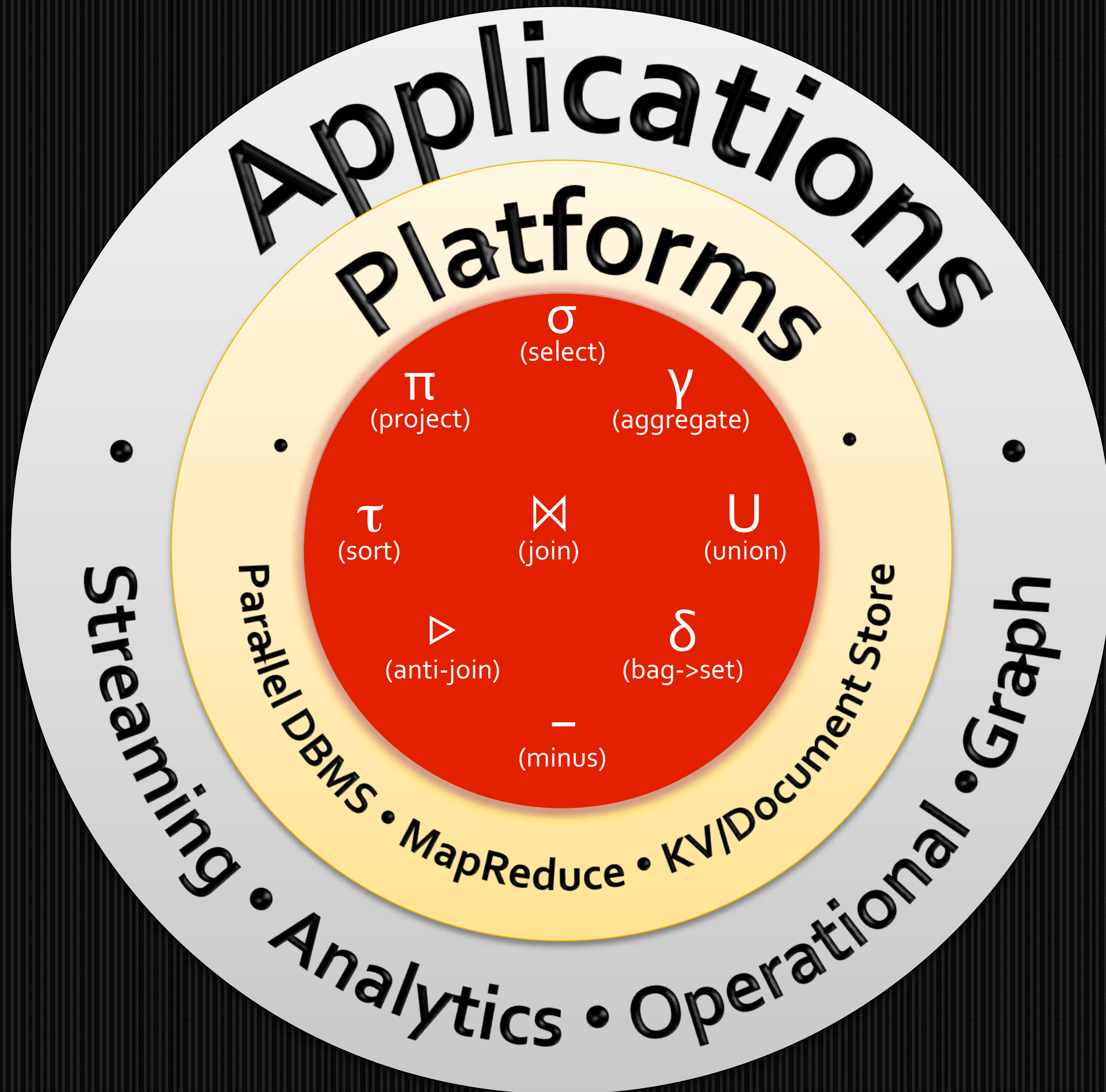
Data@ Bare Metal Speed

