Relational Fabric:

[A Tale of Software and Hardware Synergy for]

Transparent Data Transformation

Tarikul Islam Papon Ju Hyoung Mun Shahin Roozkhosh Denis Hoornaert

Ahmed Sanaullah Ulrich Drepper Renato Mancuso **Manos Athanassoulis**

Talk at ICDE 2023

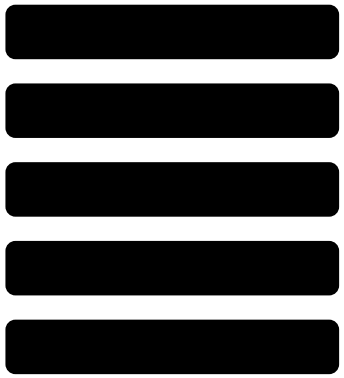
How to stop worrying about data layouts?

Tarikul Islam Papon Ju Hyoung Mun Shahin Roozkhosh Denis Hoornaert
Ahmed Sanaullah Ulrich Drepper Renato Mancuso **Manos Athanassoulis**

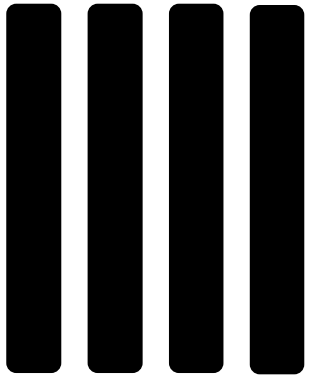
Talk at ICDE 2023

Break the *Fractured Mirrors*!

OLTP



OLAP



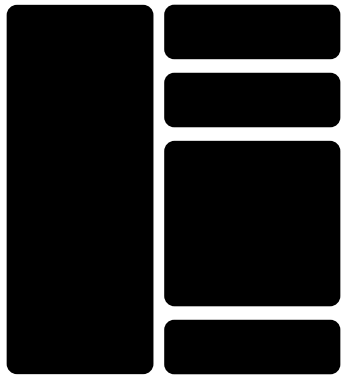
No need to have two systems!

No need to convert data!

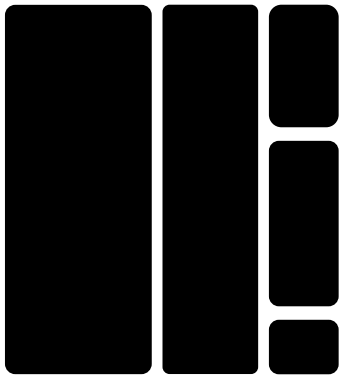
What if we could have
the benefits of both
without storing or maintaining two copies of data?

Bridge the *Archipelago* of Hybrid Layouts

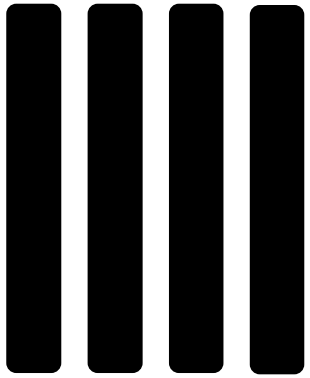
Hybrid



Hybrid



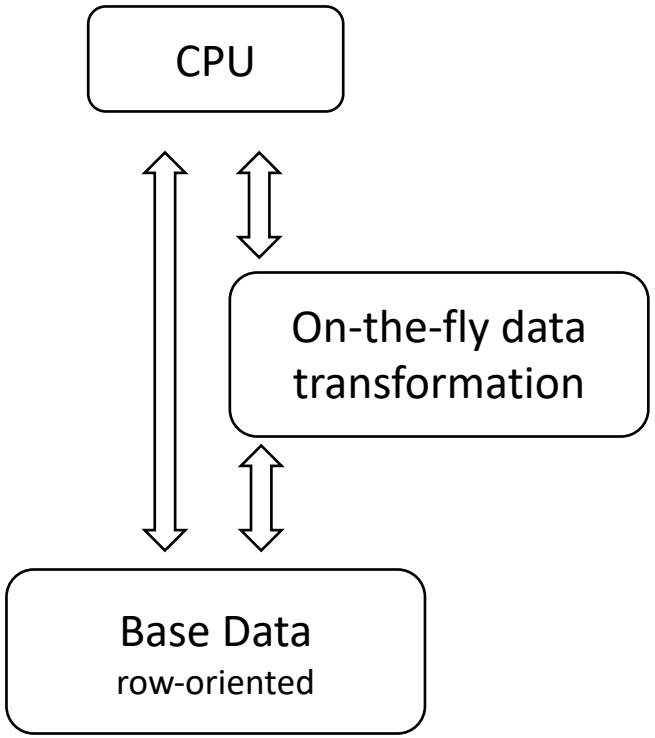
OLAP



What if we could have **the benefits of any data layout?**

For free?

Our vision: Relational Fabric



How? Specialized (Reprogrammable) Hardware

Why now? A new era!

'Moore's Law's dead,' Nvidia CEO Jensen Huang says in justifying gamin

Last Updated: S
First Published: S

By Wallace Witkc

CNBC MARKETS BUSINESS INVESTING TECH POLITICS CNBC TV INVESTING CLUB PRO

TECH

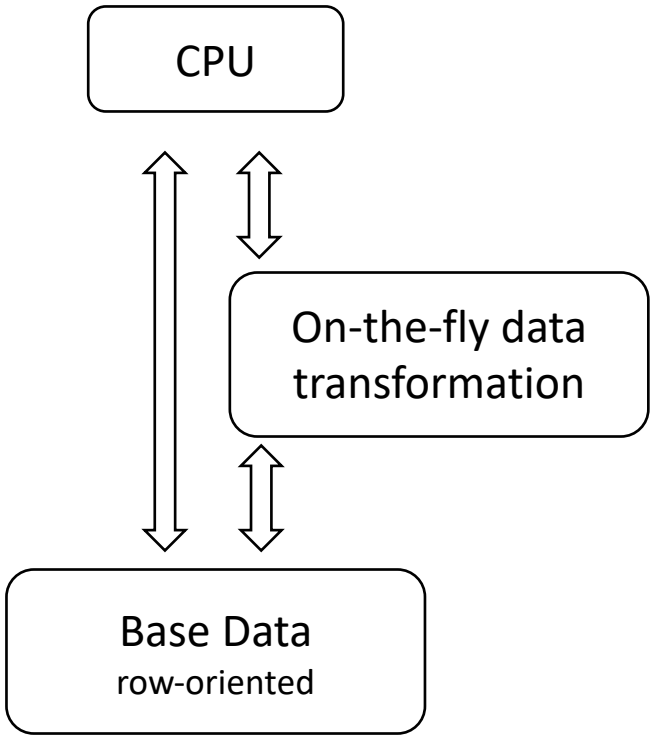
Intel says Moore's Law is still alive and well. Nvidia says it's ended.

PUBLISHED TUE, SEP 27 2022·3:26 PM EDT

Kif Leswing
@KIFLESWING

SHARE

Our vision: Relational Fabric



How? Specialized (Reprogrammable) Hardware

A New Golden Age for Computer Architecture:

Domain-Specific Hardware/Software Co-Design, Enhanced Security, Open Instruction Sets, and Agile Chip Development

John Hennessy and David Patterson
Stanford and UC Berkeley
June 4, 2018

Turing Lecture 2018

Is specialized HW for data management a new idea?

Database Machines (e.g., Gamma)

The concept of database machines became popular during the **1980s**, when it was widely believed that these special-purpose machines would be in widespread use within a decade.

from *Encyclopedia of Information Systems*

International Workshop on Database Machines (IWDM) ran from 1981 to 1989

Q100: Database Processing Unit

Columbia @ IEEE Micro 2015

Hardware/Software co-design from a database perspective

ETHZ @ CIDR 2020

Enzian: an open, general, CPU/FPGA platform for systems software research

ETHZ @ ASPLOS 2022

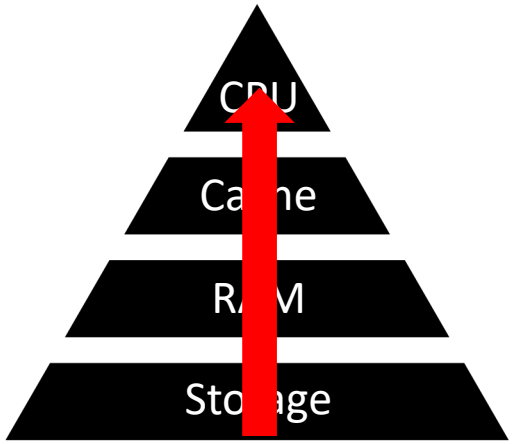
What is different in our approach?

Avoid “too much” specialization!

