NULLS!

Revisiting Null Representation in Modern Columnar Formats

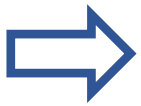
Xinyu Zeng, Ruijun Meng, Andrew Pavlo, Wes McKinney, Huanchen Zhang

Carnegie Mellon University

Null MATTERS

⇨ Every major DBMS and data file format supports Nulls today

Null MATTERS



Every major DBMS and data file format supports Nulls today

```
> CREATE TABLE t1 (i INTEGER NOT NULL, j INTEGER);
```

Null MATTERS

⇨ Every major DBMS and data file format supports Nulls today

```
> CREATE TABLE t1 (i INTEGER NOT NULL, j INTEGER);
```

⇨ More than 80% of DBMS users see Nulls often or regularly - [VLDB 2022]

Null MATTERS

⇨ Every major DBMS and data file format supports Nulls today

```
> CREATE TABLE t1 (i INTEGER NOT NULL, j INTEGER);
```

⇨ More than 80% of DBMS users see Nulls often or regularly - [VLDB 2022]

*Yet, no deep investigation on how to handle Nulls in **modern file formats***

How Nulls are Represented

Logical

data
11
NULL
33
44
NULL
66

data
11
33
44
66

valid?
1
0
1
1
0
1

Compact

data
11
33
44
66

valid?
1
0
1
1
0
1

Placeholder

How Nulls are Represented

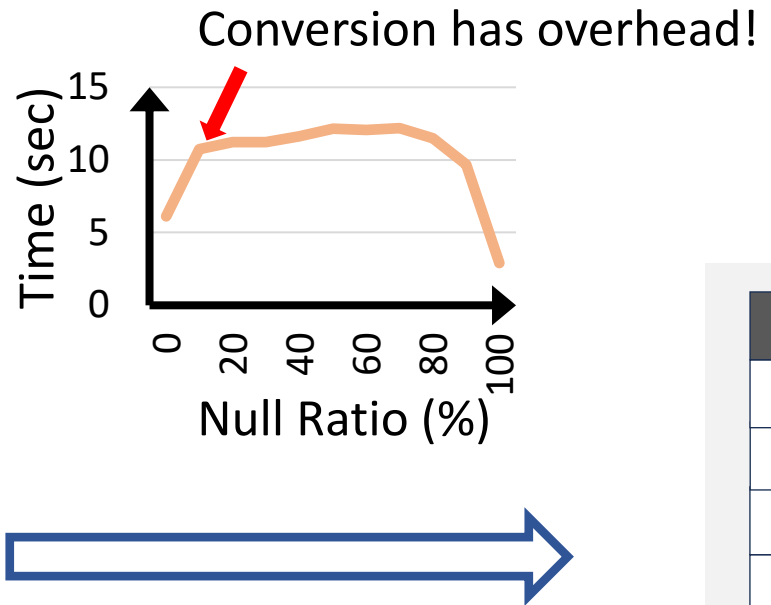
Logical

data
11
NULL
33
44
NULL
66

Parquet

data	valid?
11	1
33	0
44	1
66	1
	0
	1

Compact



RAM

APACHE ARROW

data	valid?
11	1
	0
33	1
44	1
	0
66	1

Placeholder

How Nulls are Represented

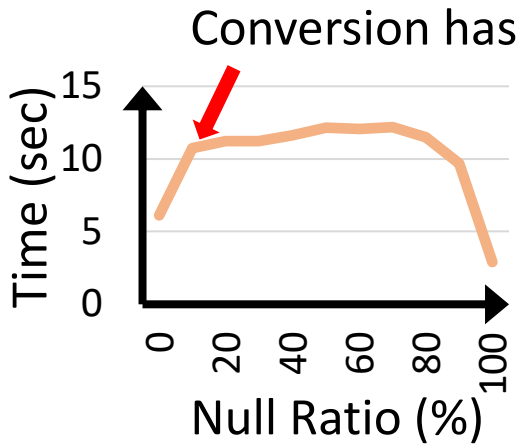
Logical

data
11
NULL
33
44
NULL
66

Parquet

data	valid?
11	1
33	0
44	1
66	1
	0
	1

Compact



S1: Optimize C->P Conversion

RAM

APACHE ARROW

data	valid?
11	1
	0
33	1
44	1
	0
66	1

Placeholder

How Nulls are Represented

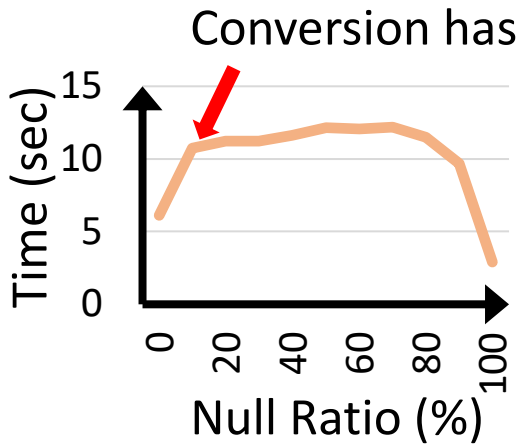
Logical

data
11
NULL
33
44
NULL
66

Parquet

data	valid?
11	1
33	0
44	1
66	1
	0
	1

Compact



S1: Optimize C->P Conversion

RAM

APACHE ARROW

data	valid?
11	1
	0
33	1
44	1
	0
66	1

Placeholder

S2: Use Placeholder for file format

Outline

S1: Optimize C->P Conversion

S2: Align the layout of in-memory and file format

⇨ Can Nulls in file formats be Placeholder?

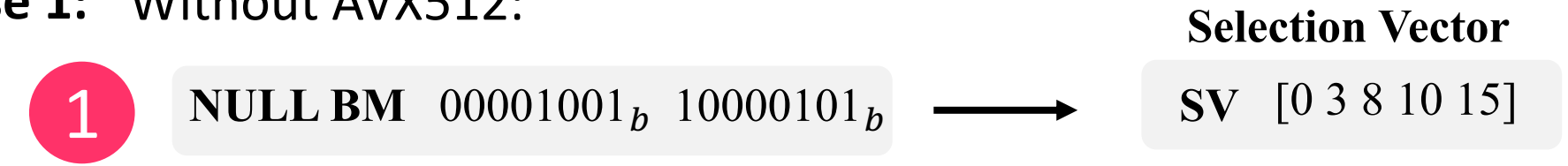
⇨ Can Nulls in in-memory formats be Compact?

S1: Optimizing the C → P Conversion

Case 1: Without AVX512:

S1: Optimizing the C → P Conversion

Case 1: Without AVX512:



- two existing solutions: *optimized scalar* and *SIMD-based methods* (used in Roaring and Velox)

S1: Optimizing the C → P Conversion

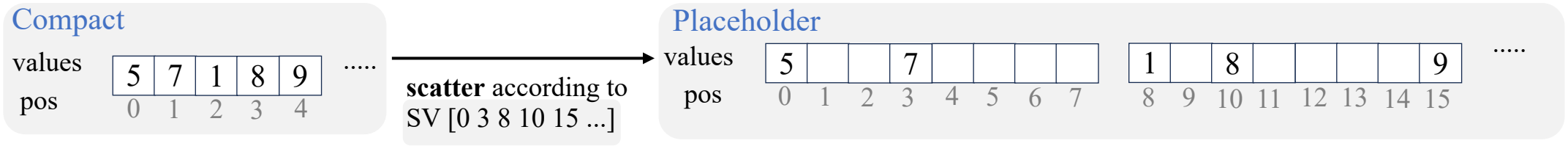
Case 1: Without AVX512:

1

NULL BM 00001001_b 10000101_b ⟶ **Selection Vector** **SV** [0 3 8 10 15]

- two existing solutions: *optimized scalar* and *SIMD-based methods* (used in Roaring and Velox)

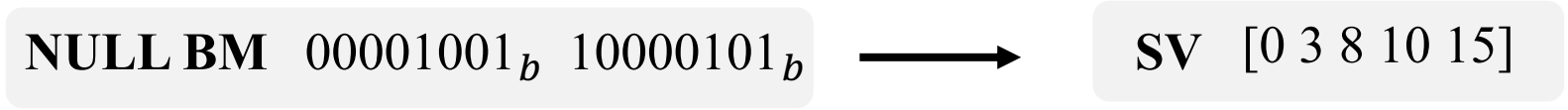
2



S1: Optimizing the C → P Conversion

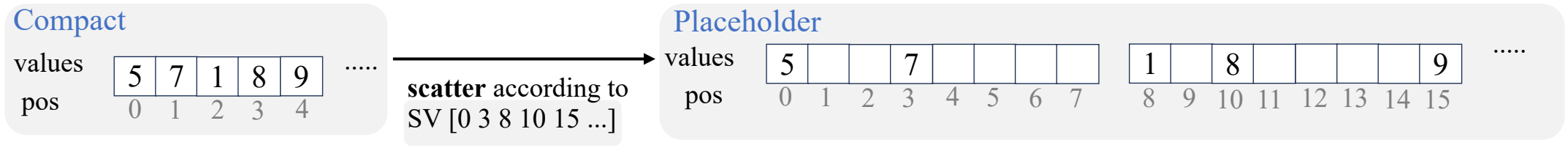
Case 1: Without AVX512:

