CS265
Data Blocks

Will Deuschle

1. What is the problem?

Concepts

OLTP/OLAP

- Compression
- Vectorization
- Compilation

2. Why is it important?

Online Transaction Processing (OLTP)

Online Analytic Processing (OLAP)

3. Why is it hard?

OLTP/OLAP Trade-off

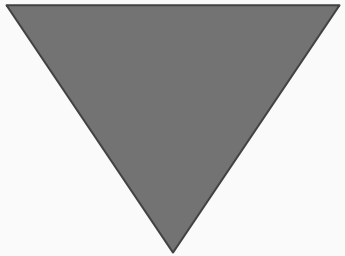
&&

4. Why don’t existing solutions work?

Today’s Metrics

Memory Footprint

Transaction Throughput (OLTP)

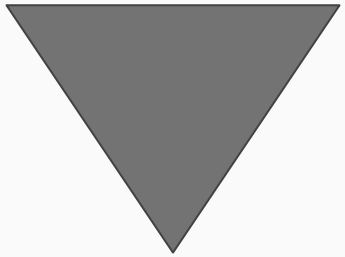


Query Performance (OLAP)

Today’s Knobs

Compression

Vectorization

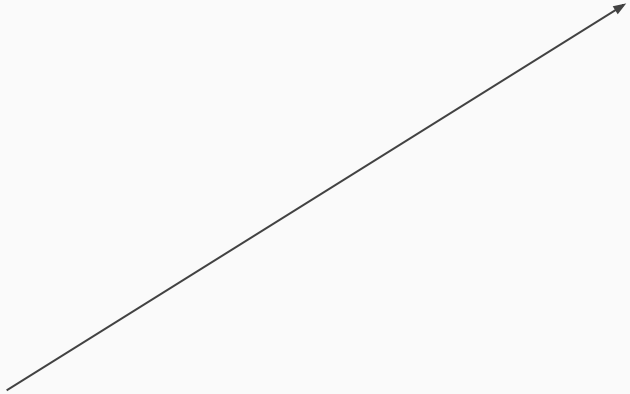


Compilation

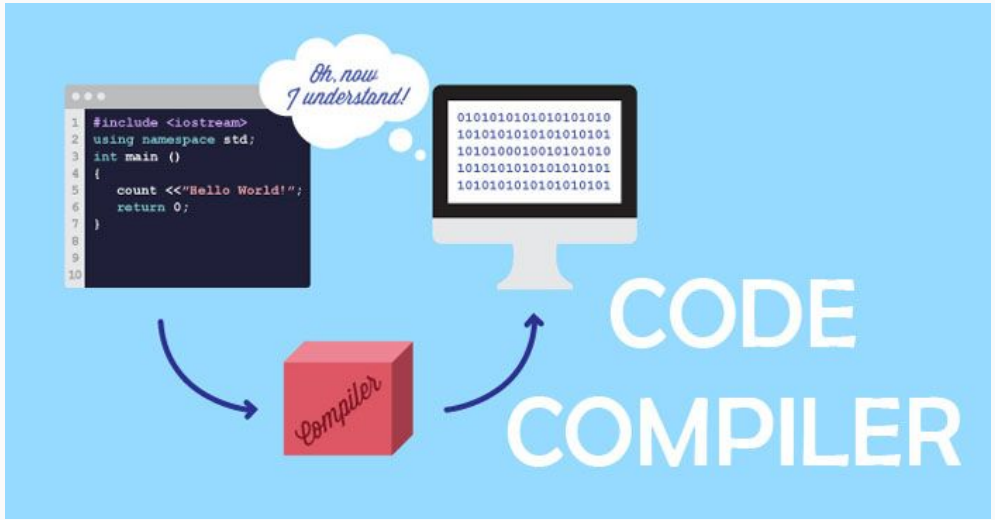
5. What is the core intuition for the solution?

Compression

Vectorization



Compilation



Enter Data Blocks

“This work aims at reducing the main-memory footprint in high performance hybrid OLTP & OLAP databases, while retaining high query performance and transactional throughput.” - First sentence of abstract

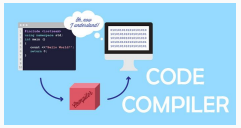
Which translates to...

Compression

OLAP

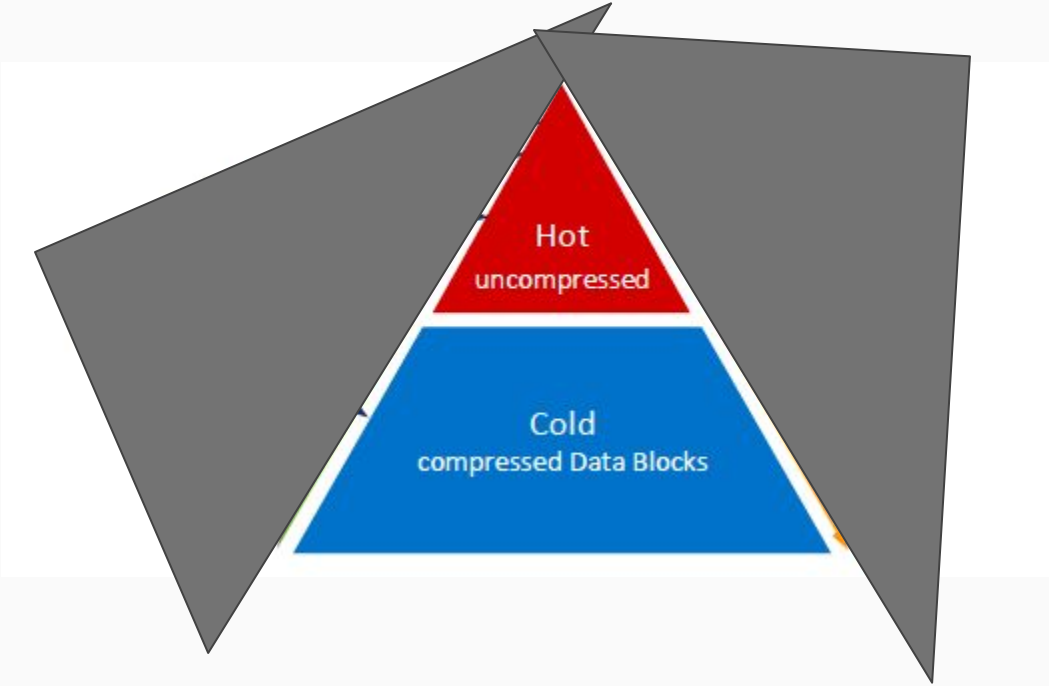
OLTP

How?