We ask: *what are common operations that data-intensive systems need?*

data movement

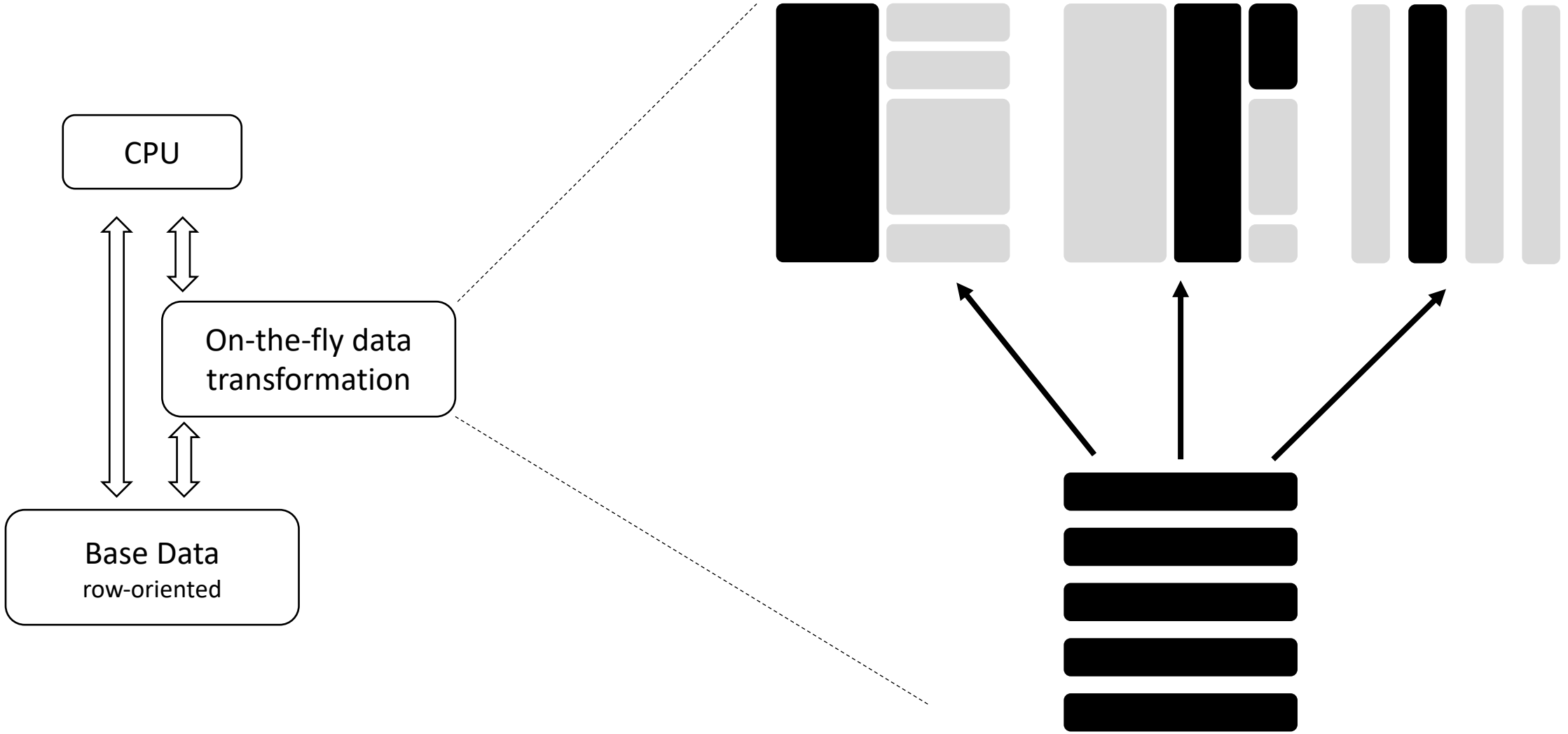
preparing the desired **data layout**



We focus on ***transparent data transformation***

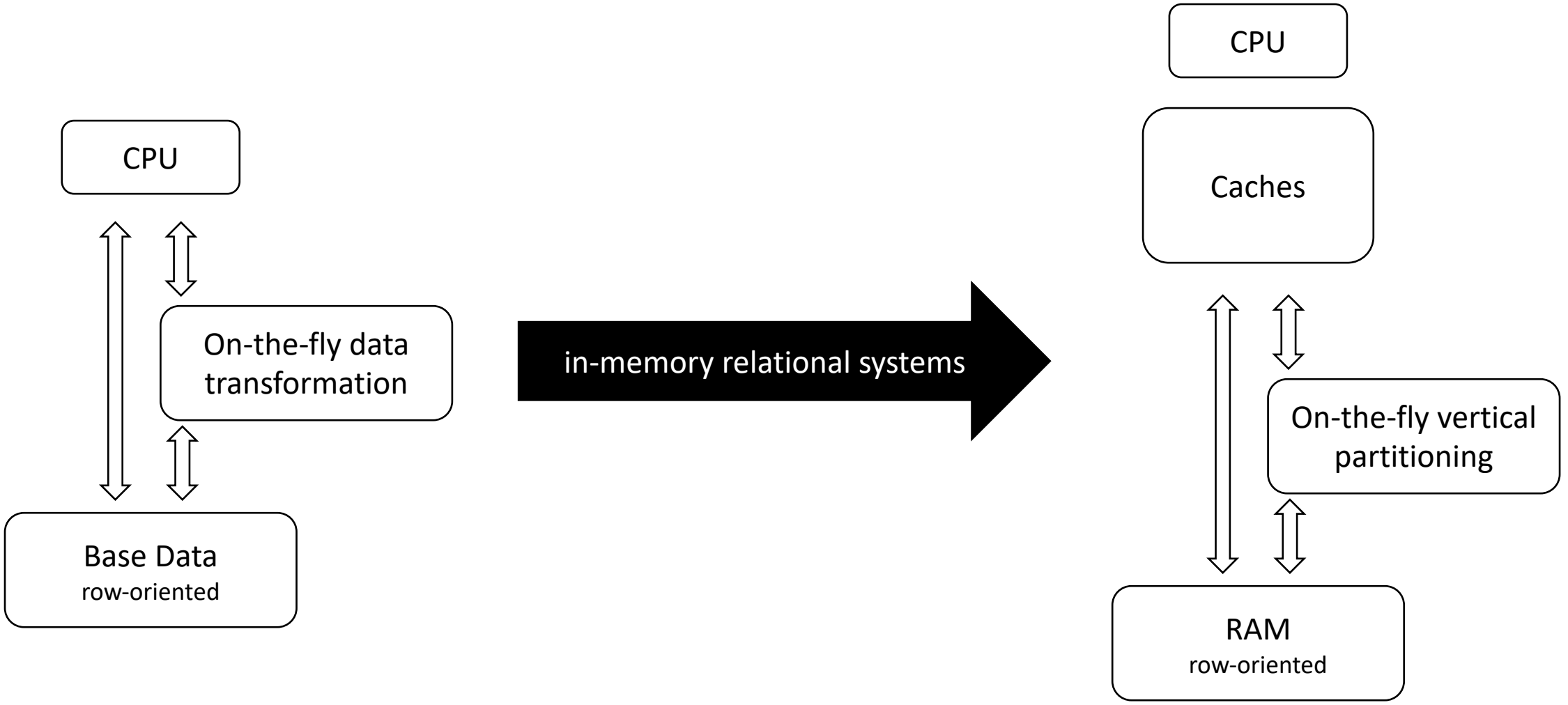
minimal **data movement**
effortless **locality**

Transparent Data Transformation



Iteration 1: **Relational Memory**

Relational Memory: Native In-Memory Accesses on Rows and Columns, ***EDBT 2023***



Iteration 1: Relational Memory

CPU

Caches

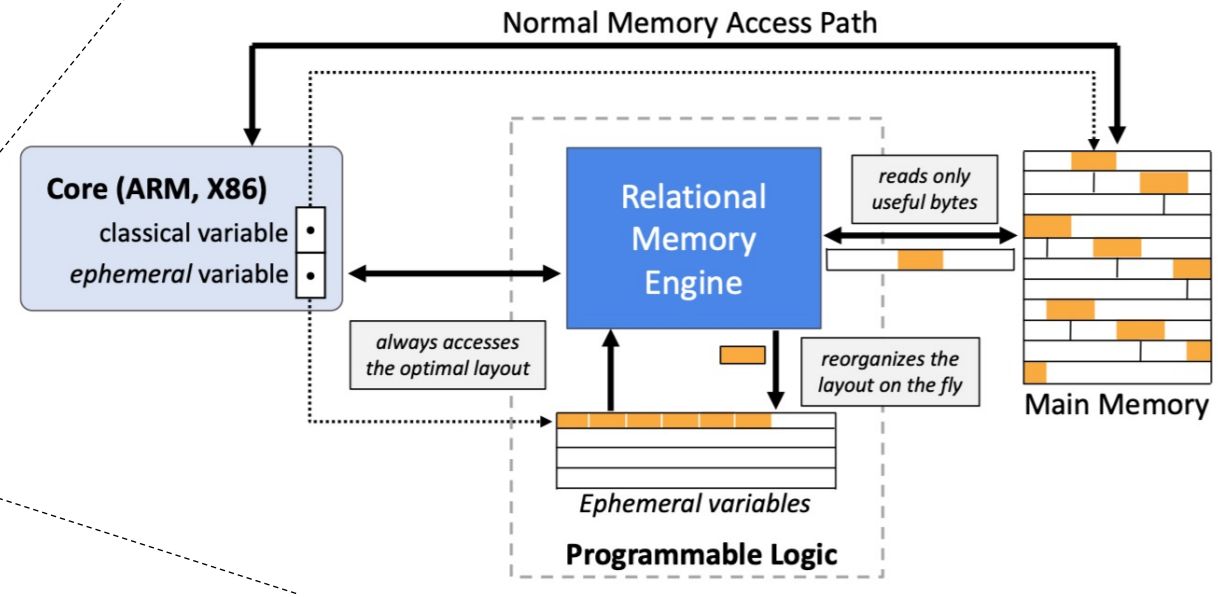
On-the-fly vertical partitioning

RAM
row-oriented

Q1: how to build?

Q2: how to use?

Relational Memory

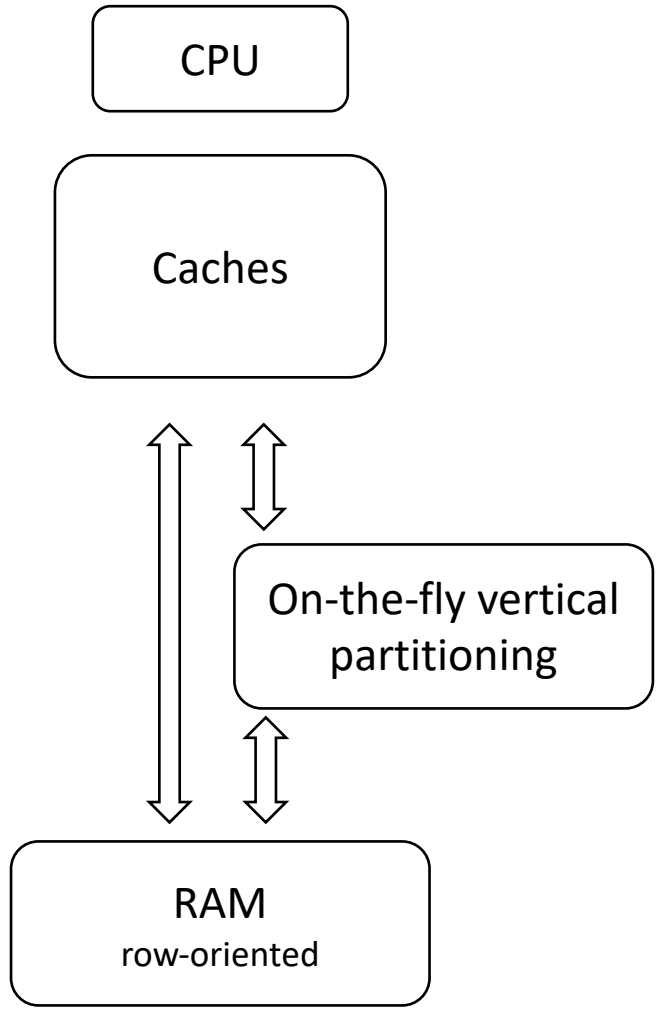


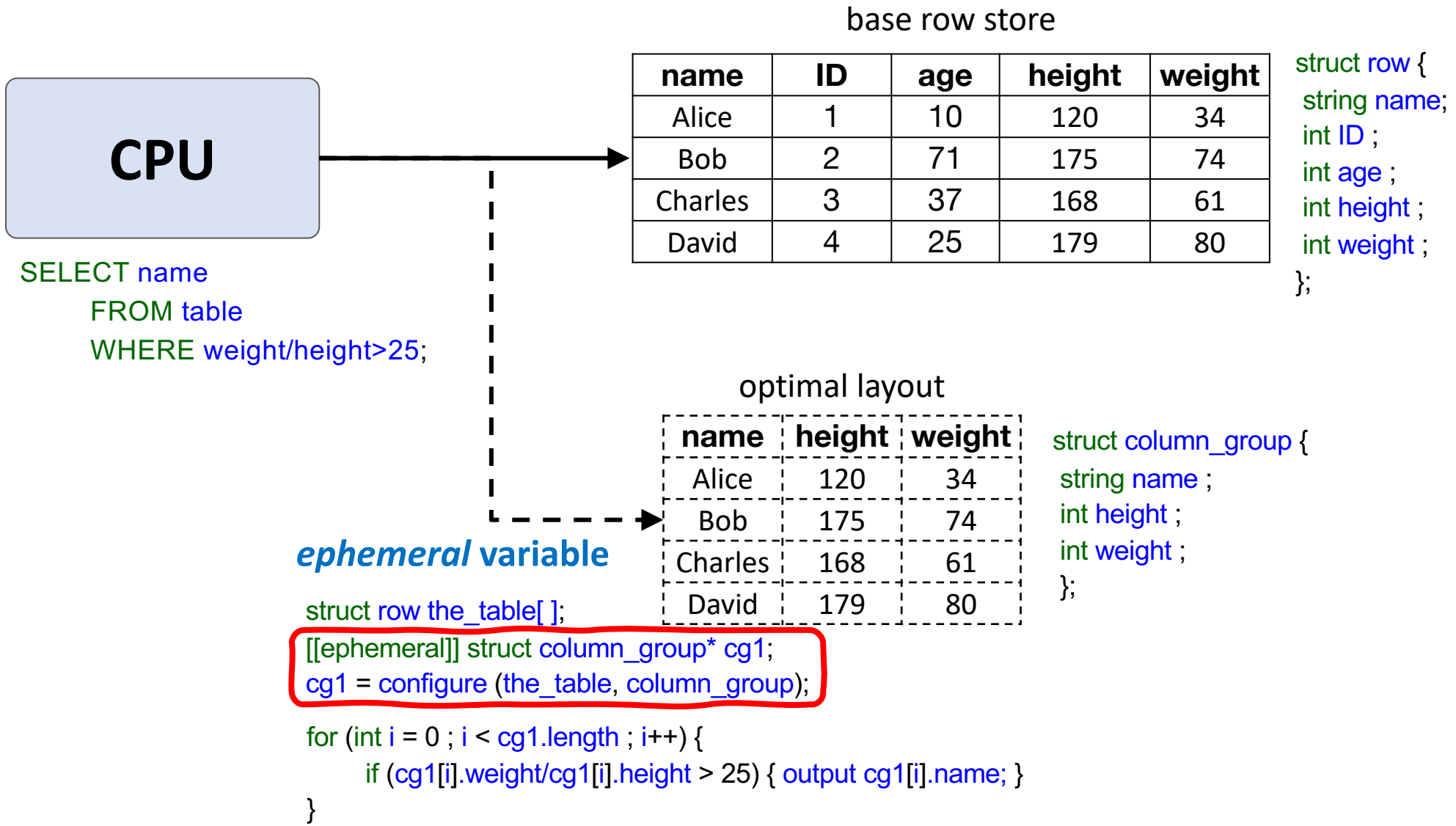
leads to normal memory accesses

```
variable table[ ];
[[ephemeral]] variable col_group[ ];
```