1



- two existing solutions: *optimized scalar* and *SIMD-based methods* (used in Roaring and Velox)

2



We optimize the implementations:

- For the scalar version, fuse the two steps
- For the SIMD version, reduce branch overhead in scatter by batching

S1: Optimizing the C → P Conversion

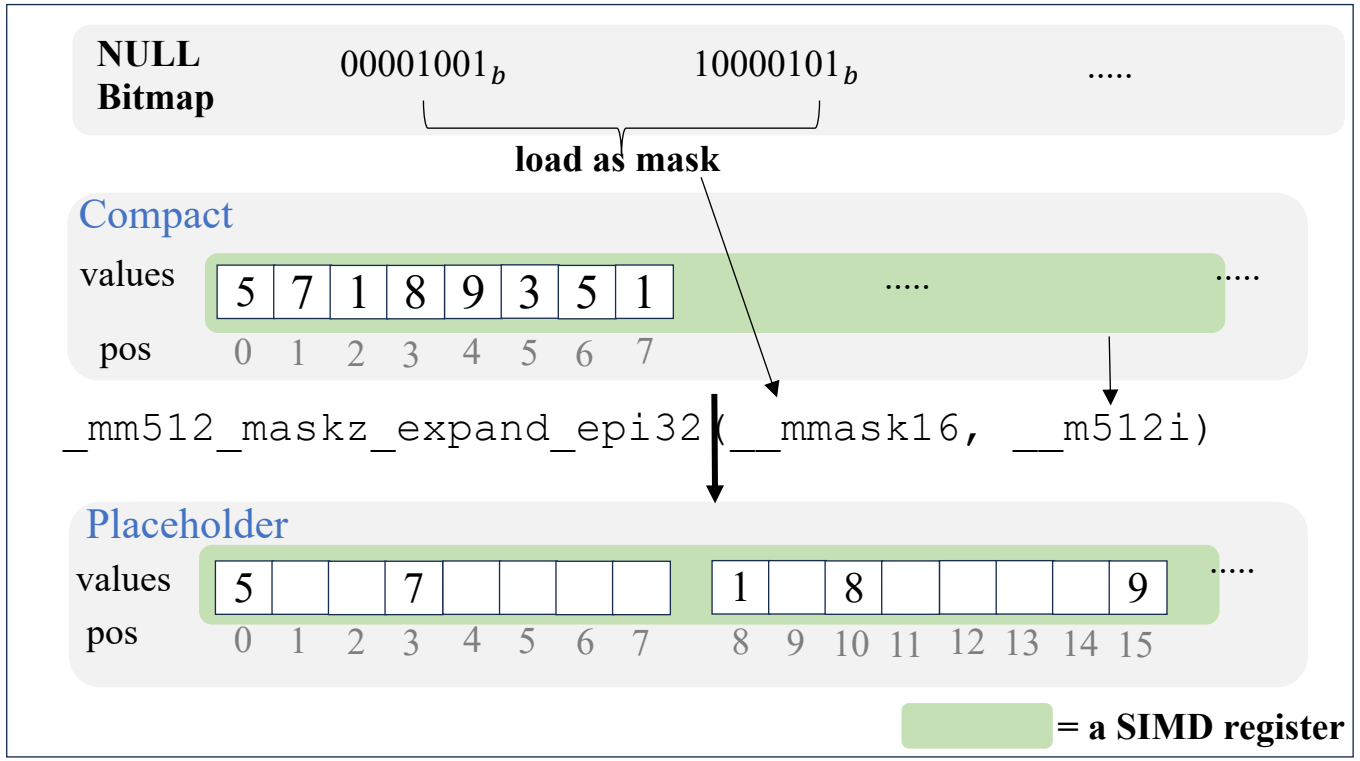
Case 2: With AVX512:

- can directly use `VPEXPANDD` to finish the conversion

S1: Optimizing the C → P Conversion

Case 2: With AVX512:

- can directly use `VPEXPANDD` to finish the conversion



S1: Optimizing the C → P Conversion

Case 2: With AVX512:

- can directly use `VPEXPANDD` to finish the conversion

S1: Optimizing the C → P Conversion

Case 2: With AVX512:

- can directly use `VPEXPANDD` to finish the conversion

No need to convert BM → SV first

Much faster than scatter-based methods

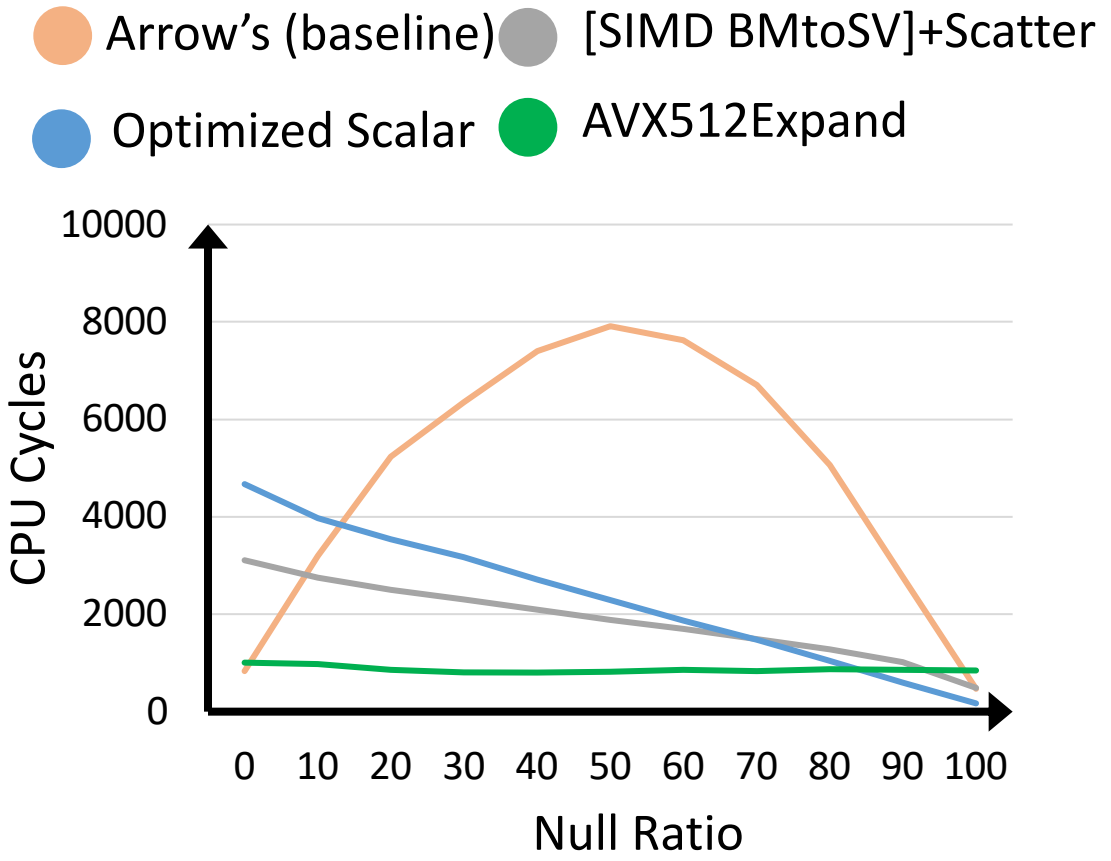
Slower than scatter-based if Null ratio is high

S1: Evaluation

Setup: 2048 int32 values

AVX512Expand is the best in most cases

[SIMD BMtoSV]+Scatter has underutilized SIMD lanes when Null ratio is high



Outline

S1: Optimize C->P Conversion

S2: Align the layout of in-memory and file format

⇨ **Can Nulls in file formats be Placeholder?**

⇨ Can Nulls in in-memory formats be Compact?

What if we also use Placeholder for file formats?

No such C → P Conversion overhead

Less space saving

Can we improve it?