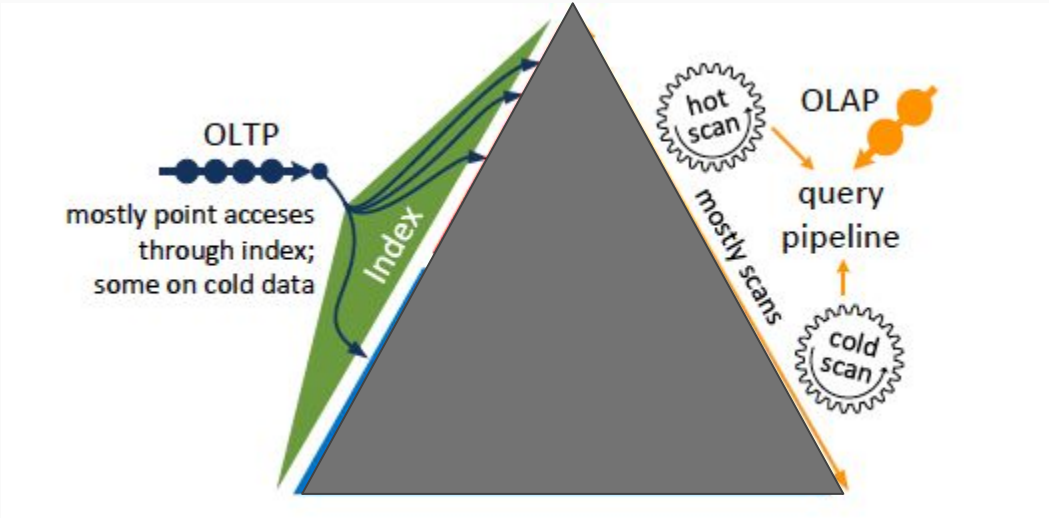


High Level View

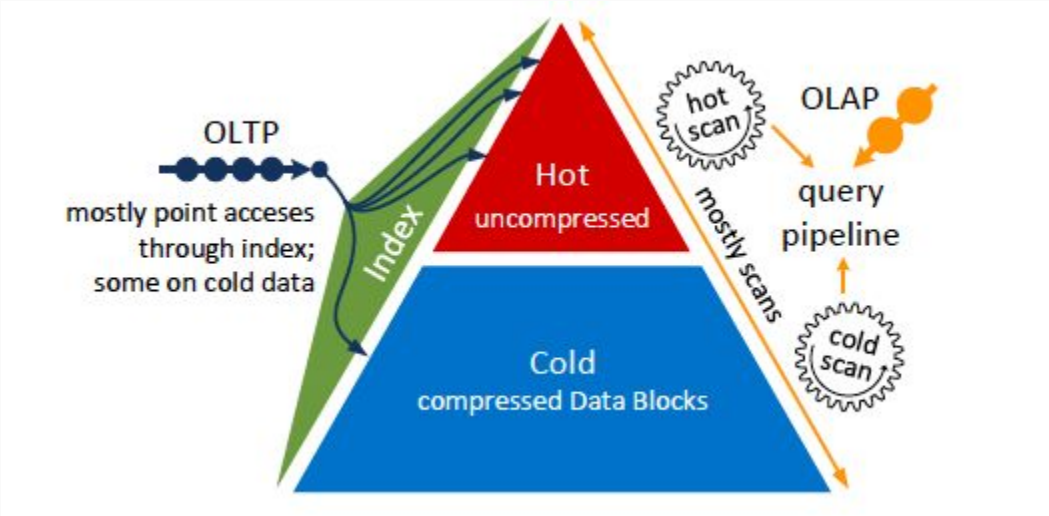
High Level View



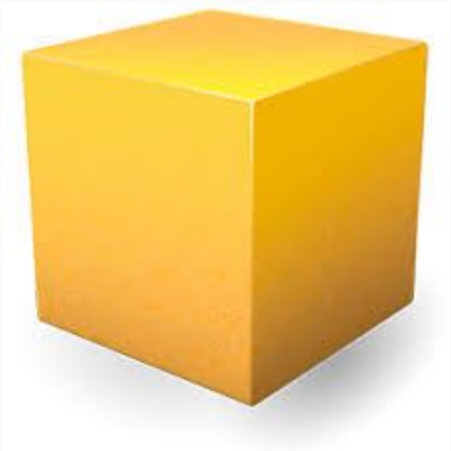
High Level View



High Level View



Unit of interest: Data Block



idea

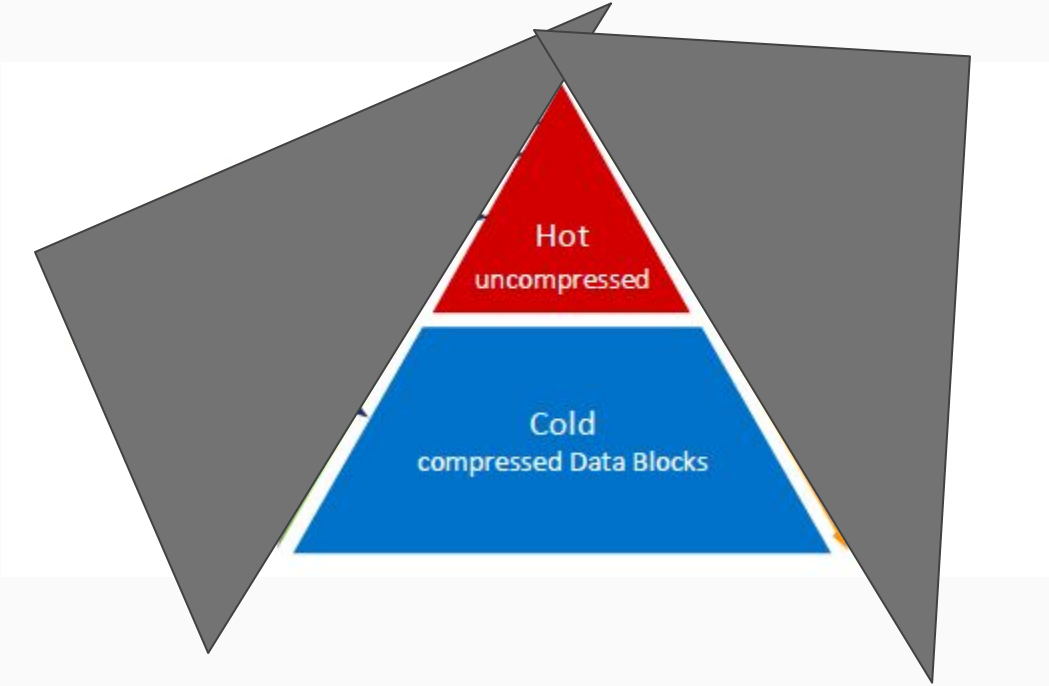
idea

idea

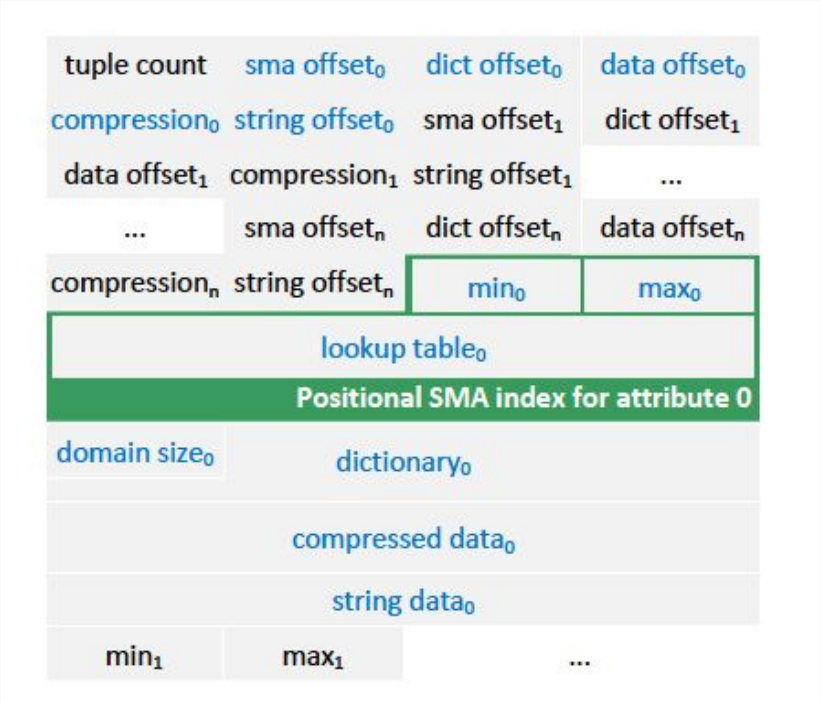
idea

idea

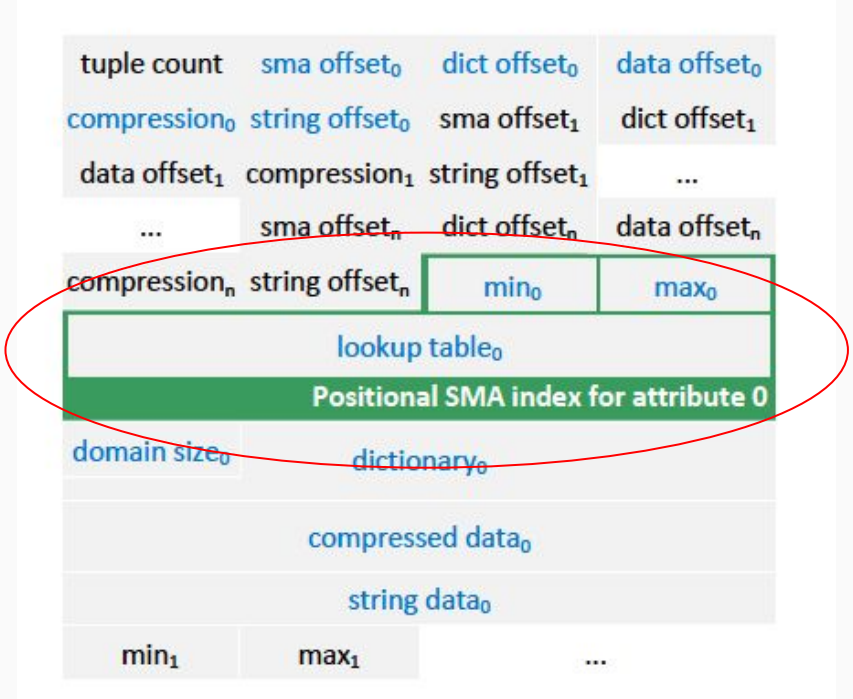
Hot & Cold, Immutability