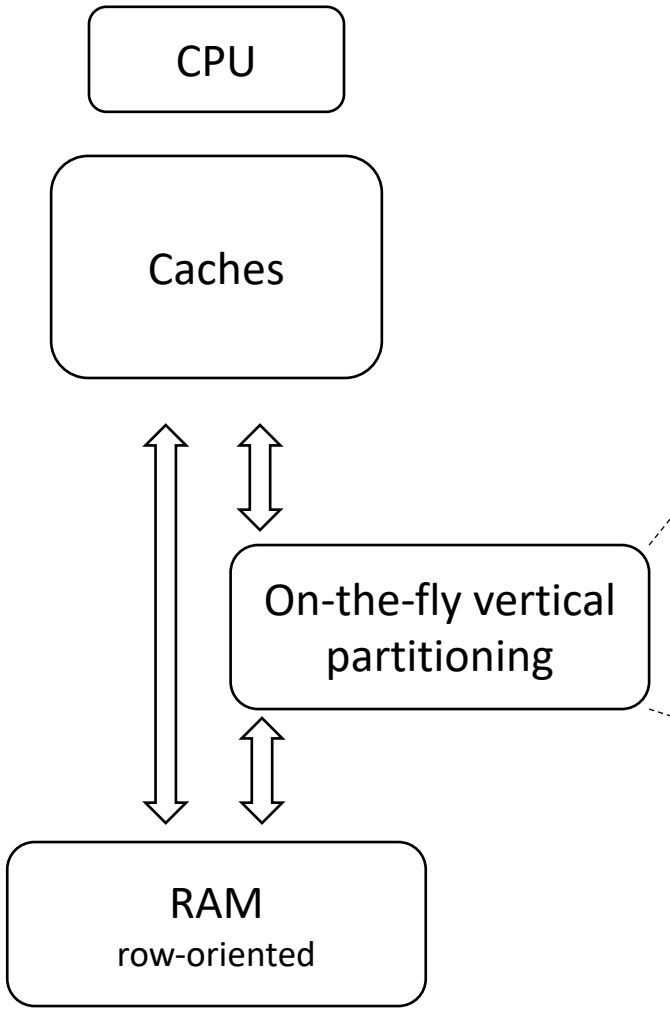
”fake” address that is intercepted by RME, but CPU thinks it exists

ephemeral variables




base row store
CPU
name ID age height weight
Alice 1 10 120 34
Bob 2 71 175 74
Charles 3 37 168 61
David 4 25 179 80
struct row {
string name;
int ID ;
int age ;
int height ;
int weight ;
};
SELECT name
FROM table
WHERE weight/height>25;
optimal layout
name height weight
Alice 120 34
Bob 175 74
Charles 168 61
David 179 80
struct column_group {
string name ;
int height ;
int weight ;
};
ephemeral variable
struct row the_table[ ];
[[ephemeral]] struct column_group* cg1;
cg1 = configure (the_table, column_group);
for (int i = 0 ; i < cg1.length ; i++) {
if (cg1[i].weight/cg1[i].height > 25) { output cg1[i].name; }
}

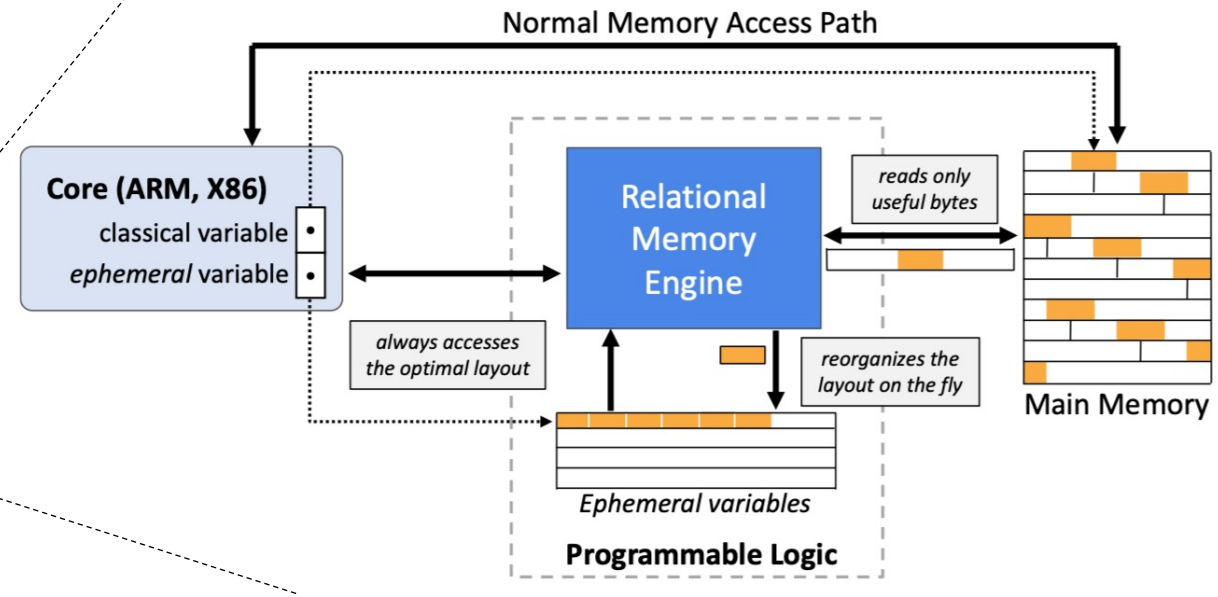
Iteration 1: Relational Memory



Q1: how to build?

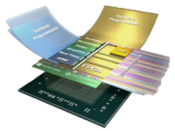
Q2: how to use?

Relational Memory



tightly integrated

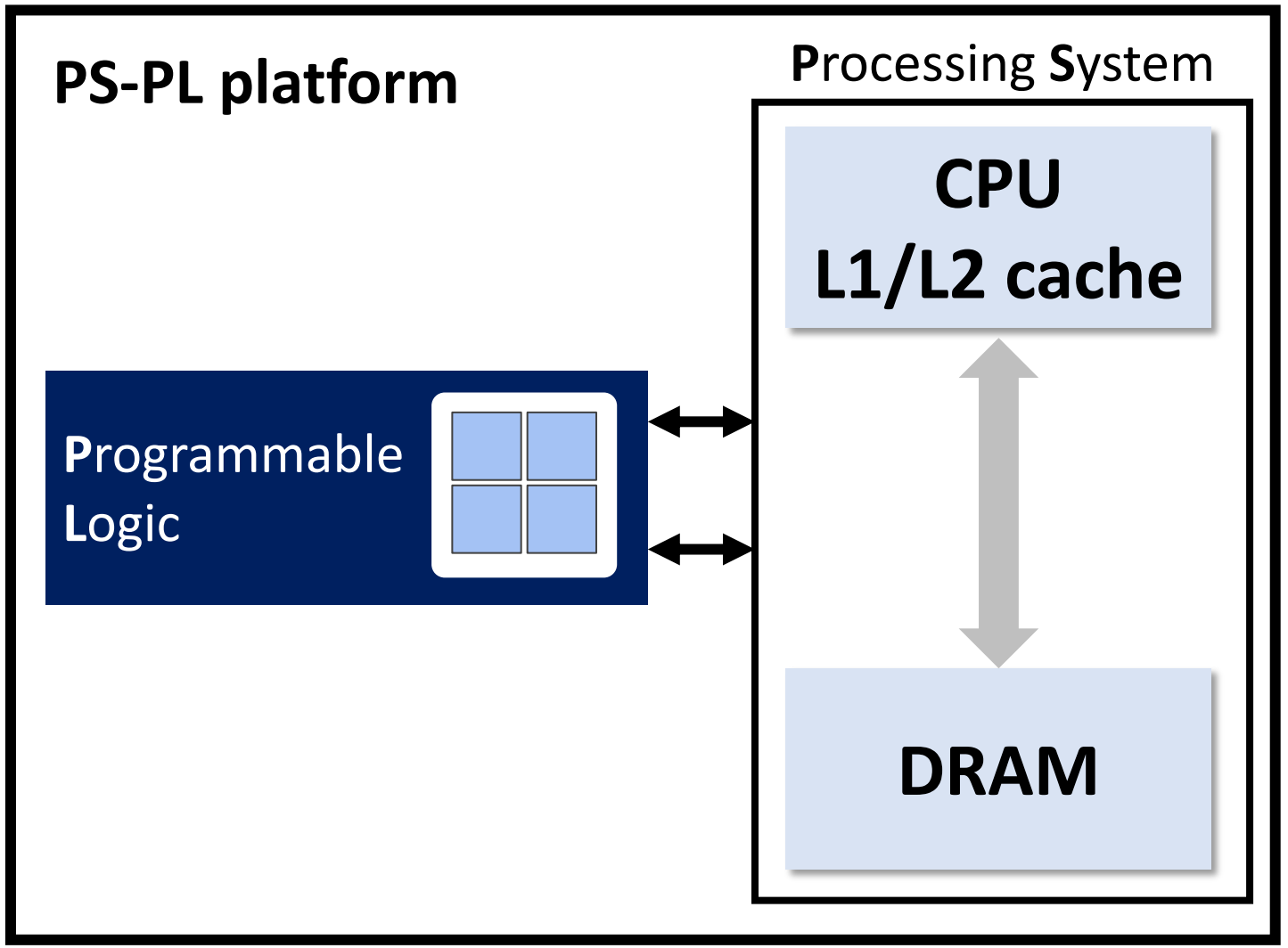
high-performance interfaces



AMD XILINX
UltraScale+

ENZIAN

intel AGILEX



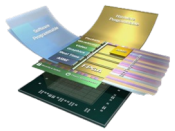
programmable logic in the middle

also known as "bump in the wire" in some communities

tightly integrated

high-performance interfaces

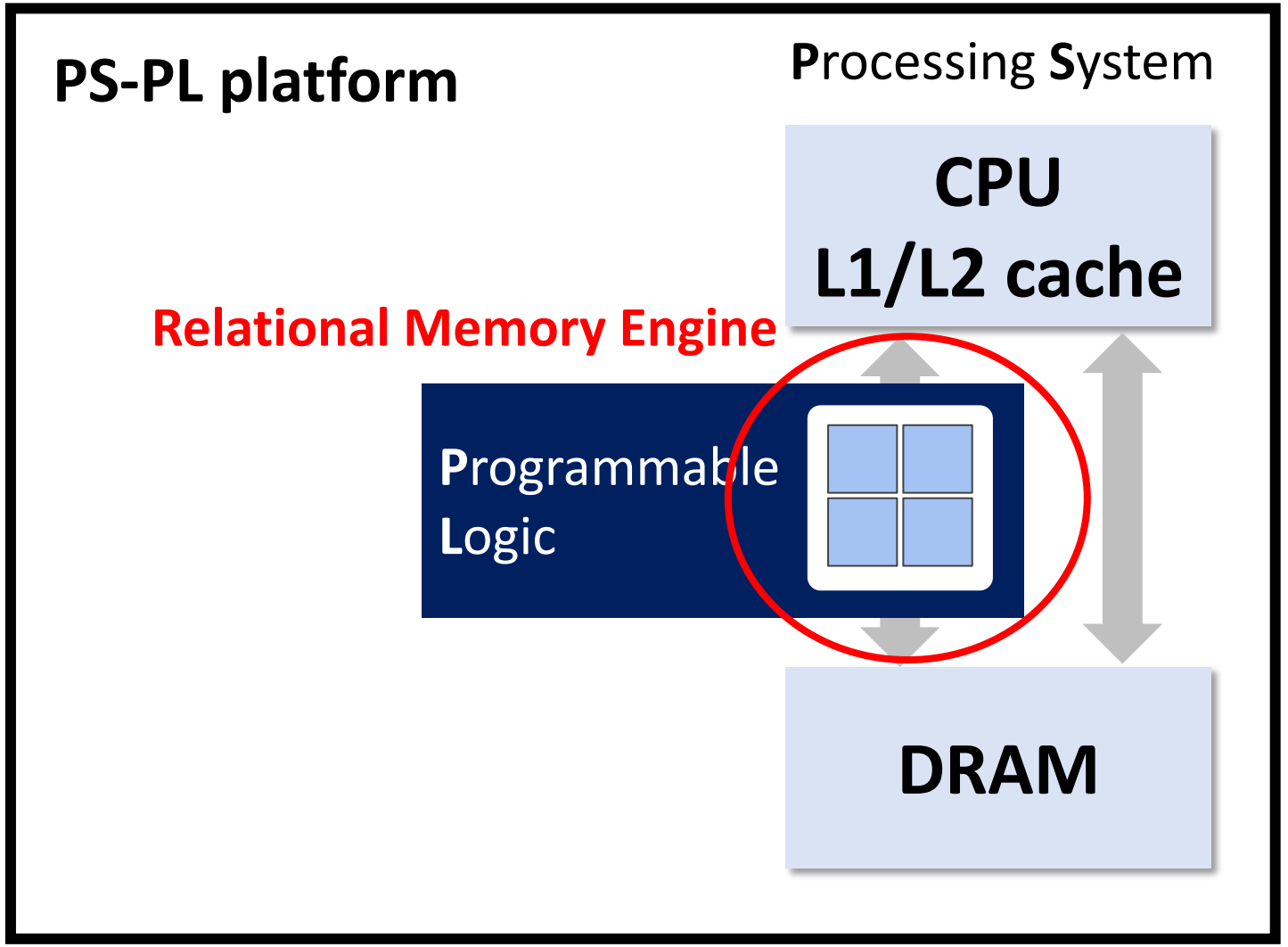
*PL can be a **go-between***
CPU in PS** & **DRAM in PS