S2 Idea: Smart Placeholder-filling

⇨ The placeholder values carry no meaning

S2 Idea: Smart Placeholder-filling

⇨ The placeholder values carry no meaning

Let’s fill it smartly for better CPR!

S2 Idea: Smart Placeholder-filling

⇨ The placeholder values carry no meaning

Let’s fill it smartly for better CPR!

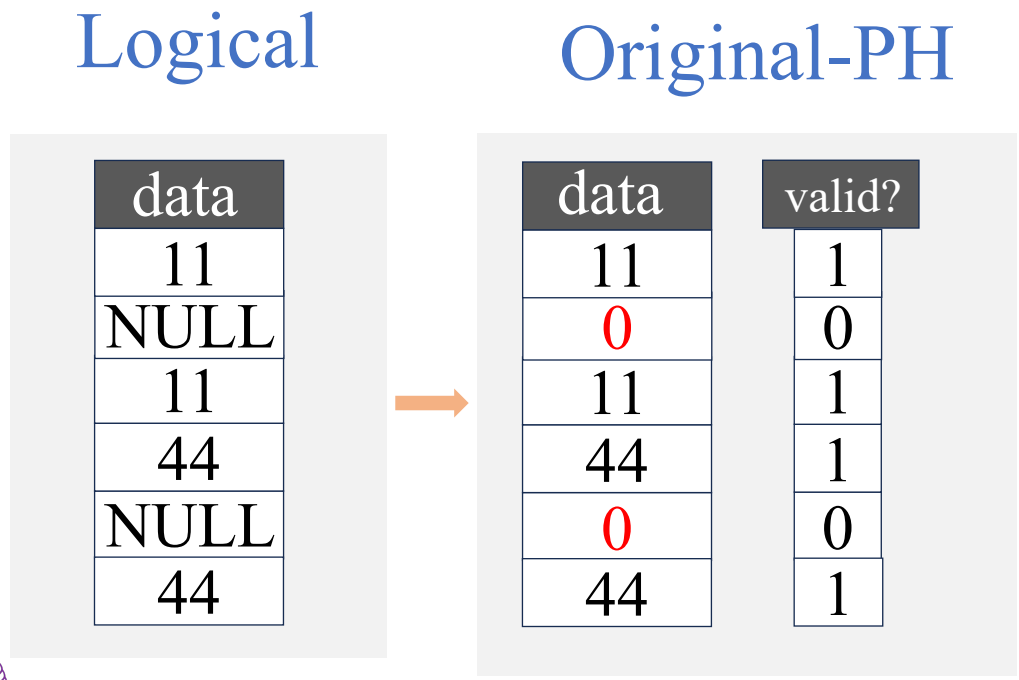
Logical

data
11
NULL
11
44
NULL
44

S2 Idea: Smart Placeholder-filling

⇨ The placeholder values carry no meaning

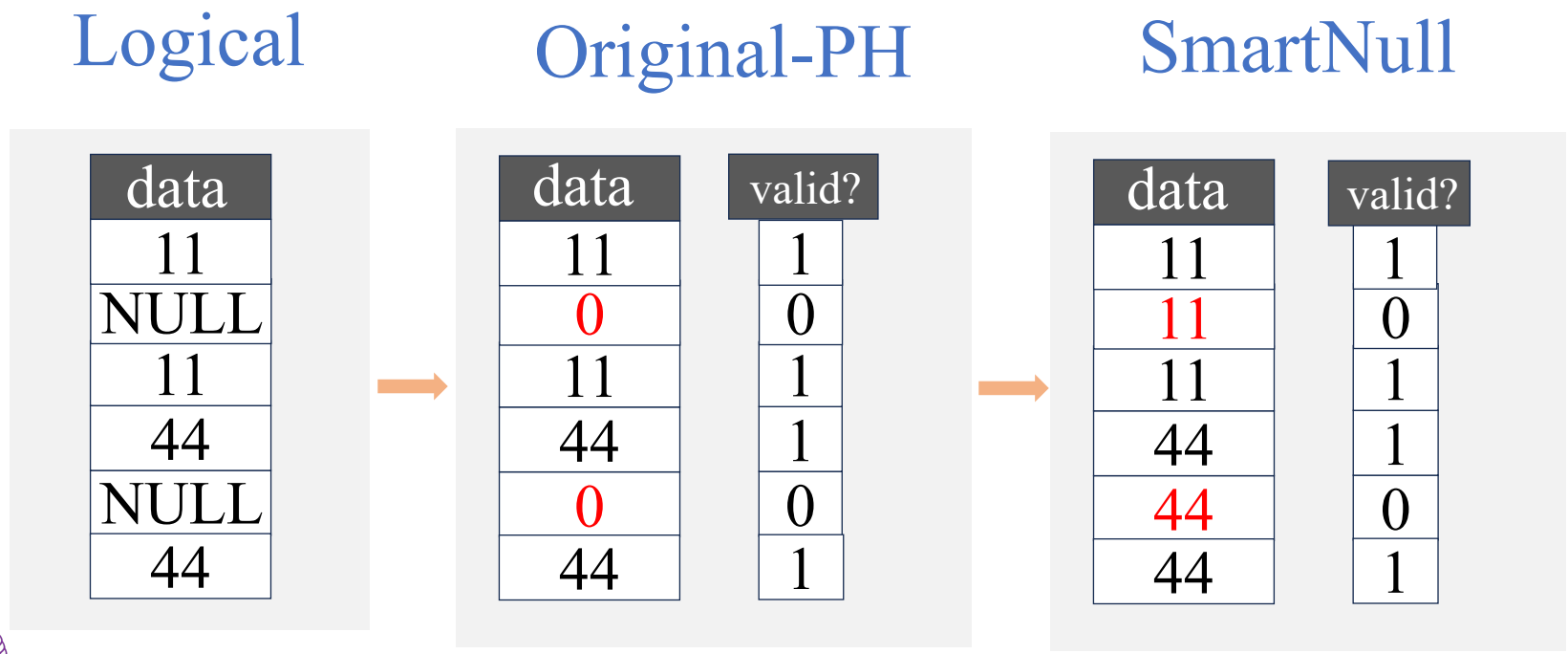
Let's fill it smartly for better CPR!



S2 Idea: Smart Placeholder-filling

The placeholder values carry no meaning

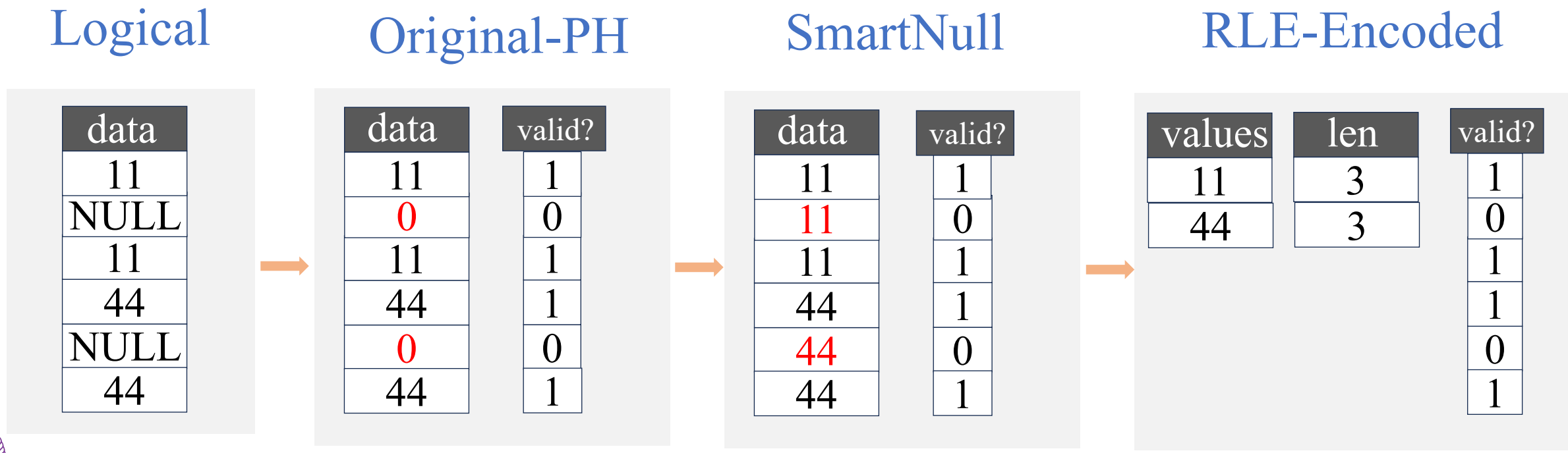
Let's fill it smartly for better CPR!