Block Layout



SMAs and PSMAs



3 Compression Schemes

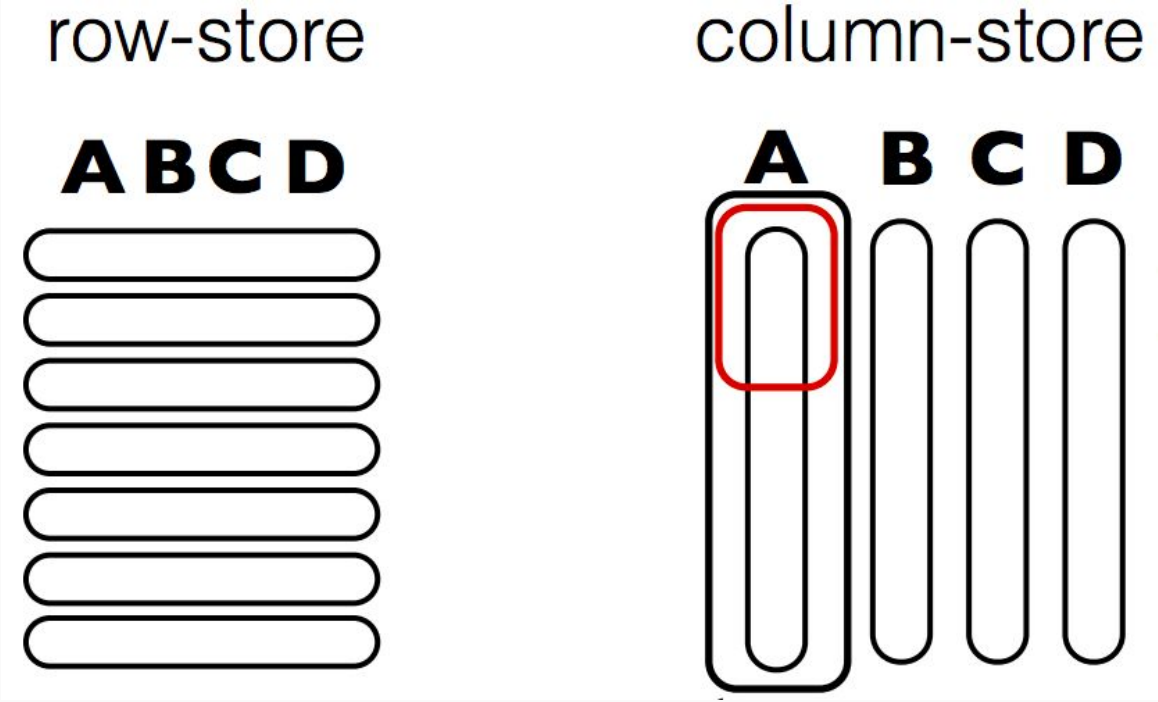
- Single value
- Ordered dictionary
- Truncation (deltas)

*important: byte-addressability and SIMD over compressed data

JIT compilers challenge

- **Problem**: Different compression scheme combinations
- **Solution:** Unrolling? Runtime branching overhead? Pre-compiled vectorized scan code?

