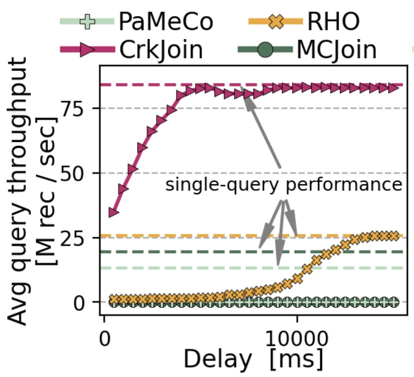


(b) In-place relation cracking stage 1-2

Cracking-Like Join outperforms the baselines by up to three orders of magnitude in a multi-tenant cloud scenario.



(a) Dataset *A*



(c) Dataset *A*