S2 Idea: Smart Placeholder-filling

⇨ The placeholder values carry no meaning

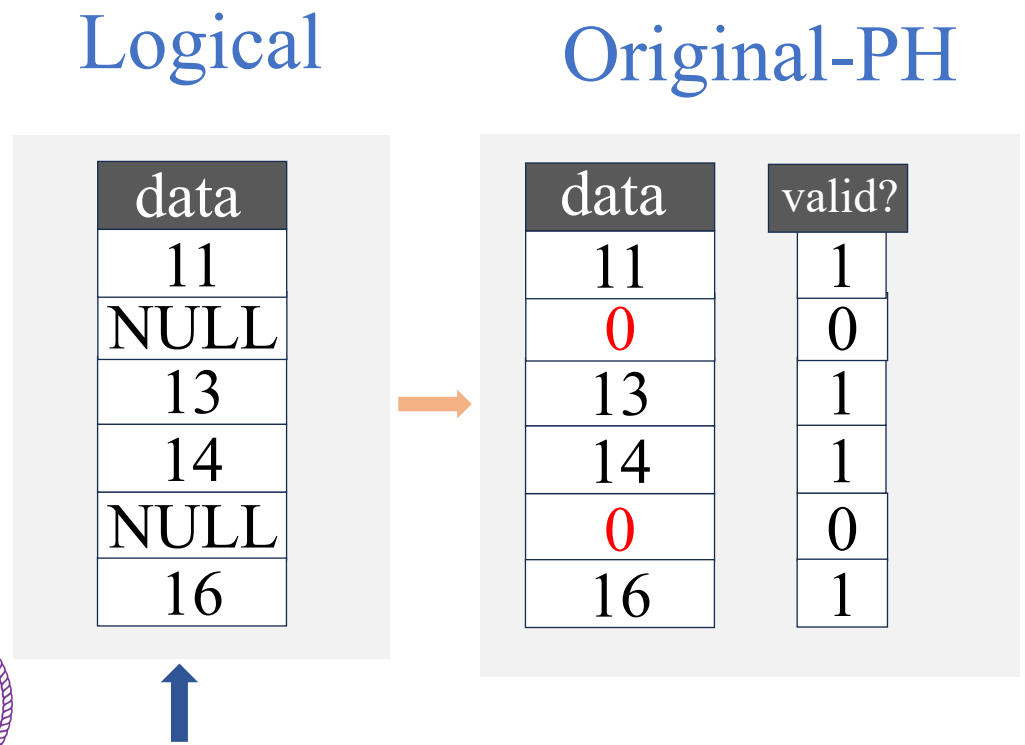
Let's fill it smartly for better CPR!



S2 Idea: Smart Placeholder-filling

⇨ The placeholder values carry no meaning

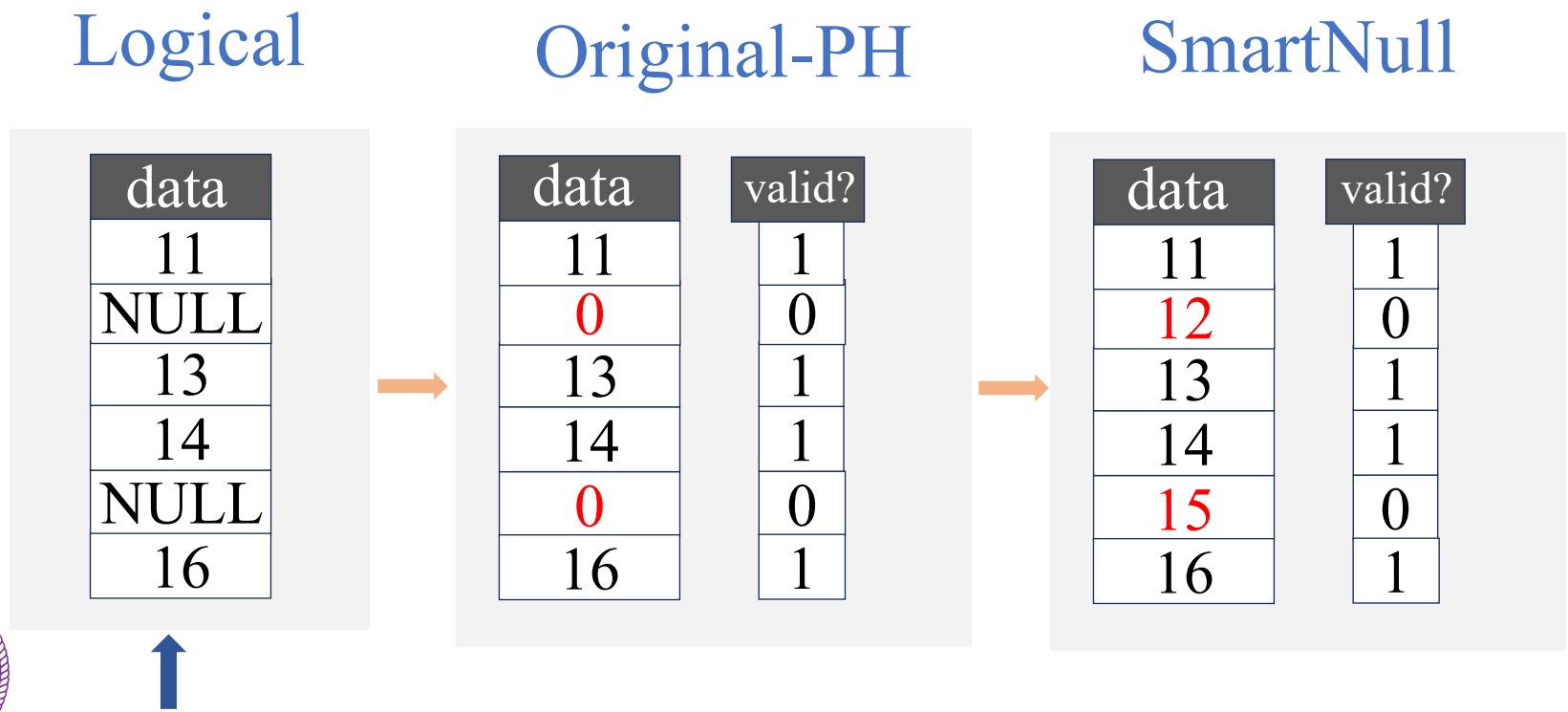
Let's fill it smartly for better CPR!



S2 Idea: Smart Placeholder-filling

⇨ The placeholder values carry no meaning

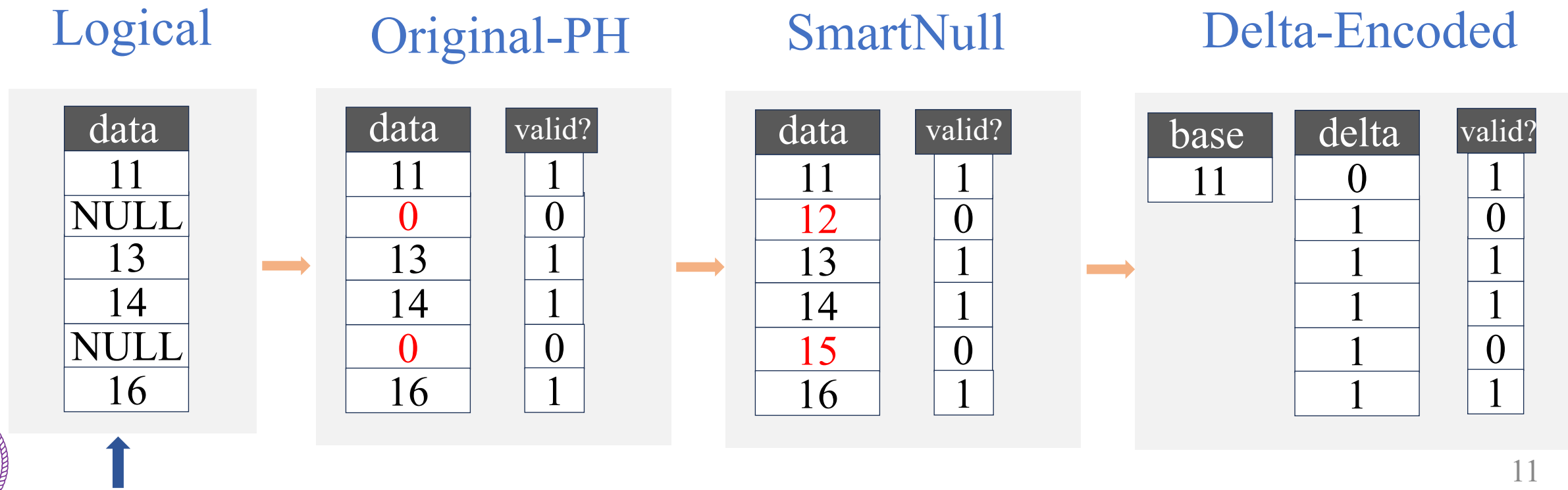
Let's fill it smartly for better CPR!