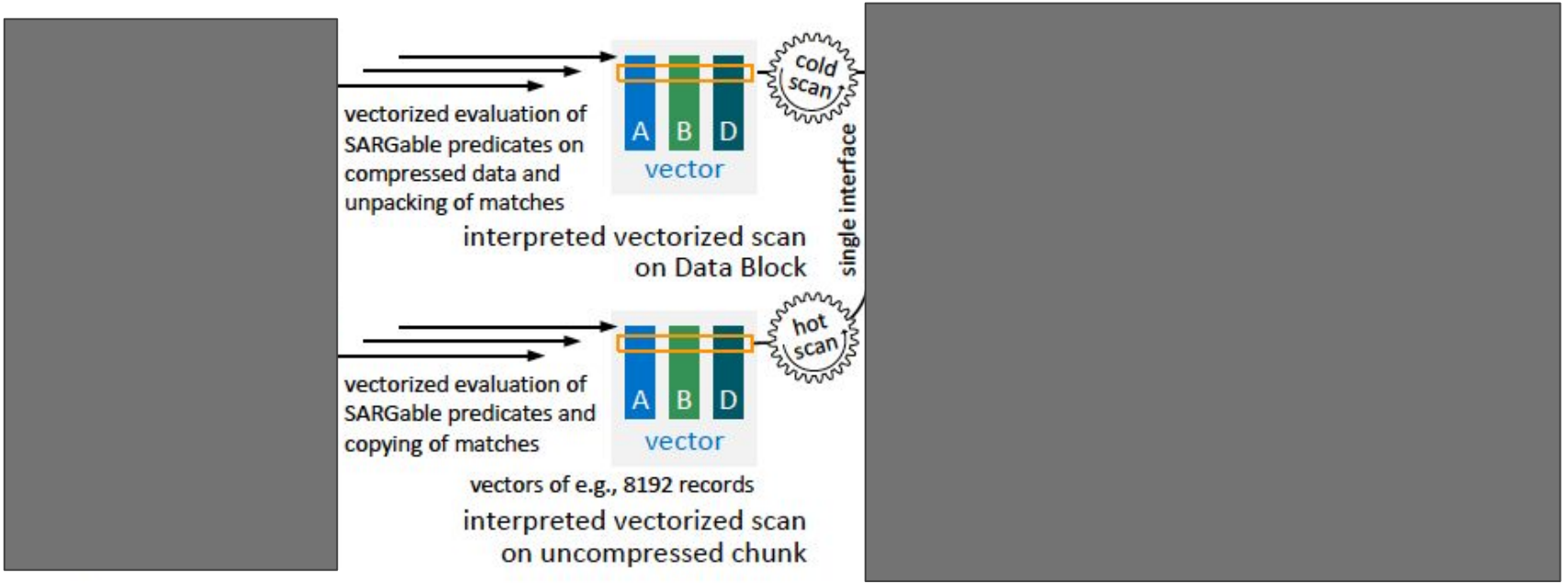
SIMD



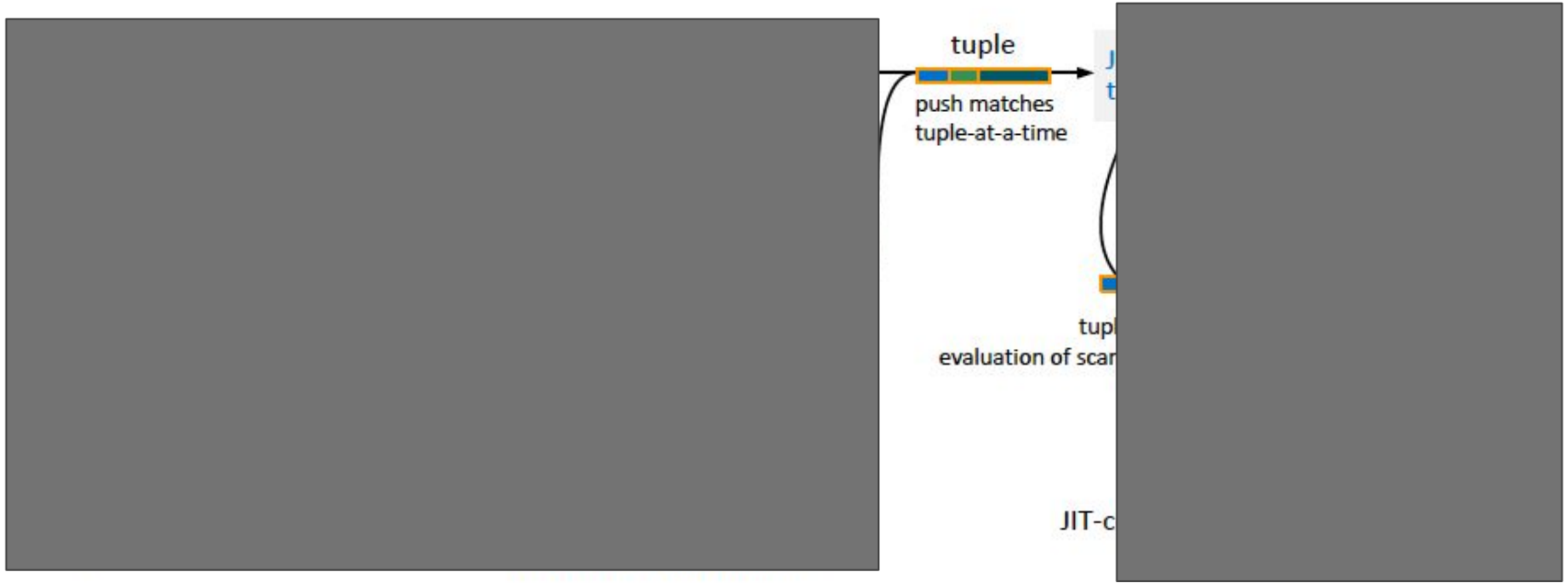
Visualization of handling queries

Visualization of handling queries

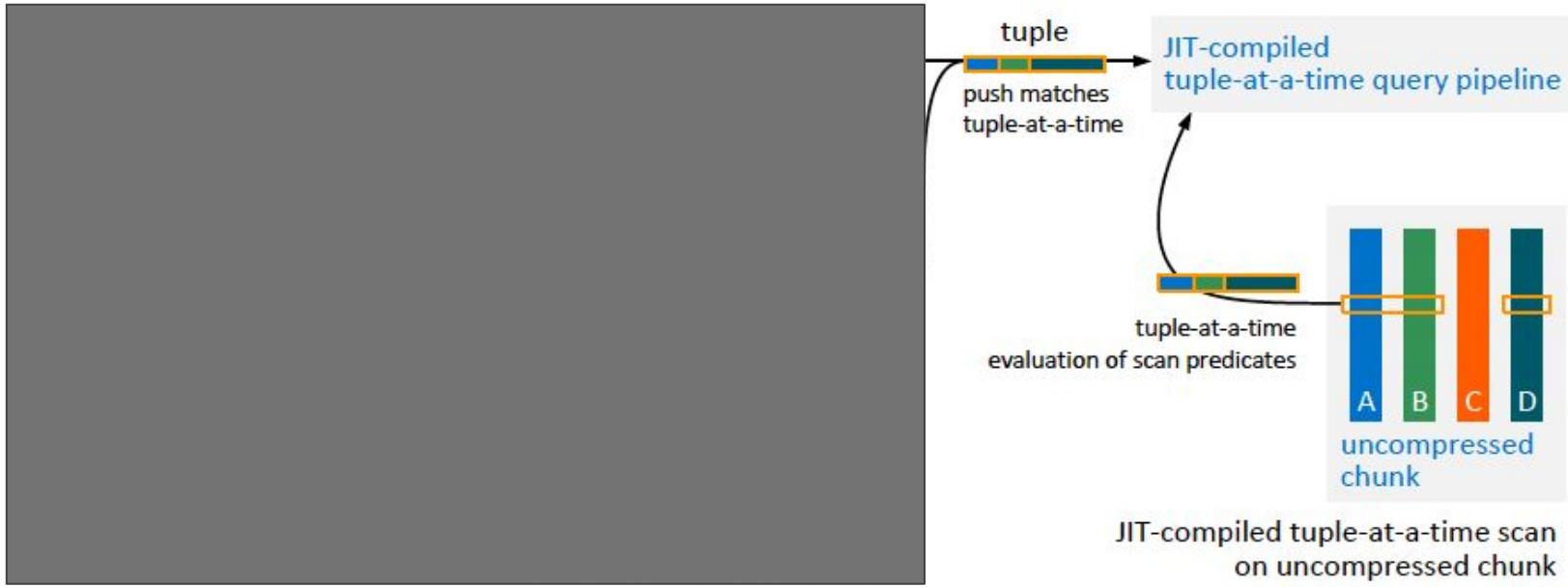
Visualization of handling queries



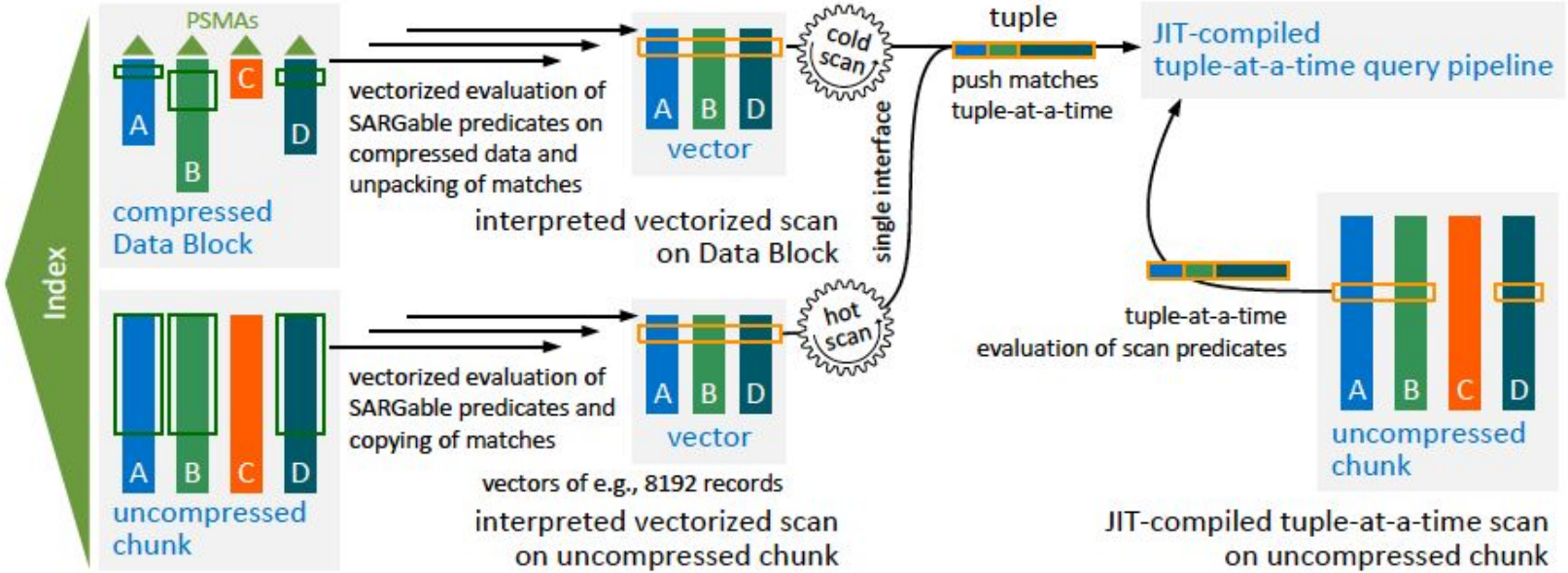
Visualization of handling queries



Visualization of handling queries



Visualization of handling queries



6. Does the paper prove its claims?

Remember our metrics

- Compression
- Query performance (OLAP)
- Transaction performance (OLTP)