AMD XILINX
UltraScale+

ENZIAN

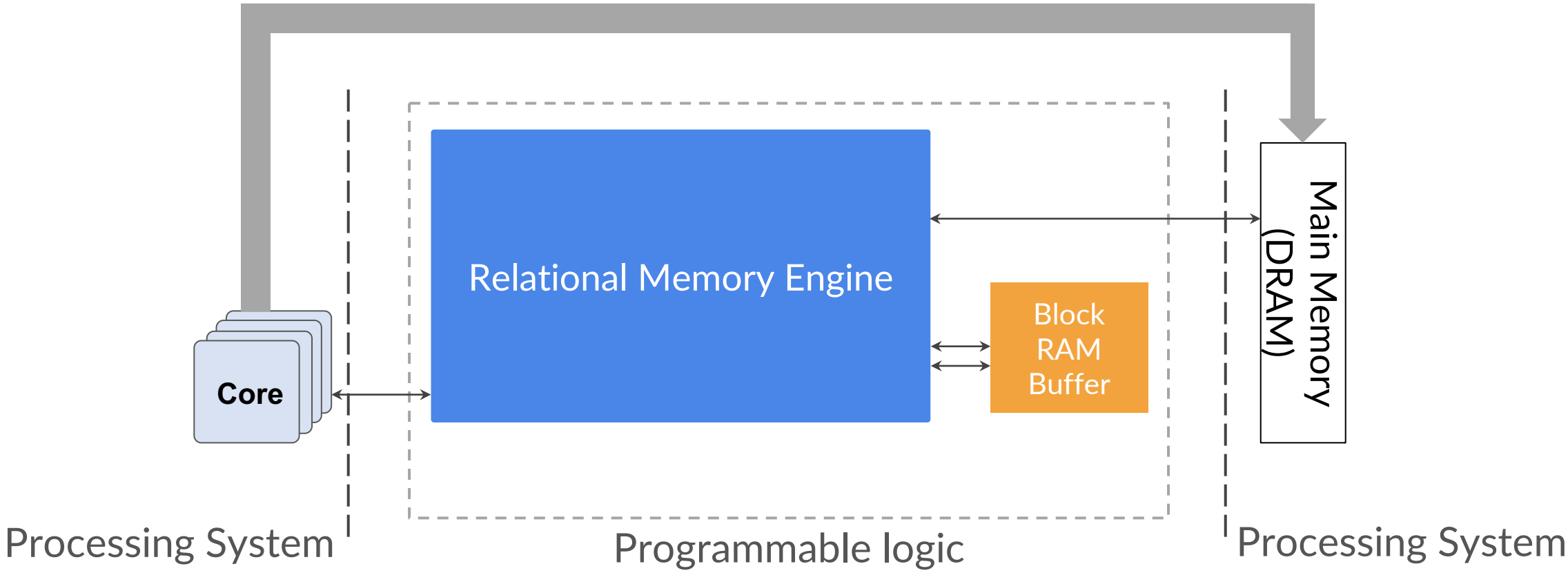
intel AGILEX



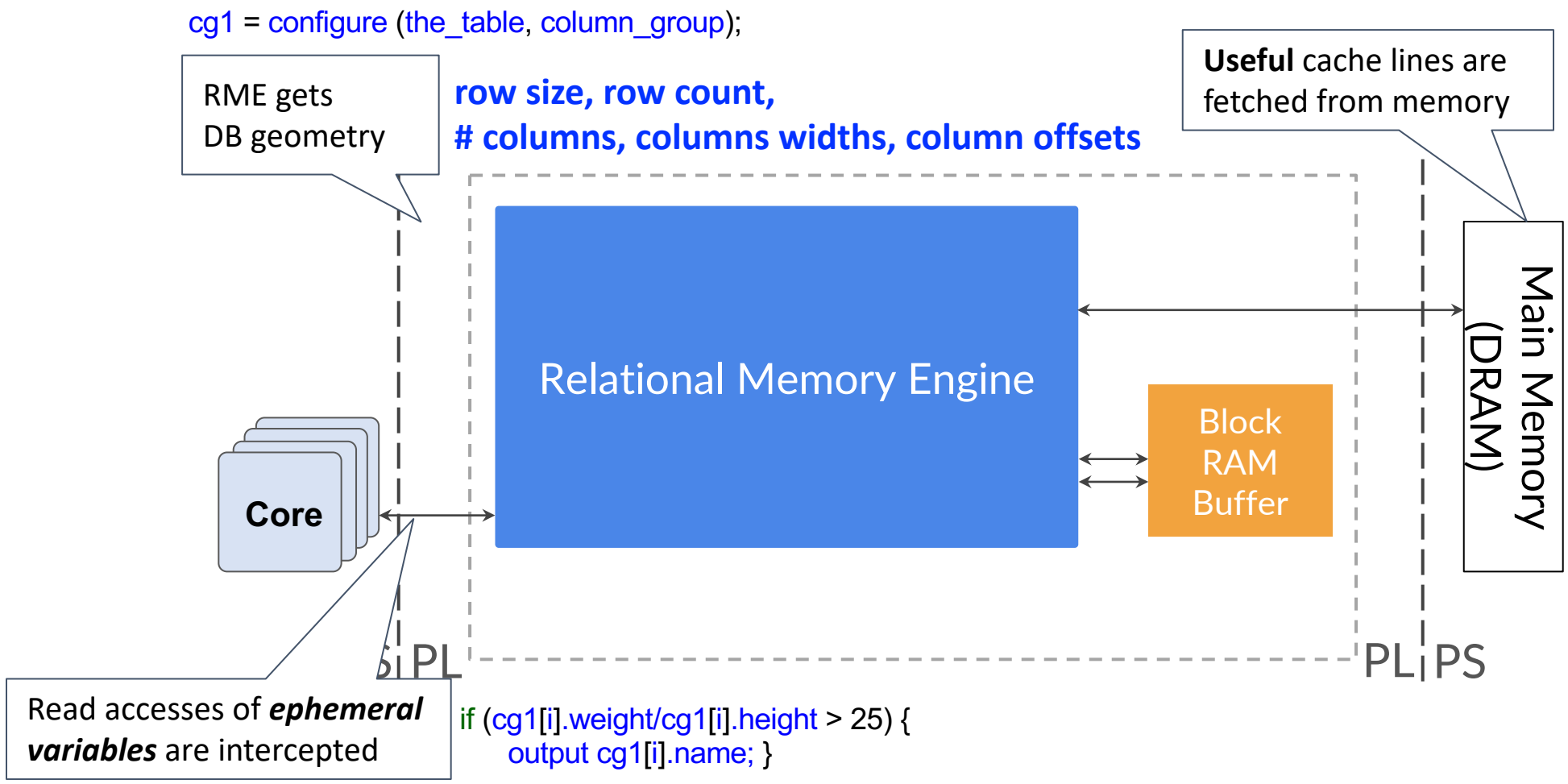
programmable logic in the middle

also known as "bump in the wire" in some communities

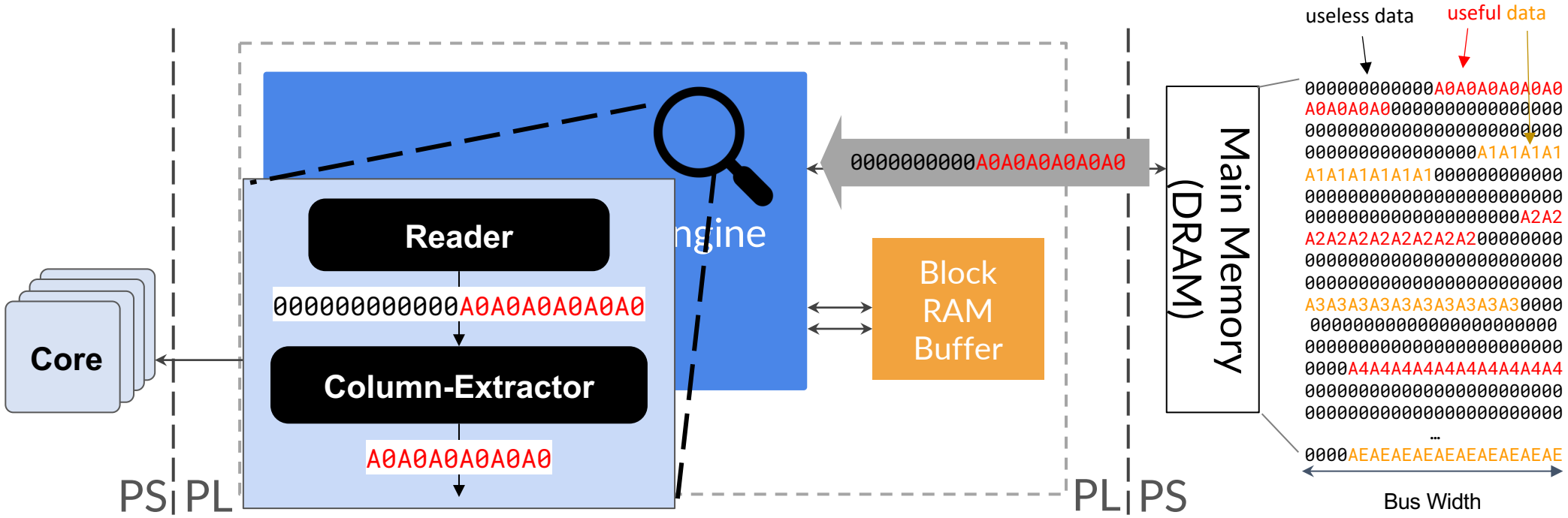
Relational Memory Engine



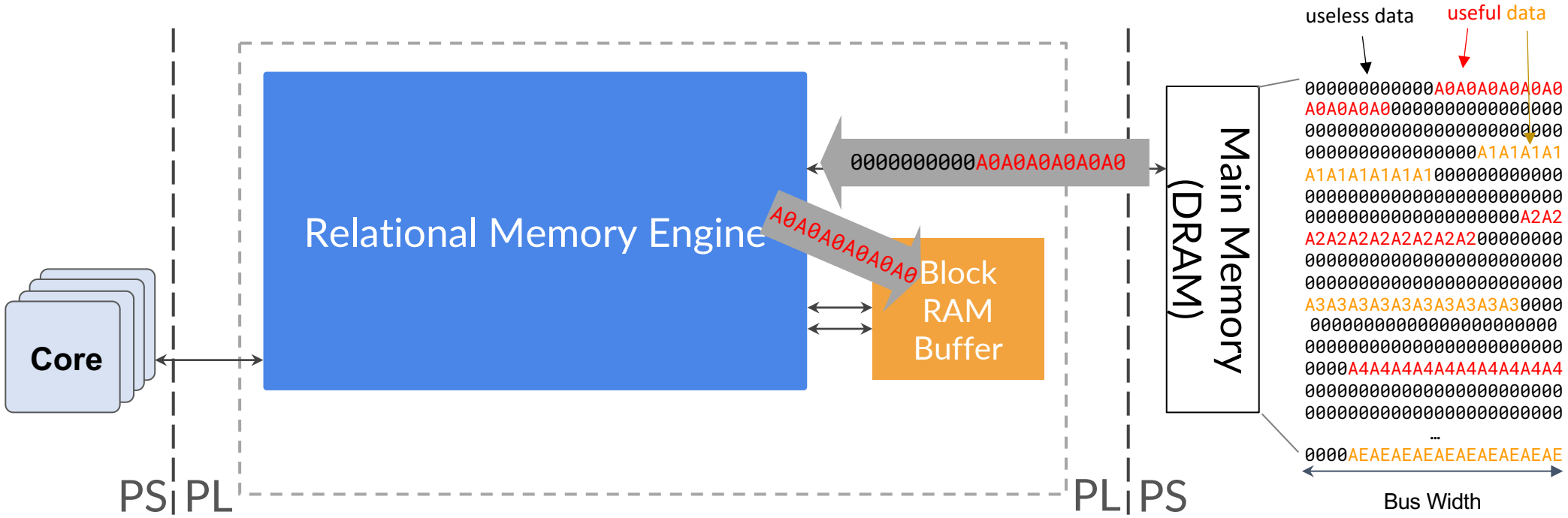
Relational Memory Engine



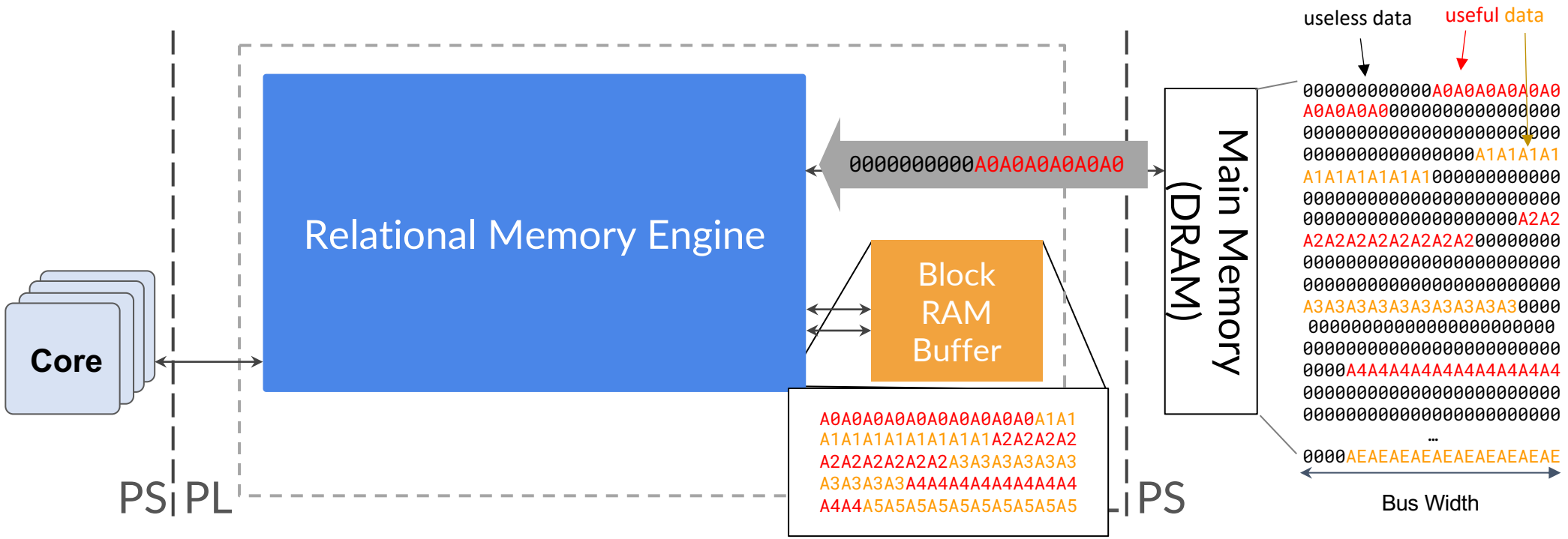
Relational Memory Engine



Relational Memory Engine

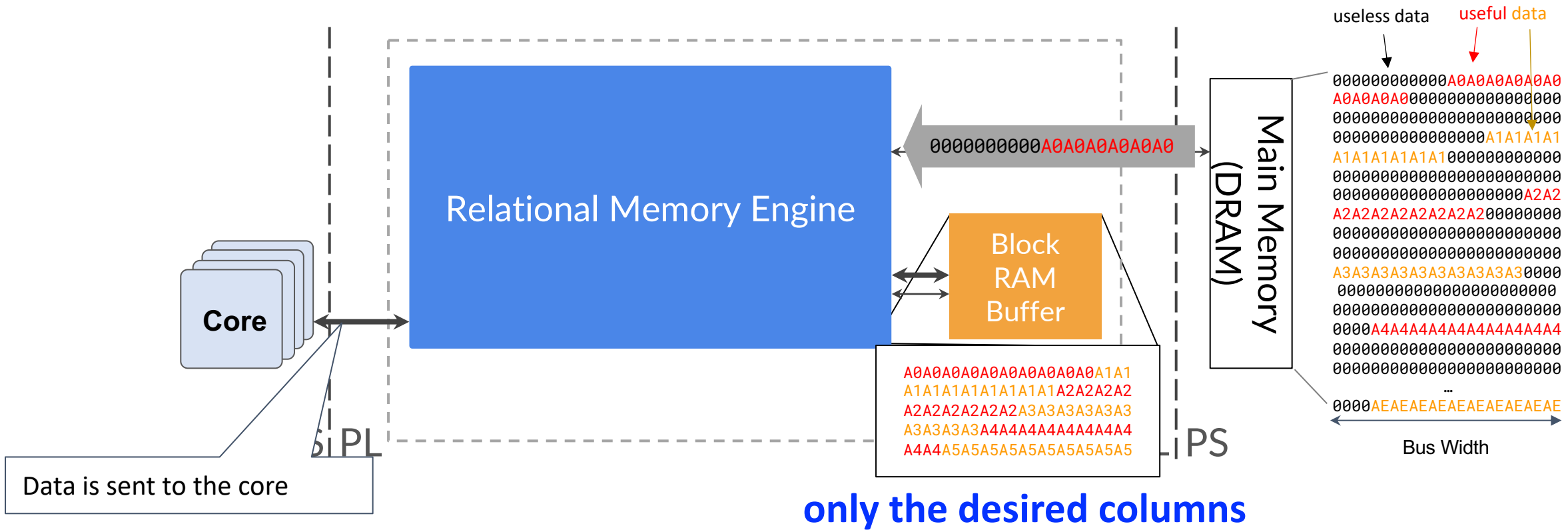


Relational Memory Engine

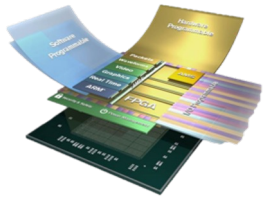