S2 Idea: Smart Placeholder-filling

⇨ The placeholder values carry no meaning

Let's fill it smartly for better CPR!



S2: Placeholder-filling Strategies

- Zero

- Random

- MostFreq

- LastNonNull

- LinearInterpolation

- **SmartNull** ⇨ piggyback Null-filling to the sampling and encoding phase
  ⇨ match filling strategy with best-fit encoding schemes

S2: Evaluation Setup

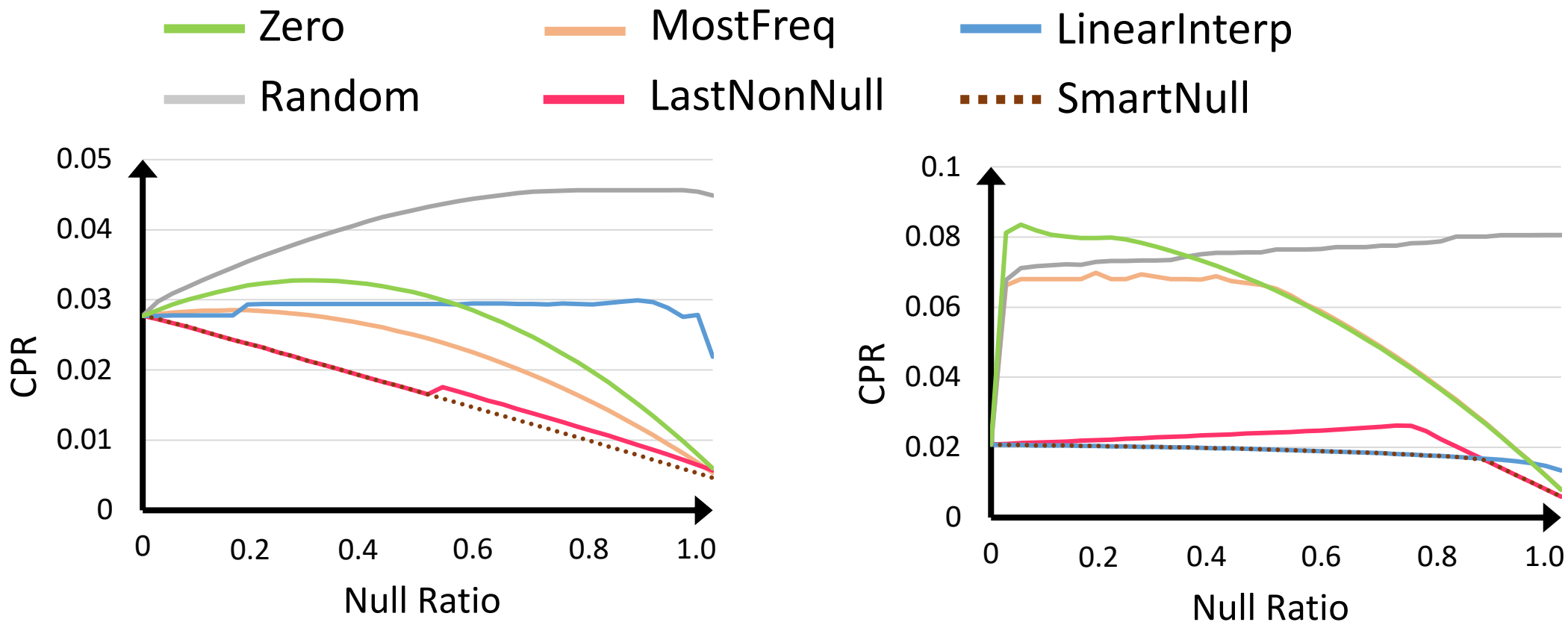
- Datasets:
  - Three from the microbenchmark in our Parquet & ORC evaluation paper

    uniform, gentle_zipf, hotspot
  - One from the “booksale” dataset in SOSD

    serial_correlated
- Varying Null ratio by changing values to Null
- Evaluated in a custom encoding framework similar to ideas in BtrBlocks

S2: Evaluation - Placeholder-filling Strategies

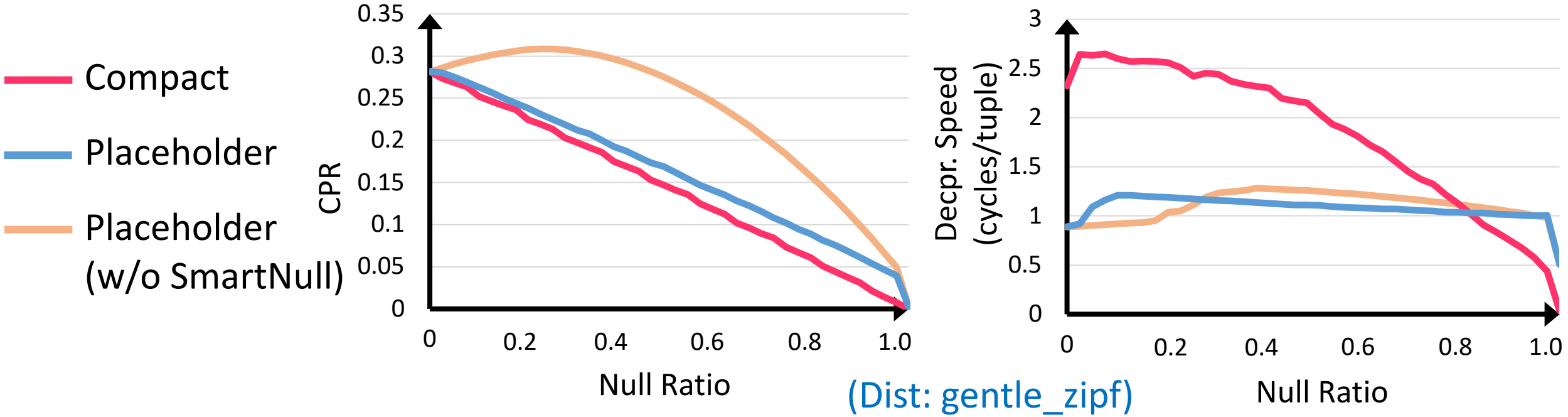
SmartNull appears to be the best across all data distributions



gentle_zipf

serial_correlated

S1 (optimized Conversion) vs. S2 (SmartNull)



(Dist: gentle_zipf)

⇨ Placeholder (w/ SmartNull) offers a better CPR-Speed trade-off for low to medium Null ratio

⇨ Compact excels for sparse data

Outline

S1: Optimize C->P Conversion

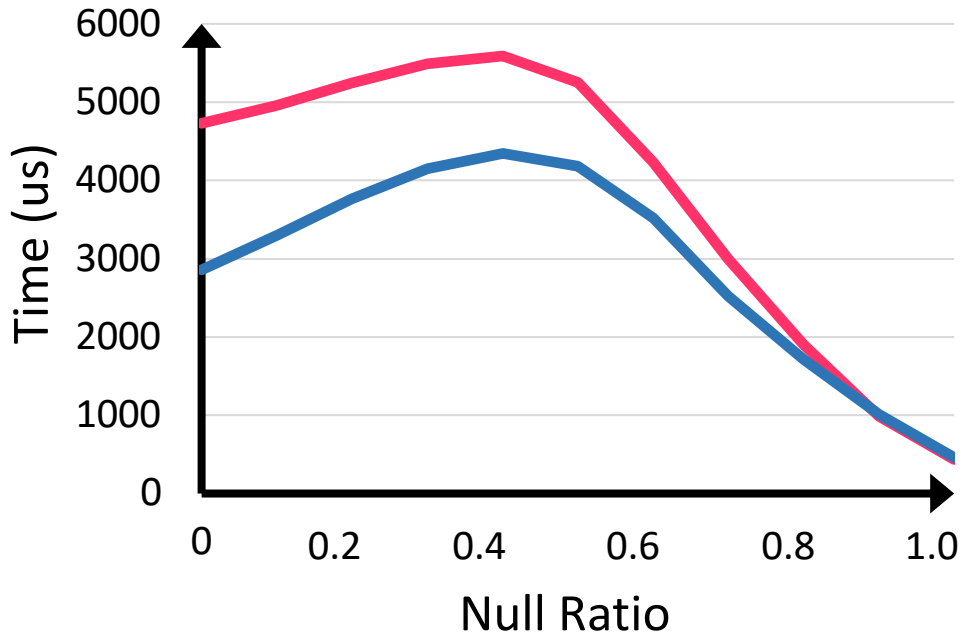
S2: Align the layout of in-memory and file format

⇨ Can Nulls in file formats be Placeholder?