7. Setup of experiments? Are they sufficient?

Compression

	TPC-H SF100	IMDB cast info	Flights
uncompressed			
CSV	107 GB	1.4 GB	12 GB
HyPer	126 GB	1.8 GB	21 GB
Vectorwise	105 GB	0.72 GB	11 GB
compressed			
HyPer	66 GB	0.50 GB	4.2 GB
Vectorwise	54 GB	0.24 GB	3.2 GB

Table 1: Size of TPC-H, IMDB cast info, and a flight details database in HyPer and Vectorwise.

Compression

	TPC-H SF100	IMDB cast info	Flights
	uncompressed		
CSV	107 GB	1.4 GB	12 GB
HyPer	126 GB	1.8 GB	21 GB
Vectorwise	105 GB	0.72 GB	11 GB
	compressed		
HyPer	66 GB	0.50 GB	4.2 GB
Vectorwise	54 GB	0.24 GB	3.2 GB

Table 1: Size of TPC-H, IMDB cast info, and a flight details database in HyPer and Vectorwise.

Compression

	TPC-H SF100	IMDB cast info	Flights
	uncompressed		
CSV	107 GB	1.4 GB	12 GB
HyPer	126 GB	1.8 GB	21 GB
Vectorwise	105 GB	0.72 GB	11 GB
	compressed		
HyPer	66 GB	0.50 GB	4.2 GB
Vectorwise	54 GB	0.24 GB	3.2 GB

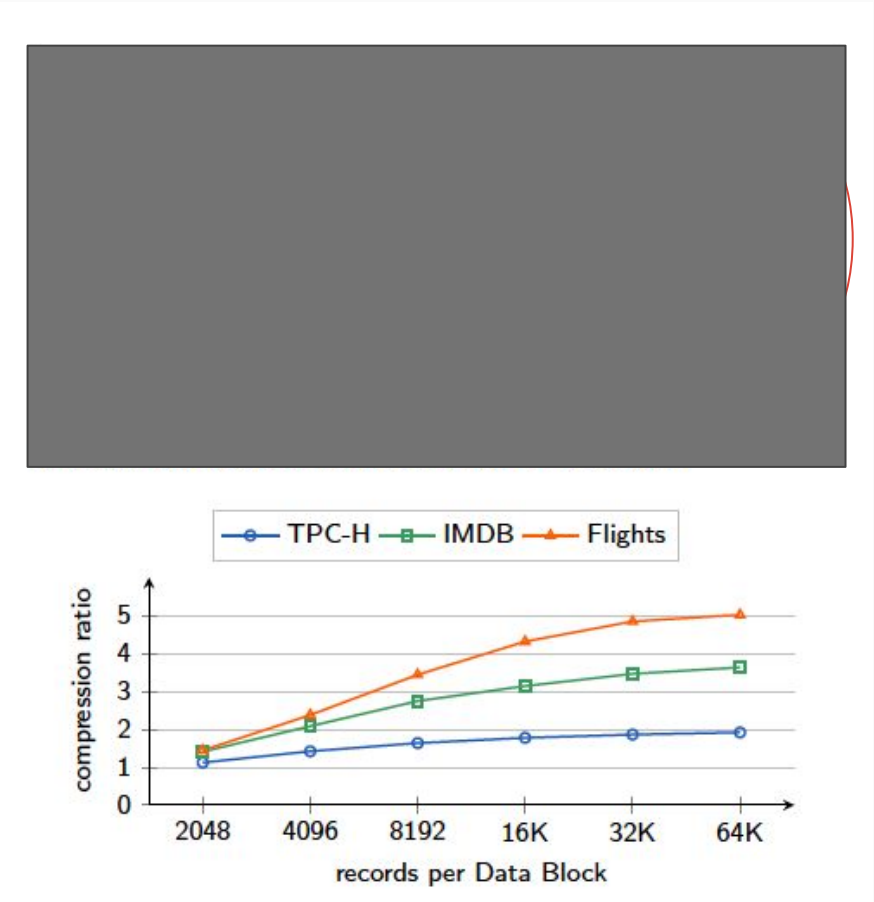
Table 1: Size of TPC-H, IMDB cast info, and a flight details database in HyPer and Vectorwise.