only the desired columns

Relational Memory Engine

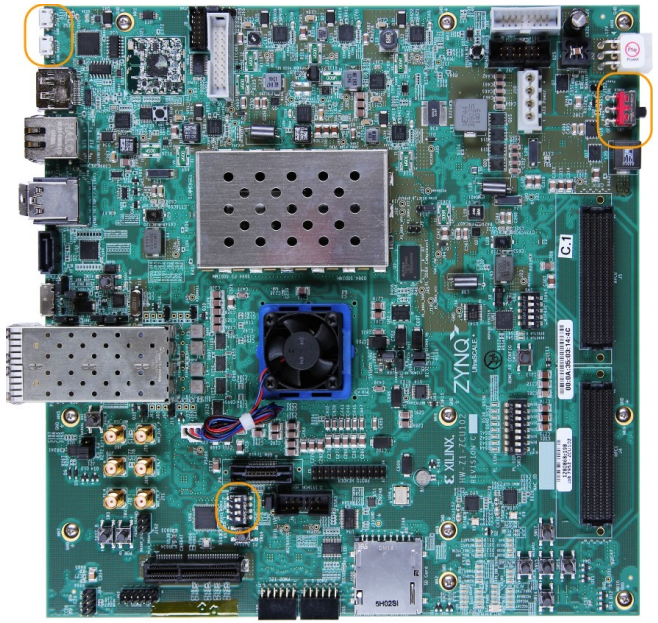


Evaluation



AMD XILINX
UltraScale+
ZCU102 platform

- CPUs : 4x ARM Cortex-A53
- L1/L2 Cache : 32+32KB I+D / 1 MB
- PS Frequency : 1.5 GHz
- PL Frequency : 100MHz



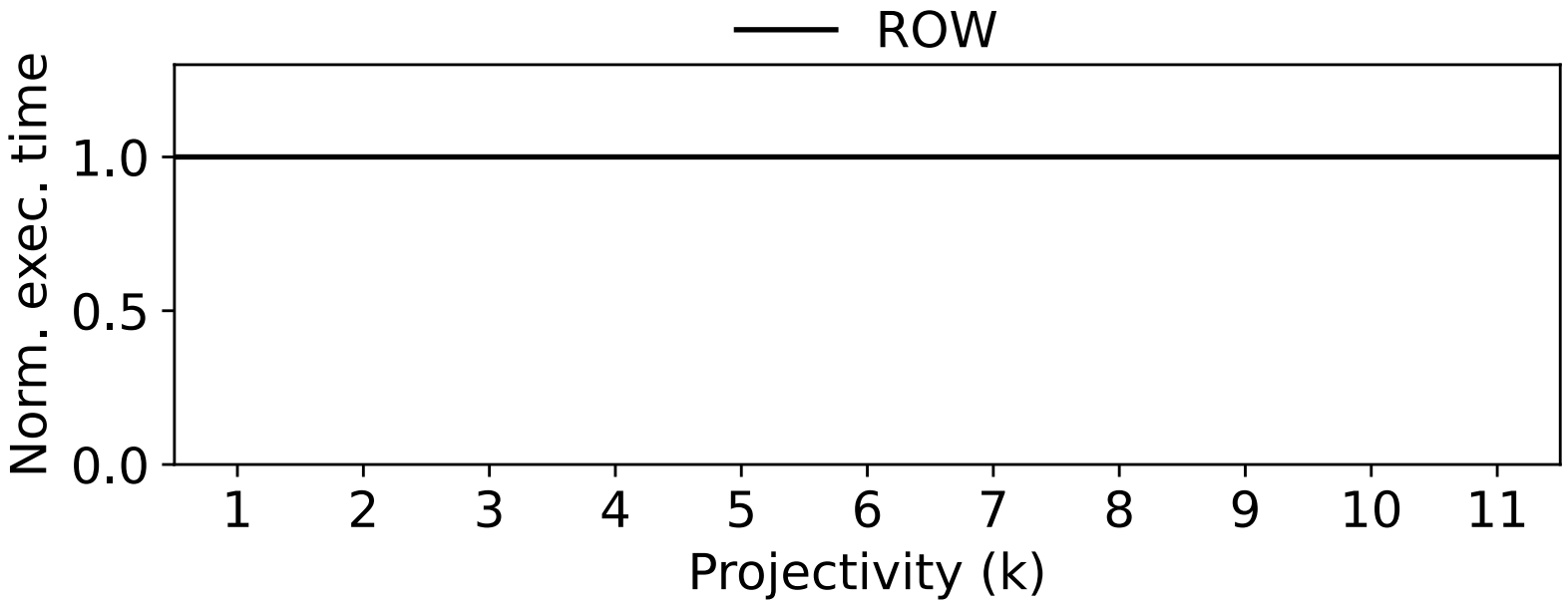
ROW : Direct row-wise access
COL : Direct columnar access
RME : using Relational Memory Engine

every approach uses the CPU

RME transforms data using PL

Queries Varying Projectivity

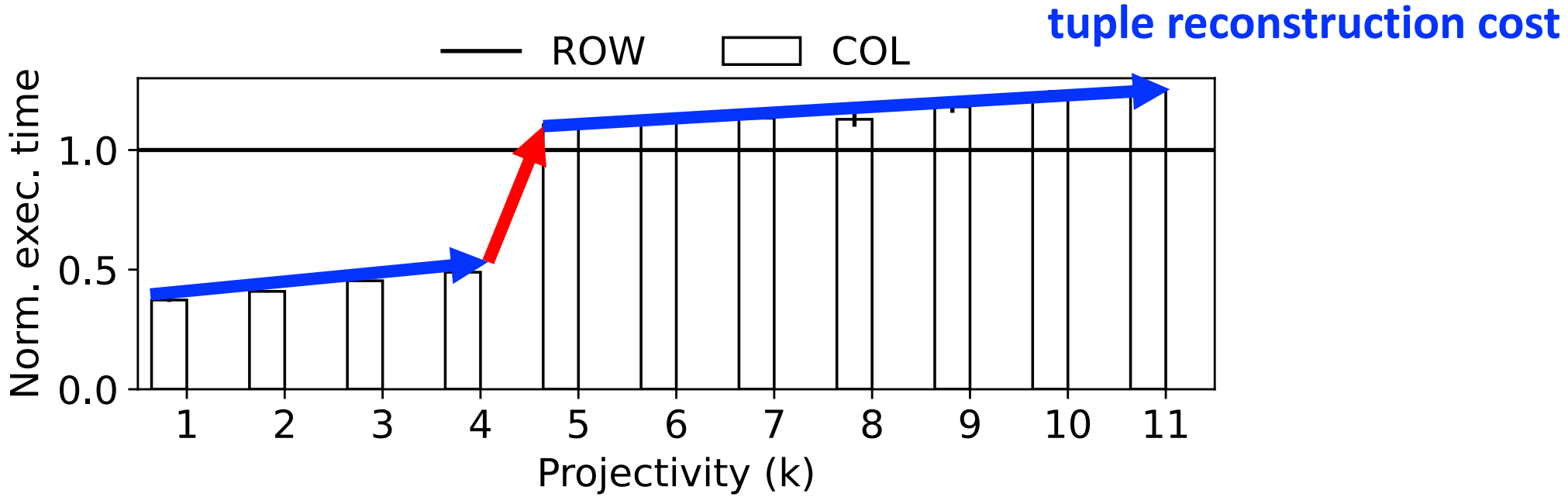
Q1: SELECT A1 , A2 , … , Ak FROM S;