⇨ **Can Nulls in in-memory formats be Compact?**

Can In-memory Formats be Compact?

Setup:
evaluate col_a < col_b
in a vectorized engine



⇨ Compact is slow because of the high random access cost

Takeaways

- Use Placeholder w/ **SmartNull** and Compact w/ **optimized C → P Conversion** in a file format, and choose them wisely
- Stay with Placeholder for in-memory formats
- Source code: https://github.com/XinyuZeng/NULLS

Backup slides

Prior works focus on the Null indicator

- [C-Store, RoaringBitmap]

Bitmap (BM)

valid?
1
0
1
1
0
1

Prior works focus on the Null indicator

- [C-Store, RoaringBitmap]

Bitmap (BM)

valid?
1
0
1
1
0
1

Selection Vector (SV)

row_id
0
2
3
5

Prior works focus on the Null indicator

- [C-Store, RoaringBitmap]

Bitmap (BM)

valid?
1
0
1
1
0
1

Selection Vector (SV)

row_id
0
2
3
5

Ranges

range
[0,1)
[2,4)
[5,6)

Prior works focus on the Null indicator

- [C-Store, RoaringBitmap]

Bitmap (BM)

valid?
1
0
1
1
0
1

Selection Vector (SV)

row_id
0
2
3
5

Ranges

range
[0,1)
[2,4)
[5,6)

How the actual data should be stored is under-discussed

Prior works focus on the Null indicator

- [C-Store, RoaringBitmap]

Bitmap (BM)

valid?
1
0
1
1
0
1

Selection Vector (SV)

row_id
0
2
3
5

Ranges

range
[0,1)
[2,4)
[5,6)

How the actual data should be stored is under-discussed

Let us assume the Null indicator is BM ---- consistent in open standard formats

How Encoders Work

Data Sample

How Encoders Work

Data Sample

data
11
NULL
11
44
NULL
44

How Encoders Work

Data Sample

data
11
NULL
11
44
NULL
44

Original-PH

data	valid?
11	1
0	0
11	1
44	1
0	0
44	1

How Encoders Work

Data Sample

data
11
NULL
11
44
NULL
44

Original-PH

data	valid?
11	1
0	0
11	1
44	1
0	0
44	1

Stats collect

Total runs: 6

Sorted? False

How Encoders Work

Data Sample

data
11
NULL
11
44
NULL
44

Original-PH

data	valid?
11	1
0	0
11	1
44	1
0	0
44	1

Stats collect

Total runs: 6

Sorted? False

Feasible encodings?

RLE ✖

Delta ✖

How SmartNull improves it

Data Sample

How SmartNull improves it

Data Sample

data
11
NULL
11
44
NULL
44

How SmartNull improves it

Data Sample

data
11
NULL
11
44
NULL
44

Our encoder's **View**
(no need to fill the Nulls)

data	valid?
11	1
11	0
11	1
44	1
44	0
44	1

How SmartNull improves it

Data Sample

data
11
NULL
11
44
NULL
44

Our encoder's **View**
(no need to fill the Nulls)

data	valid?
11	1
11	0
11	1
44	1
44	0
44	1

Stats collect

Total runs: 2

Sorted? True

How SmartNull improves it

Data Sample

data
11
NULL
11
44
NULL
44

Our encoder's **View**
(no need to fill the Nulls)

data	valid?
11	1
11	0
11	1
44	1
44	0
44	1

Stats collect

Total runs: 2

Sorted? True

Feasible encodings?

RLE +

Delta +

Detour: Special Value

Detour: Special Value

Logical

data
0
NULL
11
44
NULL
44

Detour: Special Value

Logical

data
0
NULL
11
44
NULL
44

Special Value

data
0
45
11
44
45
44

Sp.Val
45

Detour: Special Value

Logical

data
0
NULL
11
44
NULL
44

Special Value

data
0
45
11
44
45
44

Sp.Val
45

No extra bitmap

Detour: Special Value

Logical

data
0
NULL
11
44
NULL
44

Special Value

data
0
45
11
44
45
44

Sp.Val
45

No extra bitmap

CPR of actual data is similar to Placeholder w/o SmartNull

Detour: Special Value

Logical

data
0
NULL
11
44
NULL
44

Special Value

data
0
45
11
44
45
44

Sp.Val
45

No extra bitmap

CPR of actual data is similar to Placeholder w/o SmartNull

Have to fallback to Placeholder if all values are used up (e.g., Int8)

Can In-memory Formats be Compact?

Setup: evaluate col_a < col_b in a vectorized engine

Can In-memory Formats be Compact?

Setup: evaluate col_a < col_b in a vectorized engine

Col A

data
11
NULL
33
44
NULL
66

< ?

Col B

data
33
5
NULL
7
NULL
95

Can In-memory Formats be Compact?

Setup: evaluate col_a < col_b in a vectorized engine

Col A

data
11
NULL
33
44
NULL
66

< ?

Col B

data
33
5
NULL
7
NULL
95

"Active" rows - [DaMoN 2021]