Compression

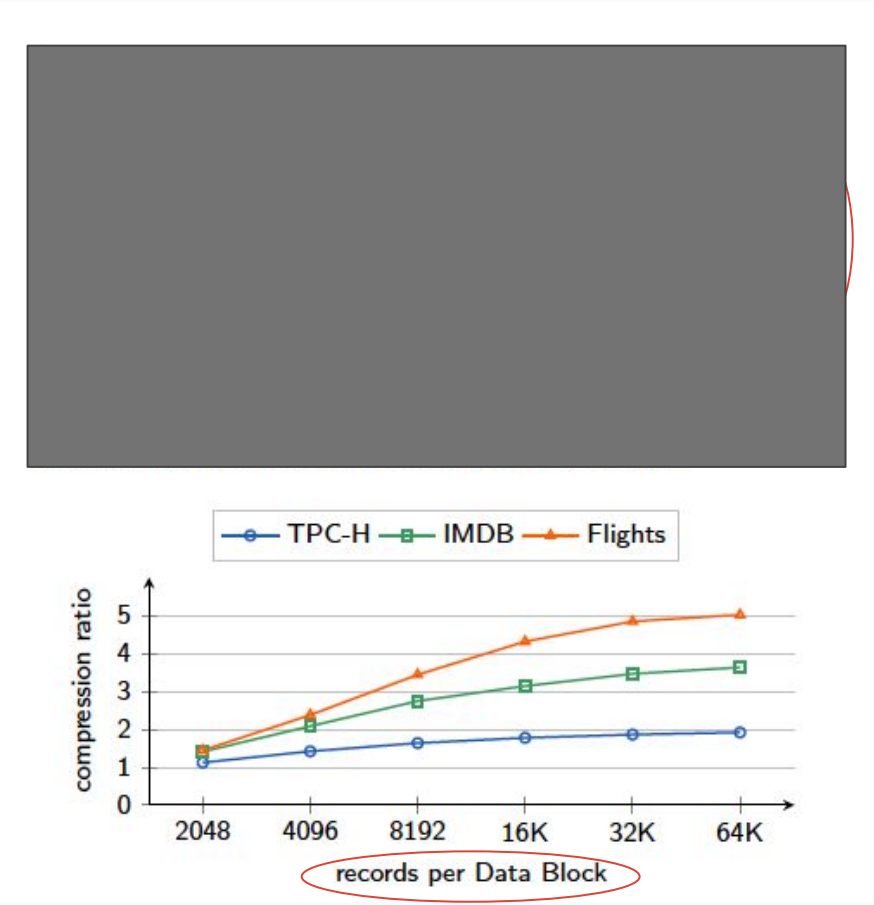
	TPC-H SF100	IMDB cast info	Flights
	uncompressed		
CSV	107 GB	1.4 GB	12 GB
HyPer	126 GB	1.8 GB	21 GB
Vectorwise	105 GB	0.72 GB	11 GB
	compressed		
HyPer	66 GB	0.50 GB	4.2 GB
Vectorwise	54 GB	0.24 GB	3.2 GB

Table 1: Size of TPC-H, IMDB cast info, and a flight details database in HyPer and Vectorwise.

Compression



Compression



Query Performance (OLAP)

scan type	geometric mean	sum
HyPer		
JIT (uncomressed)	0.586s	21.7s
Vectorized (uncompressed)	0.583s (1.01×)	21.6s
+ SARG	0.577s (1.02×)	21.8s
Data Blocks (compressed)	0.555s (1.06×)	21.5s
+ SARG/SMA	0.466s (1.26×)	20.3s
+ PSMA	0.463s (1.27×)	20.2s
Vectorwise		
uncompressed storage	2.336s	74.4s
compressed storage	2.527s (0.92×)	78.5s

Query Performance (OLAP)

scan type	geometric mean	sum
HyPer		
JIT (uncomressed)	0.586s	21.7s
Vectorized (uncompressed)	0.583s (1.01×)	21.6s
+ SARG	0.577s (1.02×)	21.8s
Data Blocks (compressed)	0.555s (1.06×)	21.5s
+ SARG/SMA	0.466s (1.26×)	20.3s
+ PSMA	0.463s (1.27×)	20.2s
Vectorwise		
uncompressed storage	2.336s	74.4s
compressed storage	2.527s (0.92×)	78.5s

Query Performance (OLAP)

scan type	geometric mean	sum
HyPer		
JIT (uncomressed)	0.586s	21.7s
Vectorized (uncompressed)	0.583s (1.01×)	21.6s
+ SARG	0.577s (1.02×)	21.8s
Data Blocks (compressed)	0.555s (1.06×)	21.5s
+ SARG/SMA	0.466s (1.26×)	20.3s
+ PSMA	0.463s (1.27×)	20.2s
Vectorwise		
uncompressed storage	2.336s	74.4s
compressed storage	2.527s (0.92×)	78.5s

Transaction Performance (OLTP)

		Ordered*	Shuffled°
uncompressed	PK index	551,268	545,554
(JIT)	no index	36	36
uncompressed	PK index	550,661	566,893
(Vectorized)	no index	26	26
Data Blocks	PK index	301,750	274,198
	no index	17,508	41
Data Blocks	PK index	276,014	294,291
+PSMA	no index	71,587	40