Row size: 64 Bytes, Column size: 4 Bytes

Queries Varying Projectivity

Q1: SELECT A1 , A2 , … , Ak FROM S;

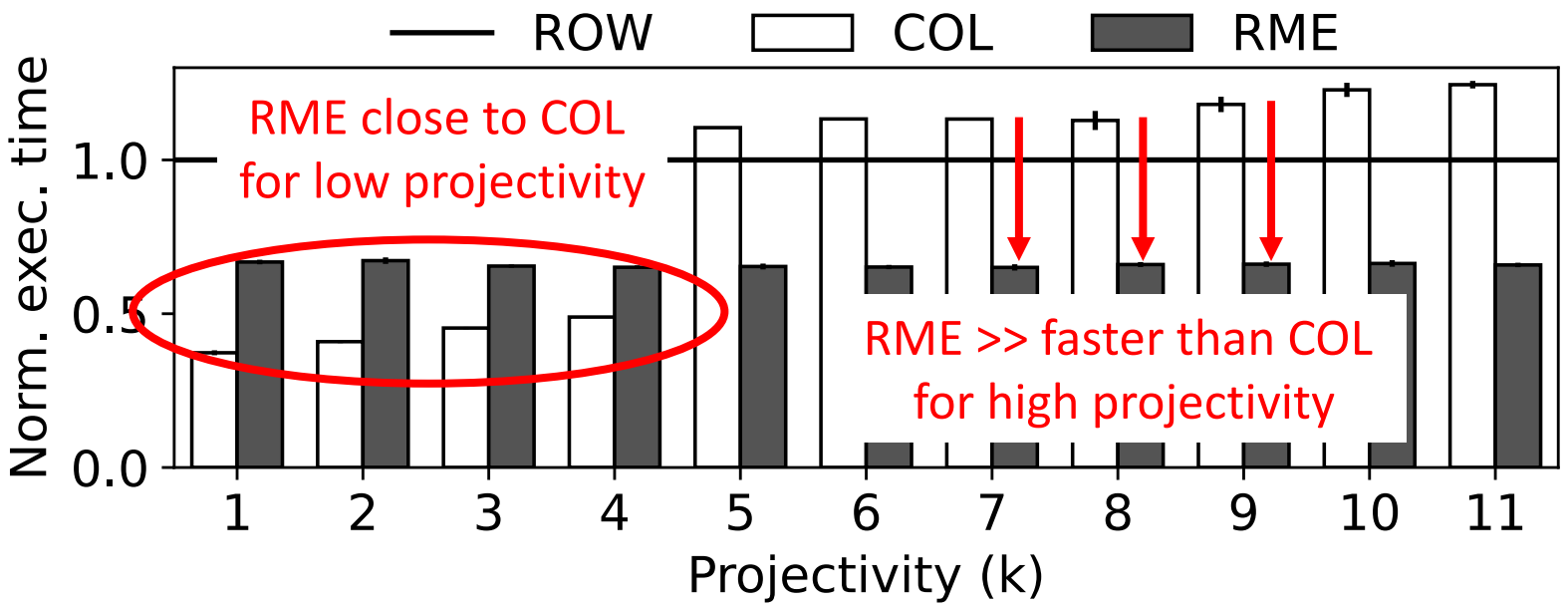


prefetcher supports up to four parallel streams

Row size: 64 Bytes, Column size: 4 Bytes

Queries Varying Projectivity

Q1: SELECT A1 , A2 , … , Ak FROM S;



RME provides stable performance irrespectively of projectivity

Row size: 64 Bytes, Column size: 4 Bytes

Are we done? What next?

Updates

Updates – Idea!

name	ID	age	height	weight
Alice	1	10	120	34
Bob	2	71	175	74
Charles	3	37	168	61
David	4	25	179	80
Eve	5	43	168	58
Frank	6	22	181	79
Greg	7	52	175	67
Henry	8	17	169	76
Iris	9	34	158	49
Jane	10	29	165	59
Kenneth	11	31	184	94
Luke	12	13	125	38

Updates – Idea!

name	ID	age	height	weight	TS_{from}	TS_{to}
Alice	1	10	120	34	t_1	∞
Bob	2	71	175	74	t_1	∞
Charles	3	37	168	61	t_1	∞
David	4	25	179	80	t_1	∞
Eve	5	43	168	58	t_1	∞
Frank	6	22	181	79	t_1	∞
Greg	7	52	175	67	t_1	∞
Henry	8	17	169	76	t_2	∞
Iris	9	34	158	49	t_2	∞
Jane	10	29	165	59	t_2	∞
Kenneth	11	31	184	94	t_2	∞
Luke	12	13	125	38	t_2	∞

Updates – Idea!

name	ID	age	height	weight	TS_{from}	TS_{to}
Alice	1	10	120	34	t_1	∞
Bob	2	71	175	74	t_1	∞
Charles	3	37	168	61	t_1	∞
David	4	25	179	80	t_1	∞
Eve	5	43	168	58	t_1	∞
Frank	6	22	181	79	t_1	∞
Greg	7	52	175	67	t_1	∞
Henry	8	17	169	76	t_2	∞
Iris	9	34	158	49	t_2	∞
Jane	10	29	165	59	t_2	∞
Kenneth	11	31	184	94	t_2	t_3
Luke	12	13	125	38	t_2	∞

At t_3:

DELETE FROM table WHERE ID = 11;

Updates – Idea!

name	ID	age	height	weight	TS_{from}	TS_{to}
Alice	1	10	120	34	t_1	∞
Bob	2	71	175	74	t_1	∞
Charles	3	37	168	61	t_1	∞
David	4	25	179	80	t_1	∞
Eve	5	43	168	58	t_1	∞
Frank	6	22	181	79	t_1	∞
Greg	7	52	175	67	t_1	∞
Henry	8	17	169	76	t_2	t_5
Iris	9	34	158	49	t_2	∞
Jane	10	29	165	59	t_2	∞
Kenneth	11	31	184	94	t_2	t_3
Luke	12	13	125	38	t_2	∞
Henry	**8**	**17**	**169**	**82**	**t_5**	**∞**

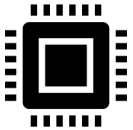
can be part of the programmable logic ✓

At t_5:
UPDATE weight=82 FROM table WHERE ID = 8;

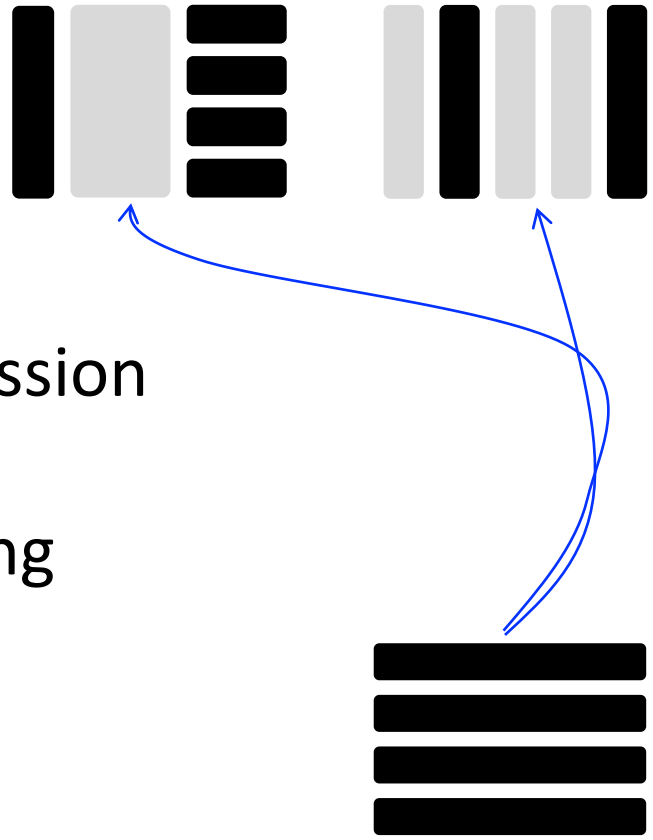
At t_3:
DELETE FROM table WHERE ID = 11;

Are we done? What next?

Updates



Compression



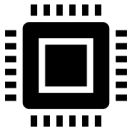
dictionary compression

run-length encoding

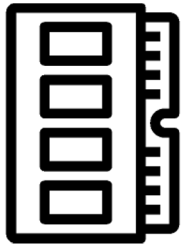
more research is needed!

Are we done? What next?

Updates



Compression



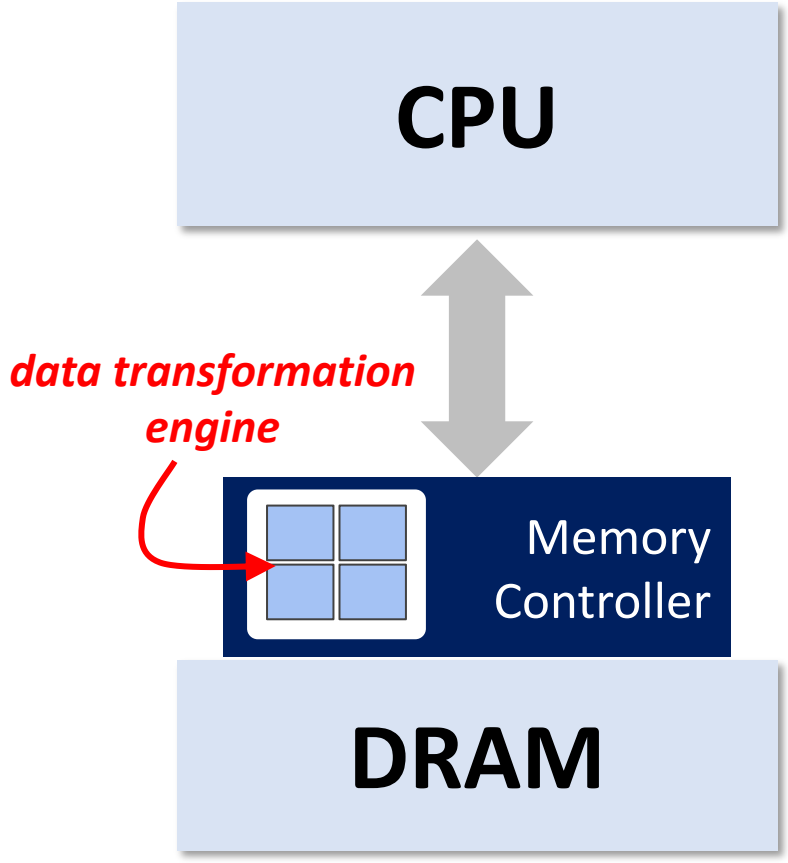
Integrate Relational Memory logic in the memory controller

Push ***data transformation*** within the ***memory controller***

ongoing work funded by Red Hat

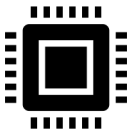
accessible to any codebase
more efficient DRAM utilization

need to extend the processor's ISA!

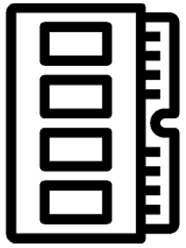


Are we done? What next?