BM

active?
1
1
0
1
1
0

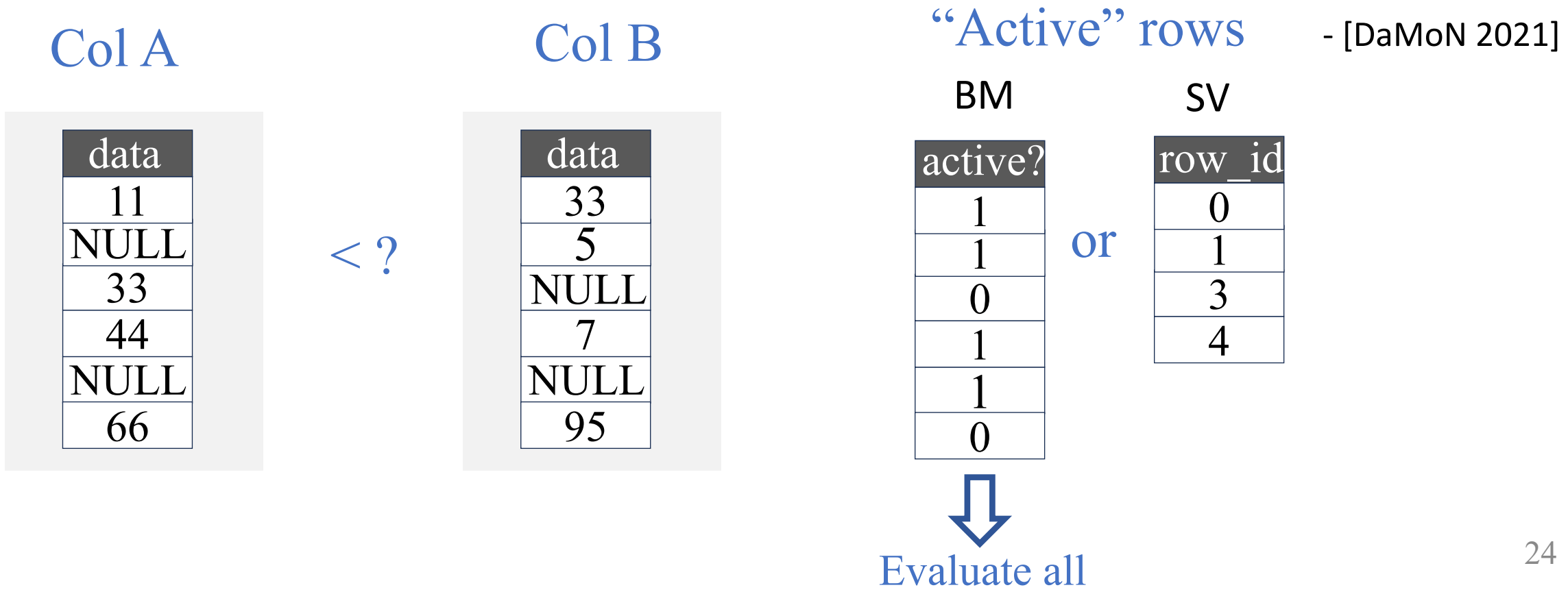
or

SV

row_id
0
1
3
4

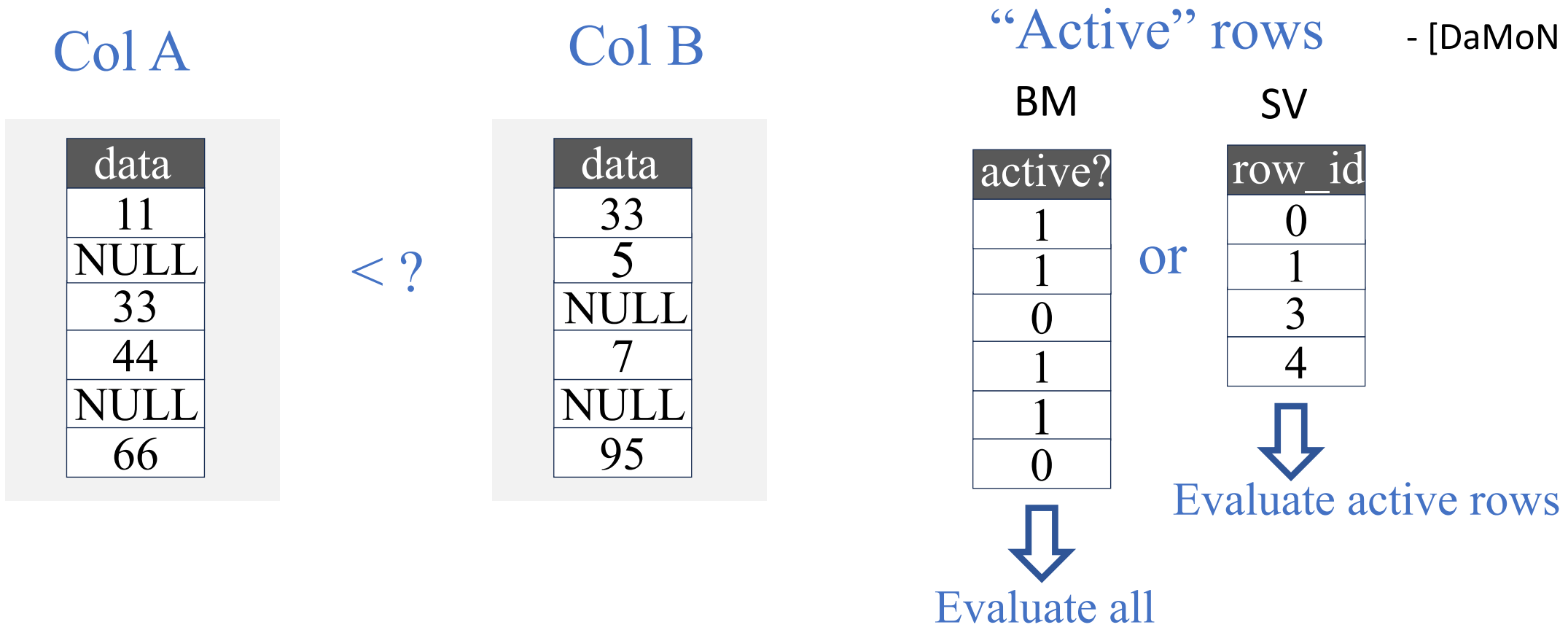
Can In-memory Formats be Compact?

Setup: evaluate col_a < col_b in a vectorized engine



Can In-memory Formats be Compact?