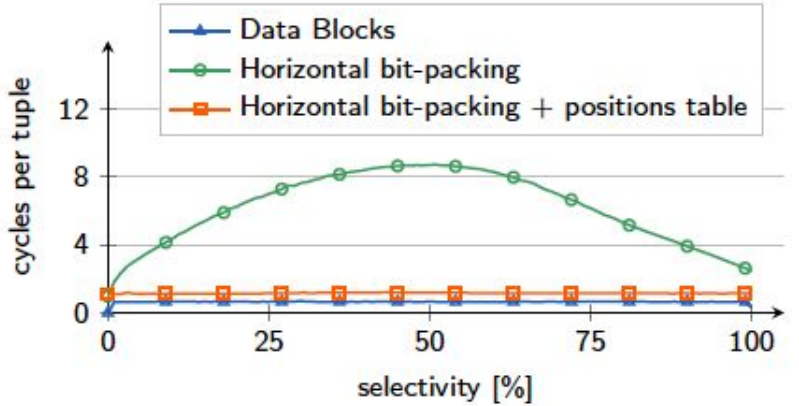
Transaction Performance (OLTP)

		Ordered*	Shuffled°
uncompressed (JIT)	PK index	551,268	545,554
	no index	36	36
uncompressed (Vectorized)	PK index	550,661	566,893
	no index	26	26
Data Blocks	PK index	301,750	274,198
	no index	17,508	41
Data Blocks +PSMA	PK index	276,014	294,291
	no index	71,587	40

Transaction Performance (OLTP)

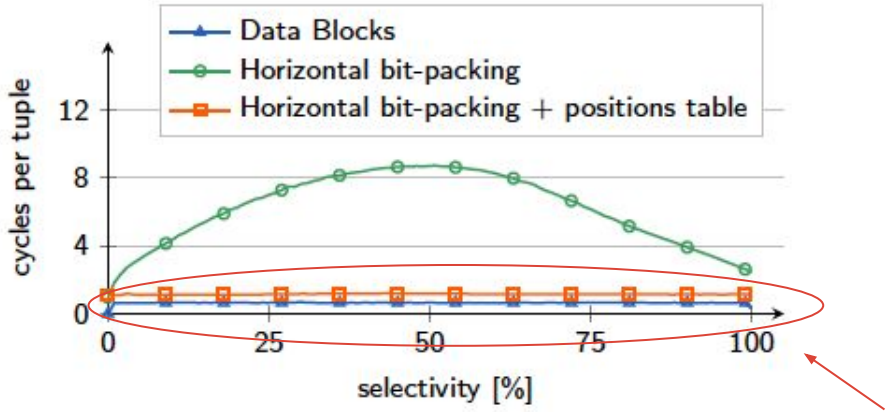
		Ordered*	Shuffled°
uncompressed	PK index	551,268	545,554
(JIT)	no index	36	36
uncompressed	PK index	550,661	566,893
(Vectorized)	no index	26	26
Data Blocks	PK index	301,750	274,198
	no index	17,508	41
Data Blocks	PK index	276,014	294,291
+PSMA	no index	71,587	40

Byte-addressability



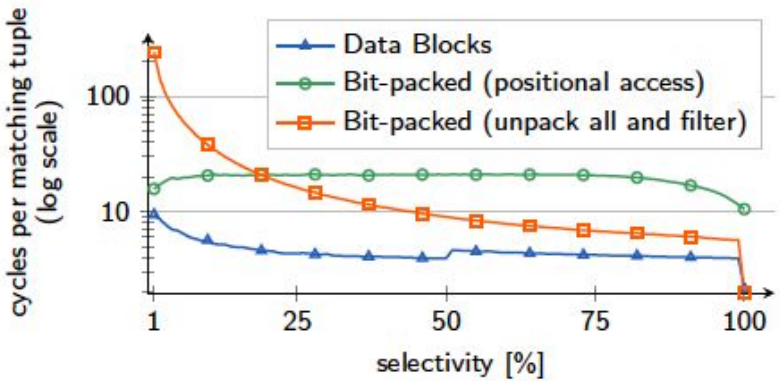
(a) Cost of evaluating a SARGable predicate of type $l \leq A \leq r$ with varying selectivities

Byte-addressability



(a) Cost of evaluating a SARGable predicate of type $l \leq A \leq r$ with varying selectivities

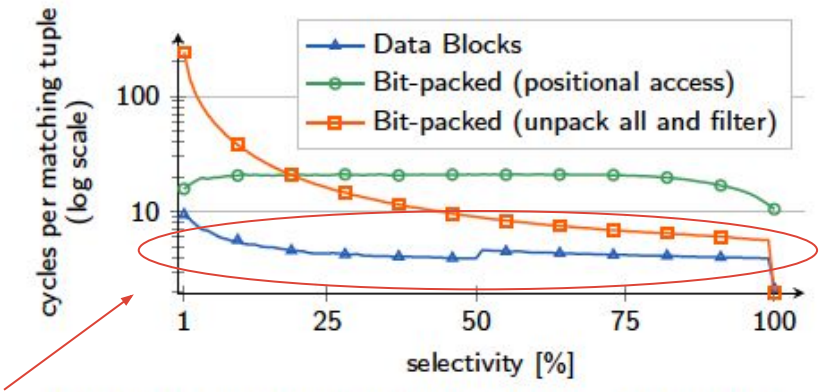
Byte-addressability



(b) Cost of unpacking matching tuples (3 attributes)

Figure 12: Horizontal bit-packing compared to Data Blocks with byte-addressable compression schemes

Byte-addressability



(b) Cost of unpacking matching tuples (3 attributes)

Figure 12: Horizontal bit-packing compared to Data Blocks with byte-addressable compression schemes

Conclusions

Main takeaways

Compression

OLAP

OLTP

Reduce main-memory footprint in hybrid systems without a significant performance hit.

8 & 9. Gaps in the logic and proof - next steps for us?

Interesting directions

- Further compression (subbyte encodings) when moving to and from disk?
- Compression schemes over value domain: ML problem?
- Compression schemes for string data?
- SIMD vs. traditional point accesses/filtration schemes depending on various factors (previous filtration steps, distribution of the data, integer size, etc.): ML problem?