Updates



Compression



Integrate Relational Memory logic in the memory controller

Relational Storage

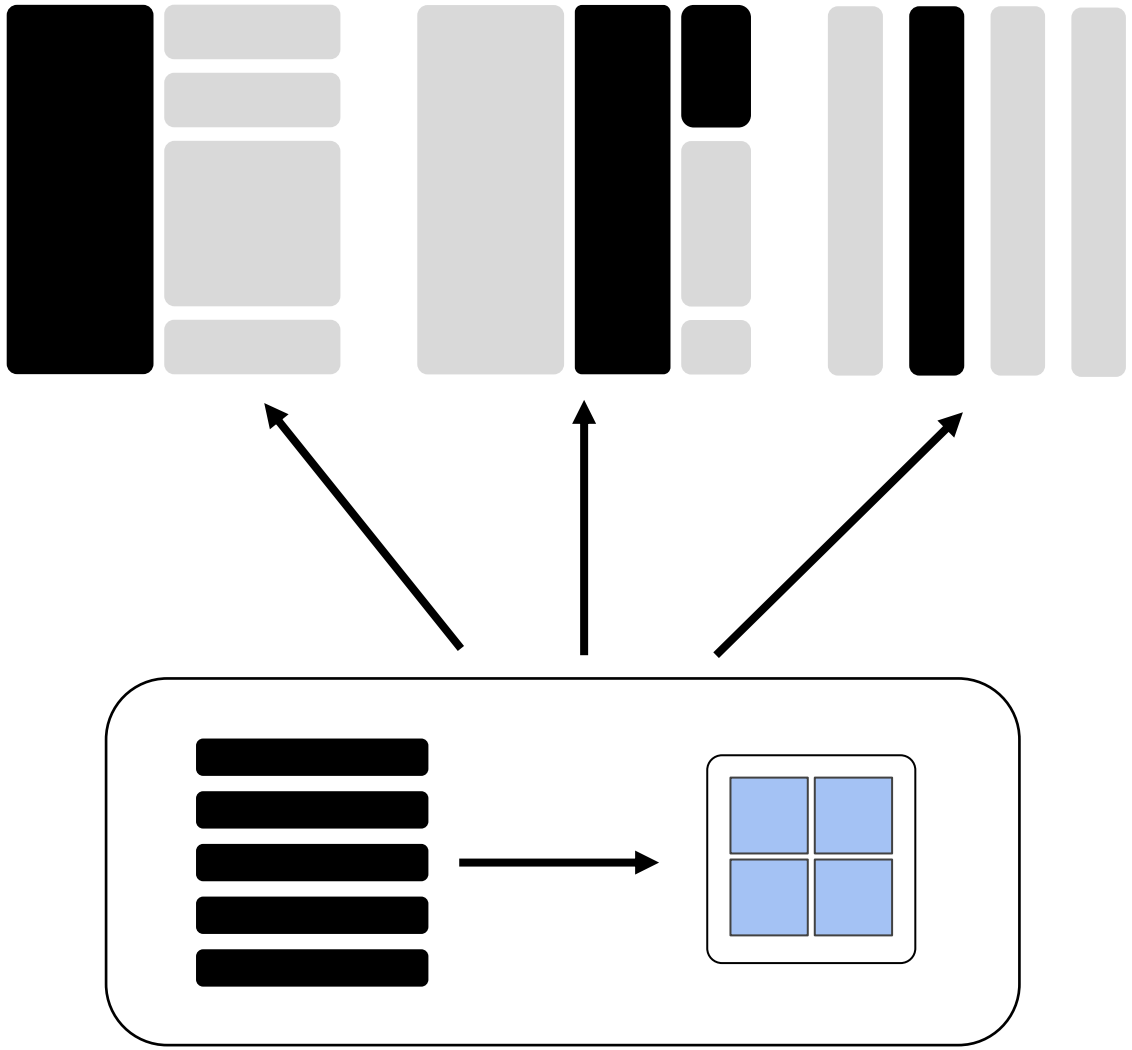
Beyond memory?

Computational SSDs

Relational Storage – Idea 1

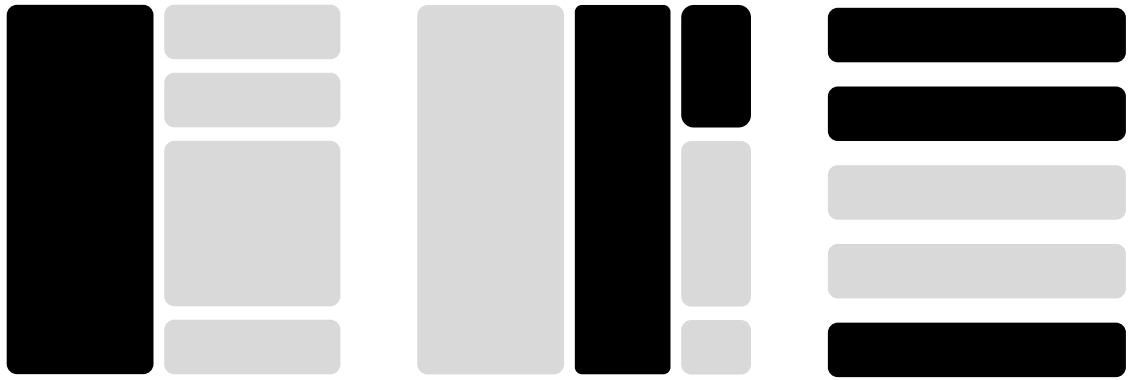
use computational SSD
to *transform* data on the fly

use computational SSD
to *decompress* data on the fly



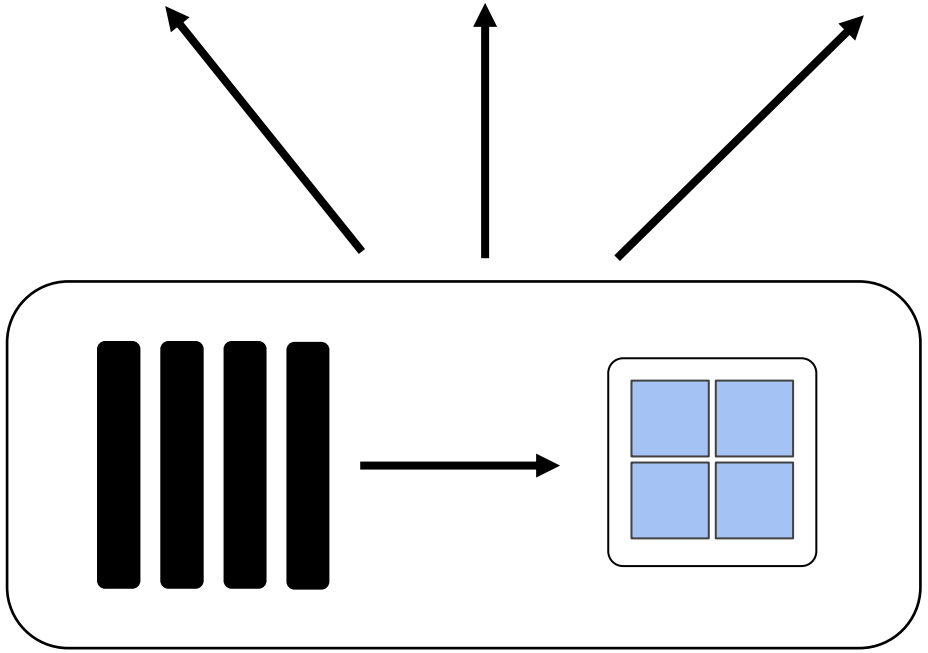
Relational Storage – Idea 2

flip the layouts!



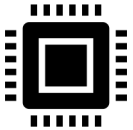
store columns in SSD

perform *tuple reconstruction* on the fly



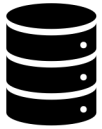
Are we done? What next?