Setup: evaluate col_a < col_b in a vectorized engine



Evaluation: Compact cannot Compete

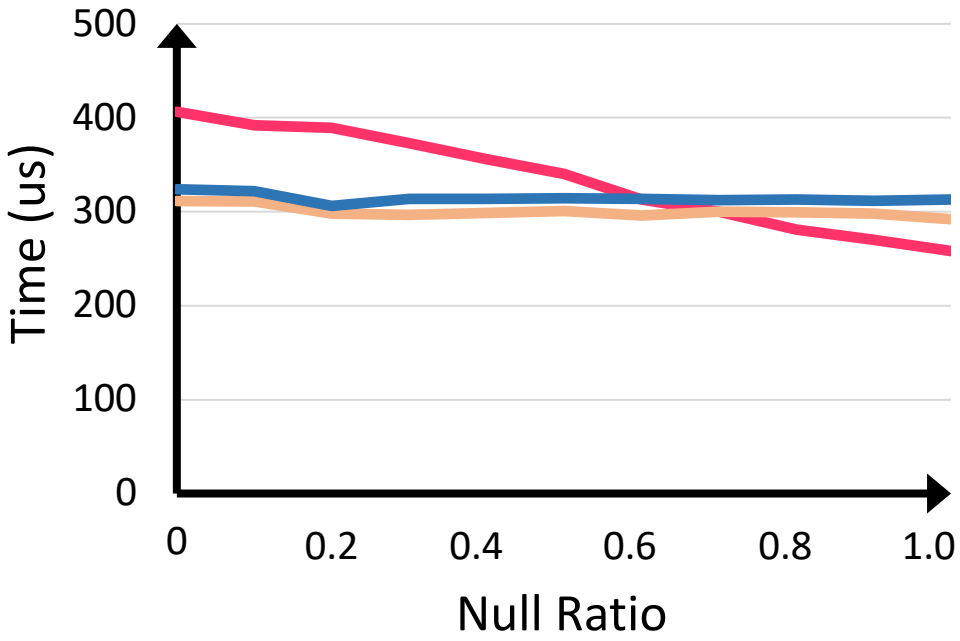
1M values

Compact

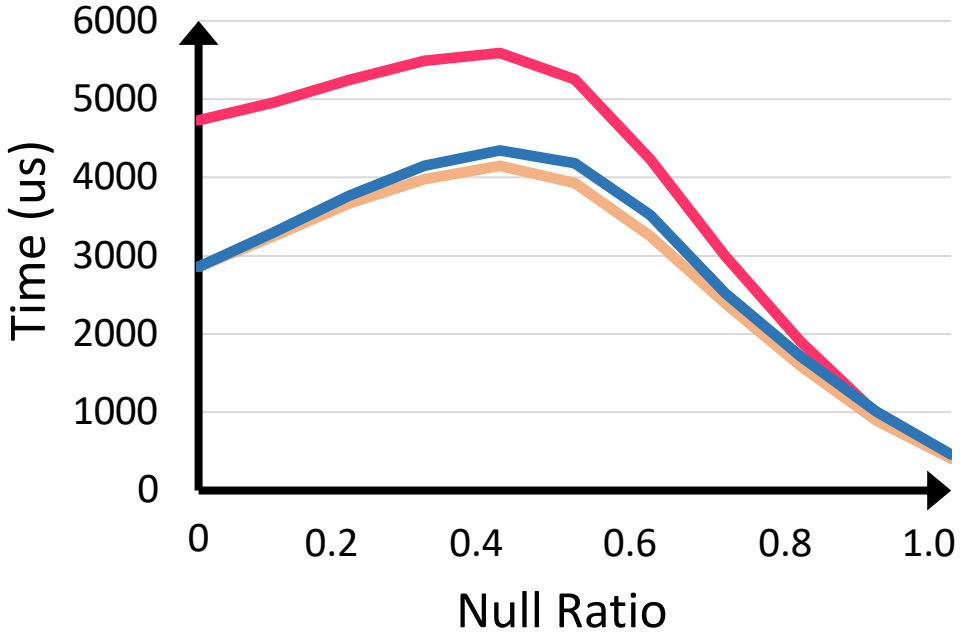
Placeholder

SpecialVal

Evaluate all



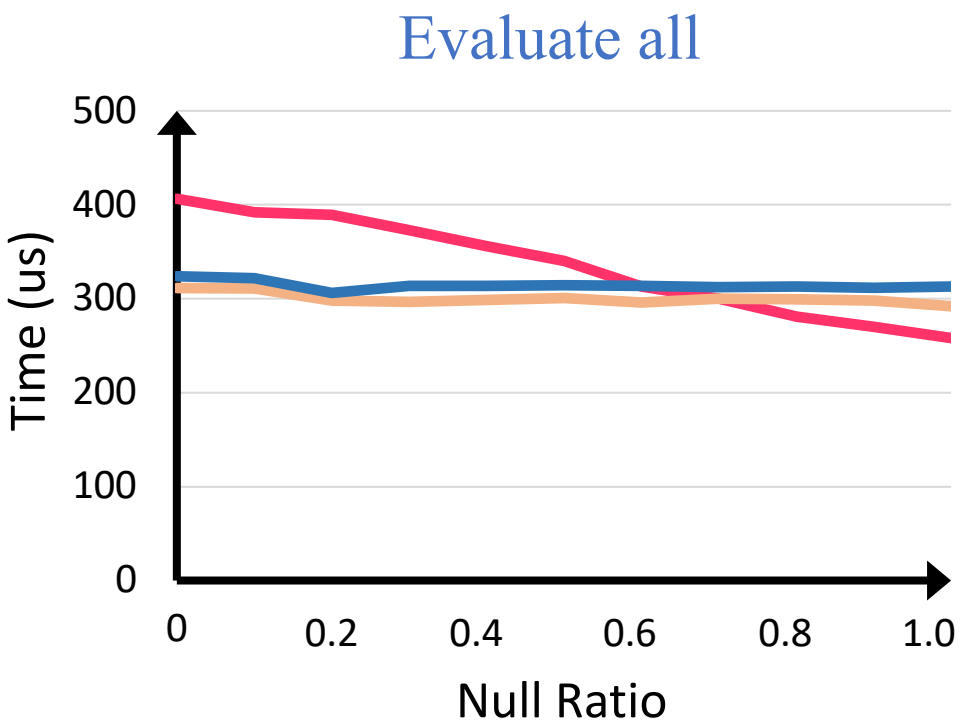
Evaluate active rows

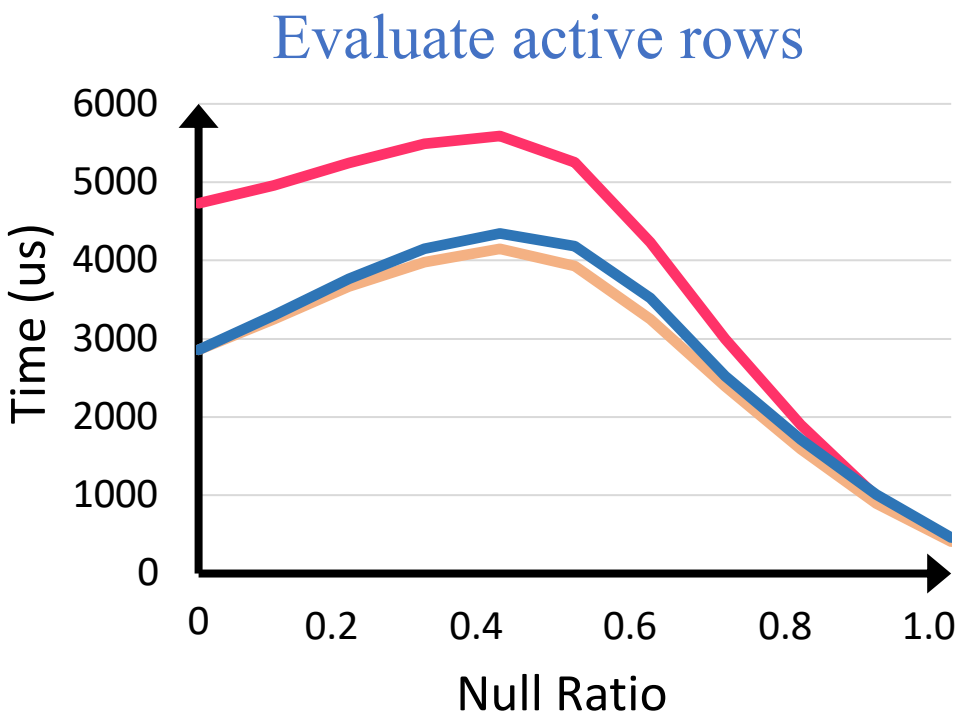


Evaluation: Compact cannot Compete

1M values

Compact
Placeholder
SpecialVal





⇨ Compact is only the best when Null Ratio is high w/ “Evaluate All”

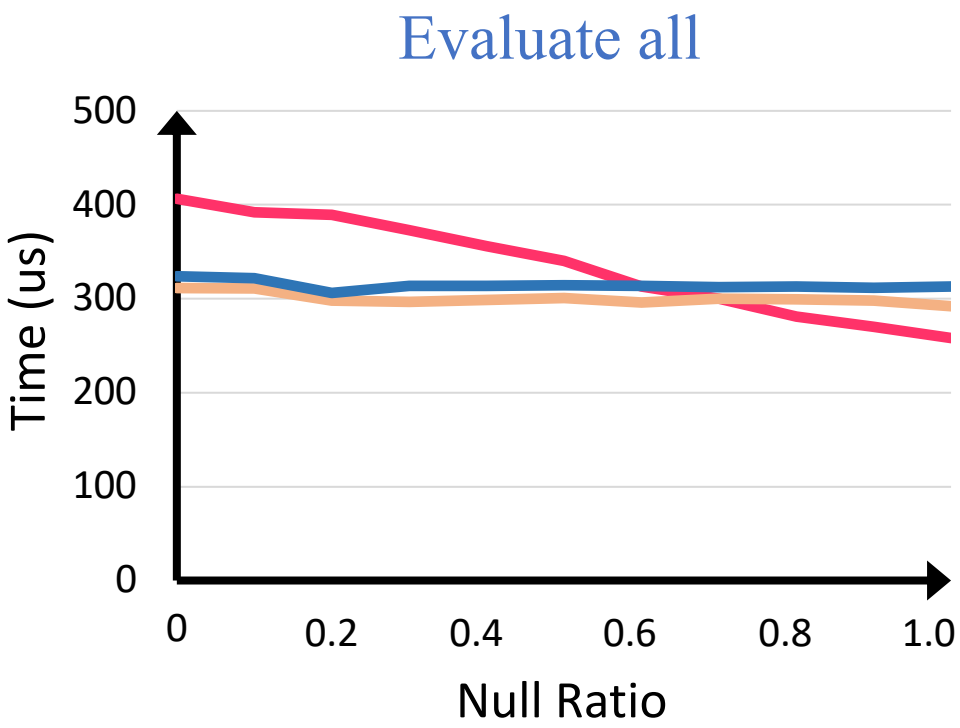
Evaluation: Compact cannot Compete

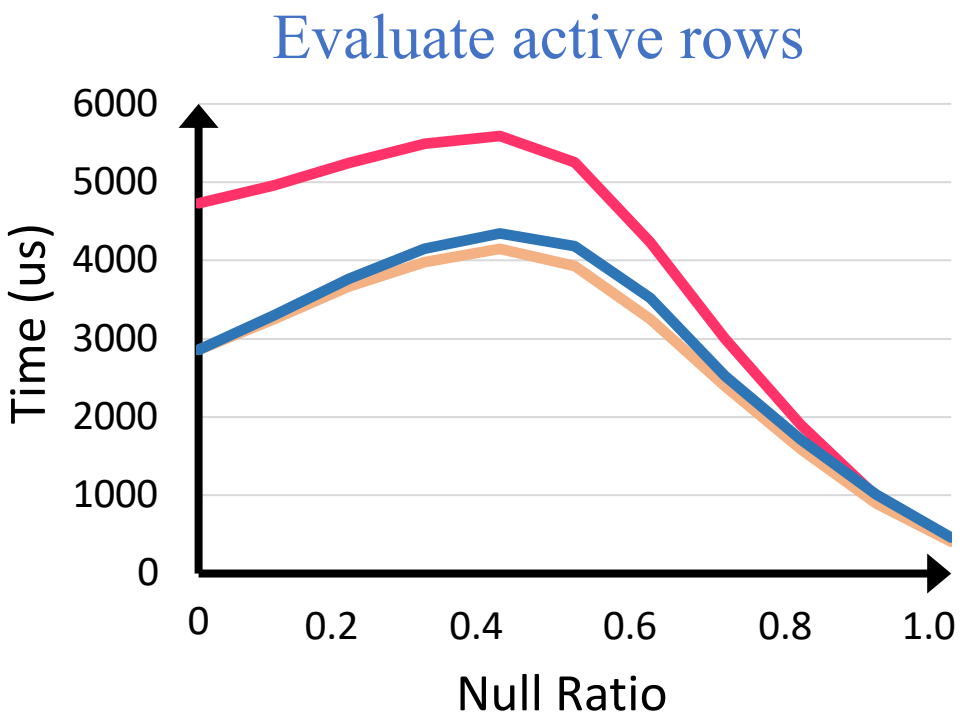
1M values

Compact

Placeholder

SpecialVal





⇨ Compact is only the best when Null Ratio is high w/ “Evaluate All”

⇨ SpecialVal is slightly better than Placeholder