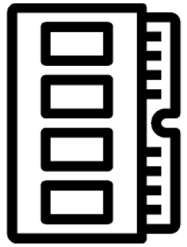

Updates

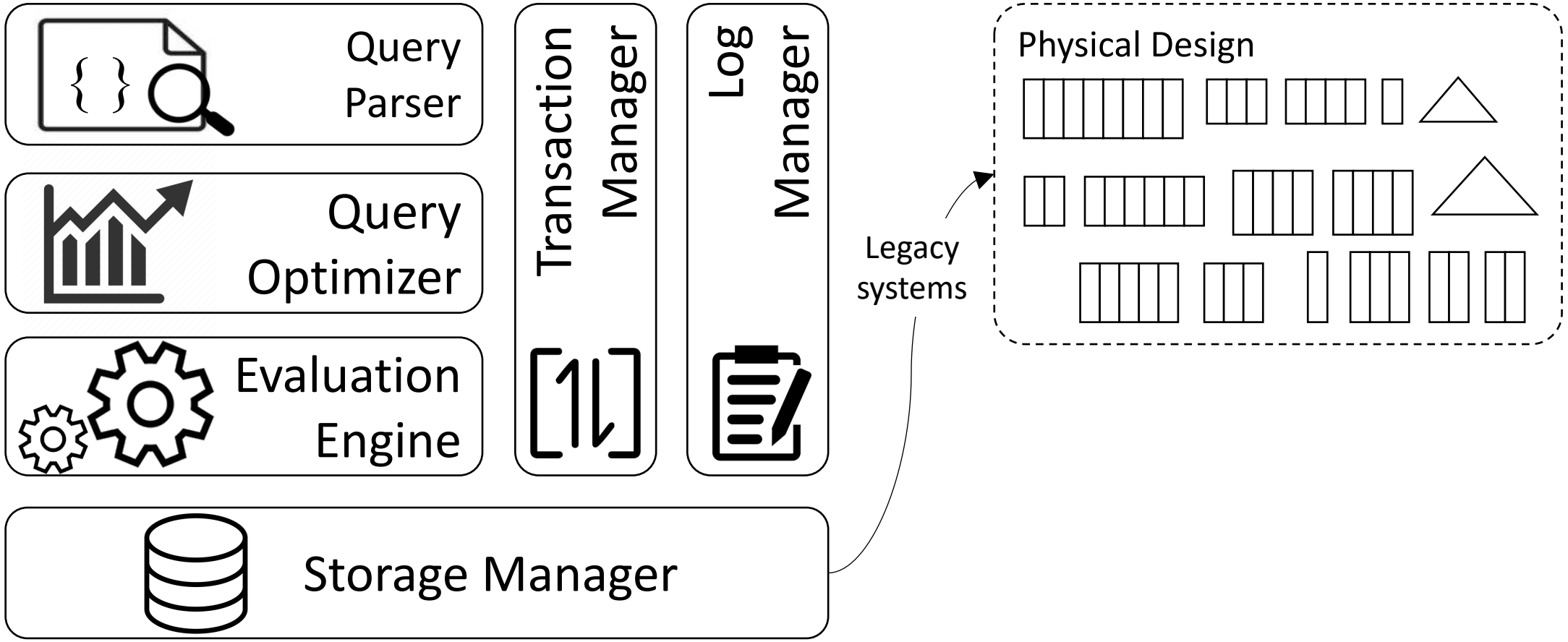

Compression

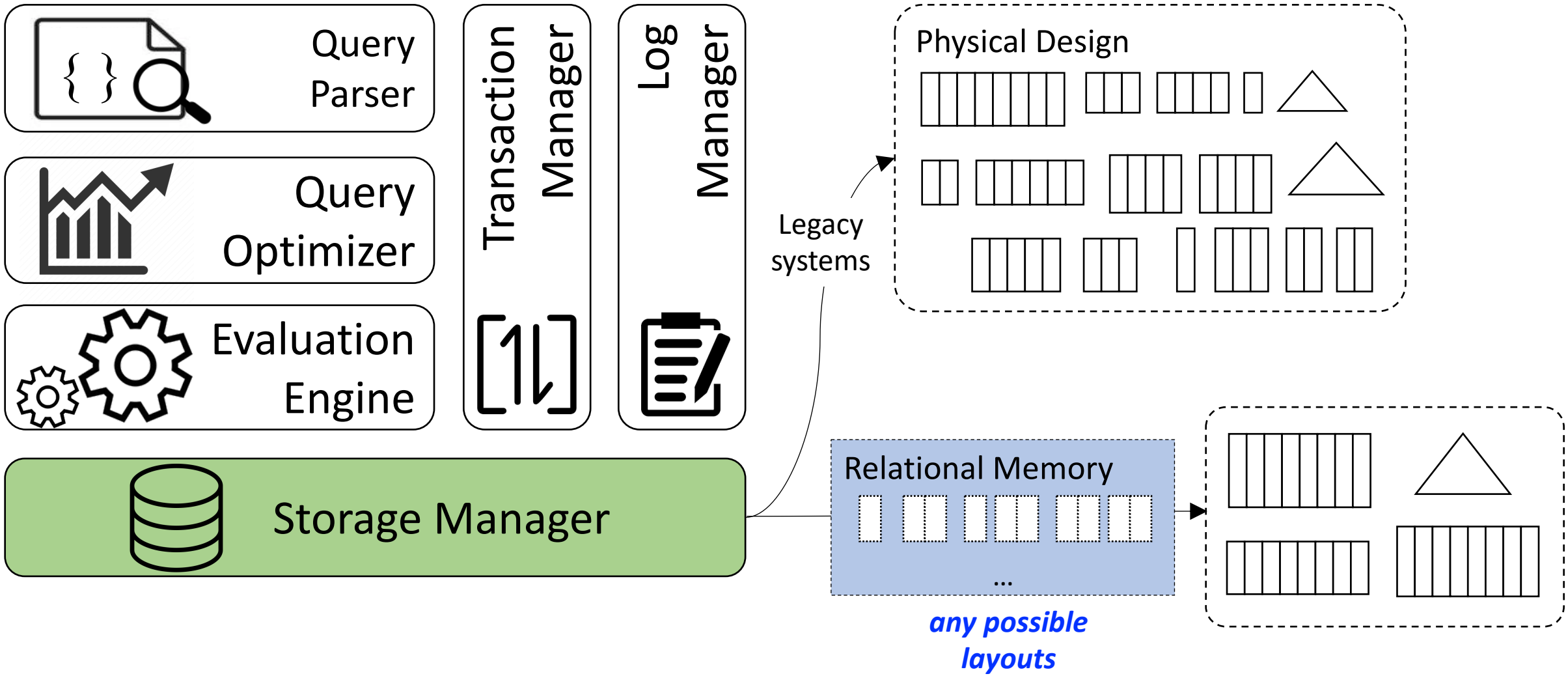

Relational Storage


Computational SSDs

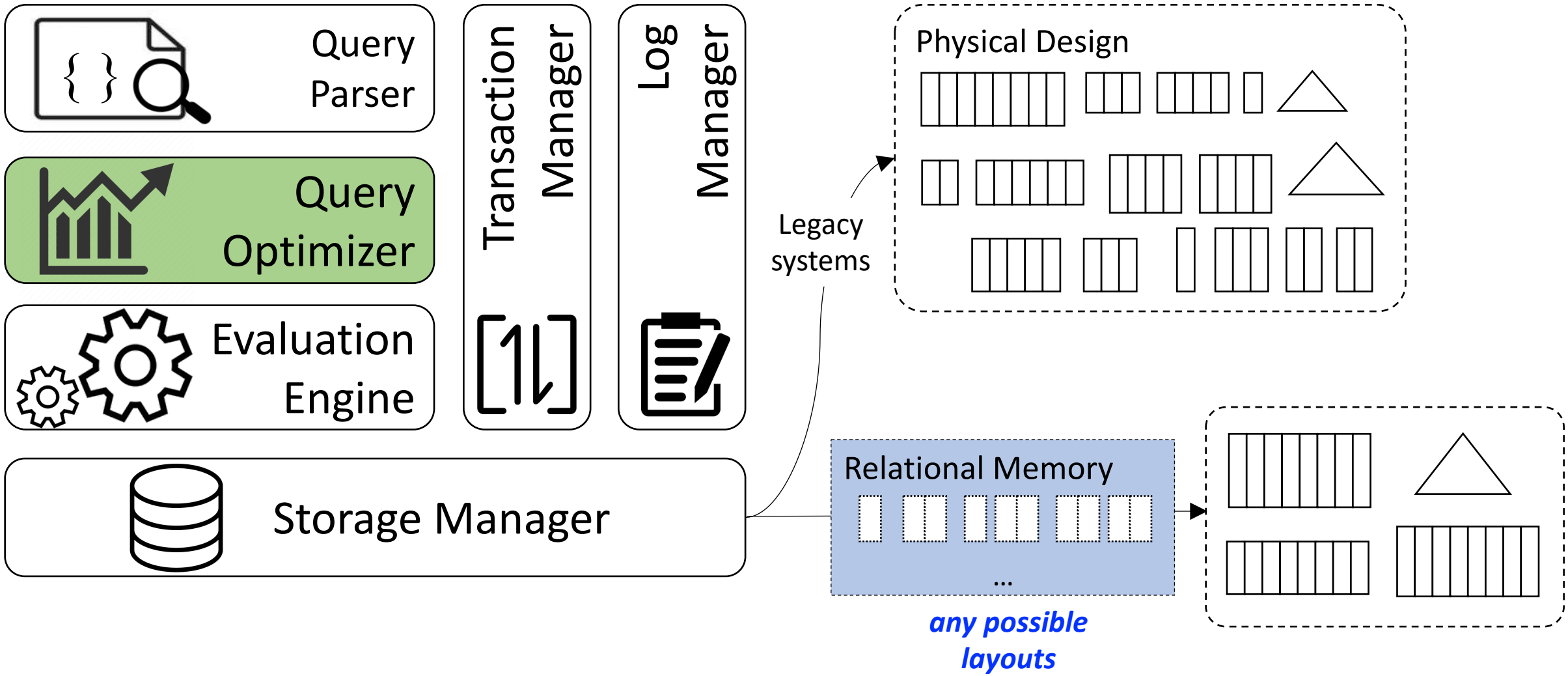

Impact on DB architectures


Integrate Relational Memory logic in the memory controller

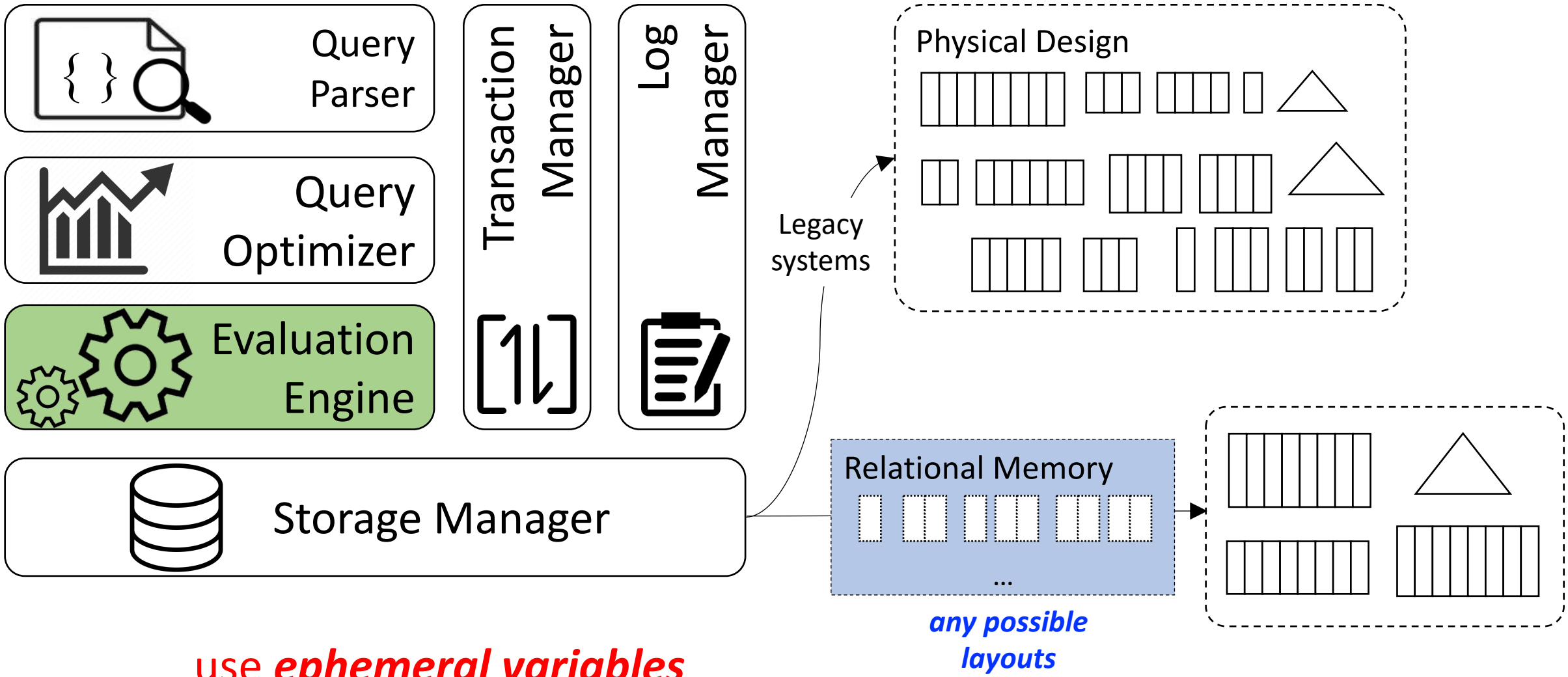

Query Parser
Query Optimizer
Evaluation Engine
Transaction Manager
Log Manager
Storage Manager
Legacy systems
Physical Design



simplify physical design!



the ***optimal layout*** is ***always available***



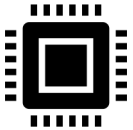
use ***ephemeral variables***

updates to the ***base*** data

code generation of the optimal plan

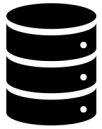
Are we done? What next?

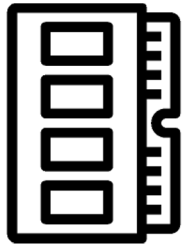

Updates


Compression


Relational Storage


Computational SSDs

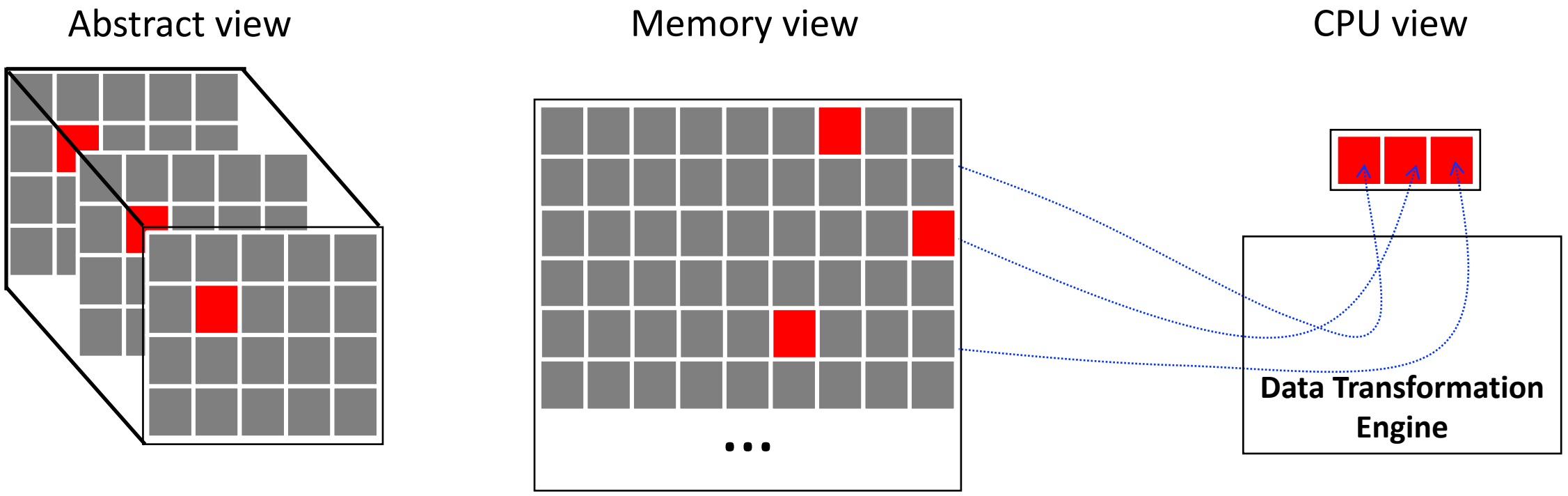

Impact on DB architectures


Integrate Relational Memory logic in the memory controller


Data Transformation for ML workloads, e.g., tensor slicing

Tensor Data Transformation

e.g., slicing, transposing

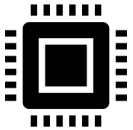


develop a *data transformation algebra* for any high-dimensional data object

integrate with memory controller

Are we done? What next?

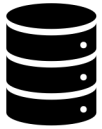
Updates



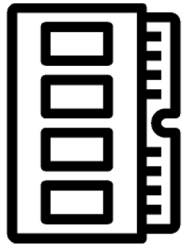
Compression

Relational Storage

Computational SSDs



Impact on DB architectures



Integrate Relational Memory logic in the memory controller

Data Transformation for ML workloads, e.g., tensor slicing

room for (a lot of) innovation!

Relational Fabric – Transparent Data Transformation

Team

Sponsors

Red Hat cisco NSF

Thank you!

disc.bu.edu/relational-memory

disc.bu.edu