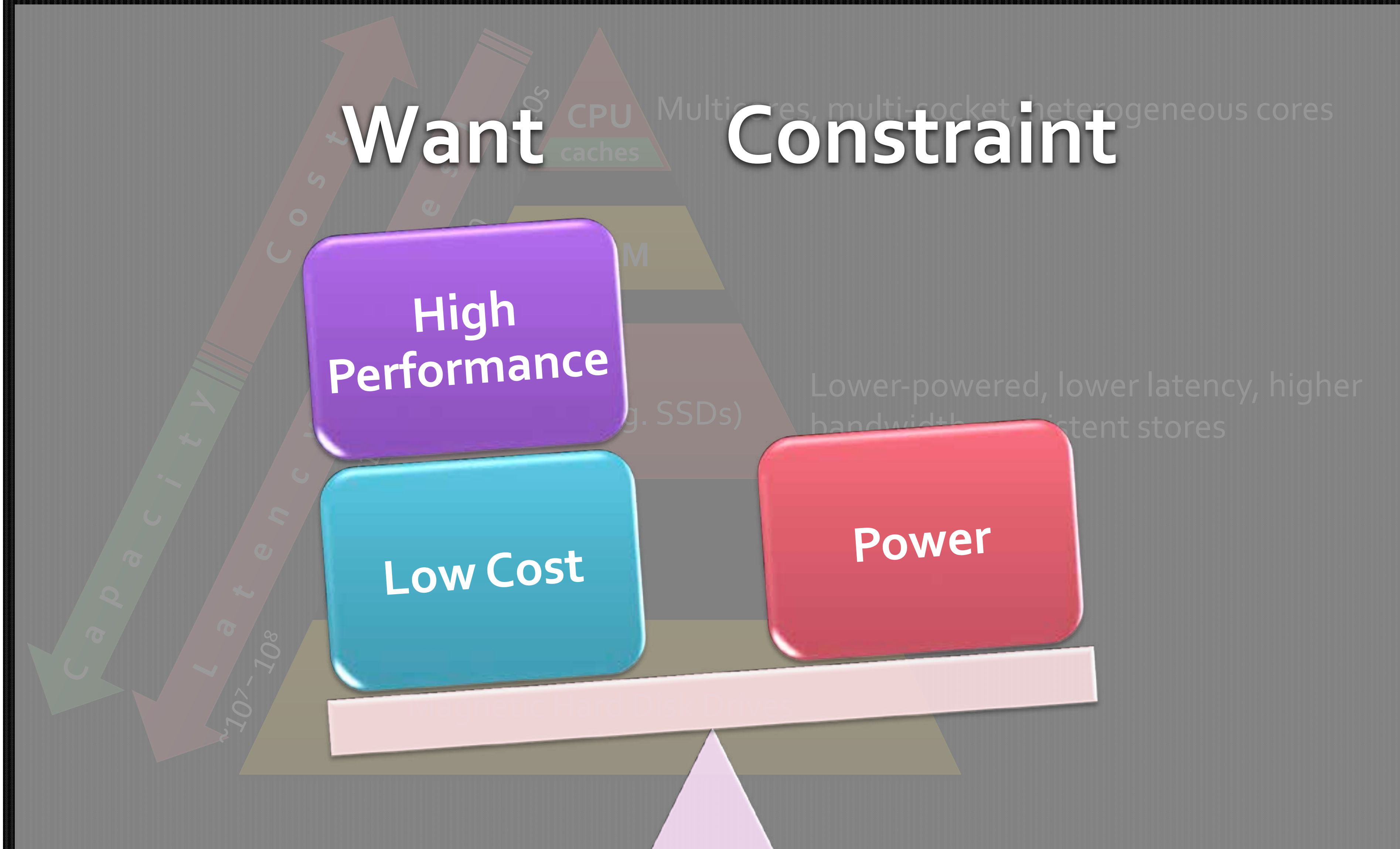
Jignesh M. Patel

THE UNIVERSITY of WISCONSIN MADISON

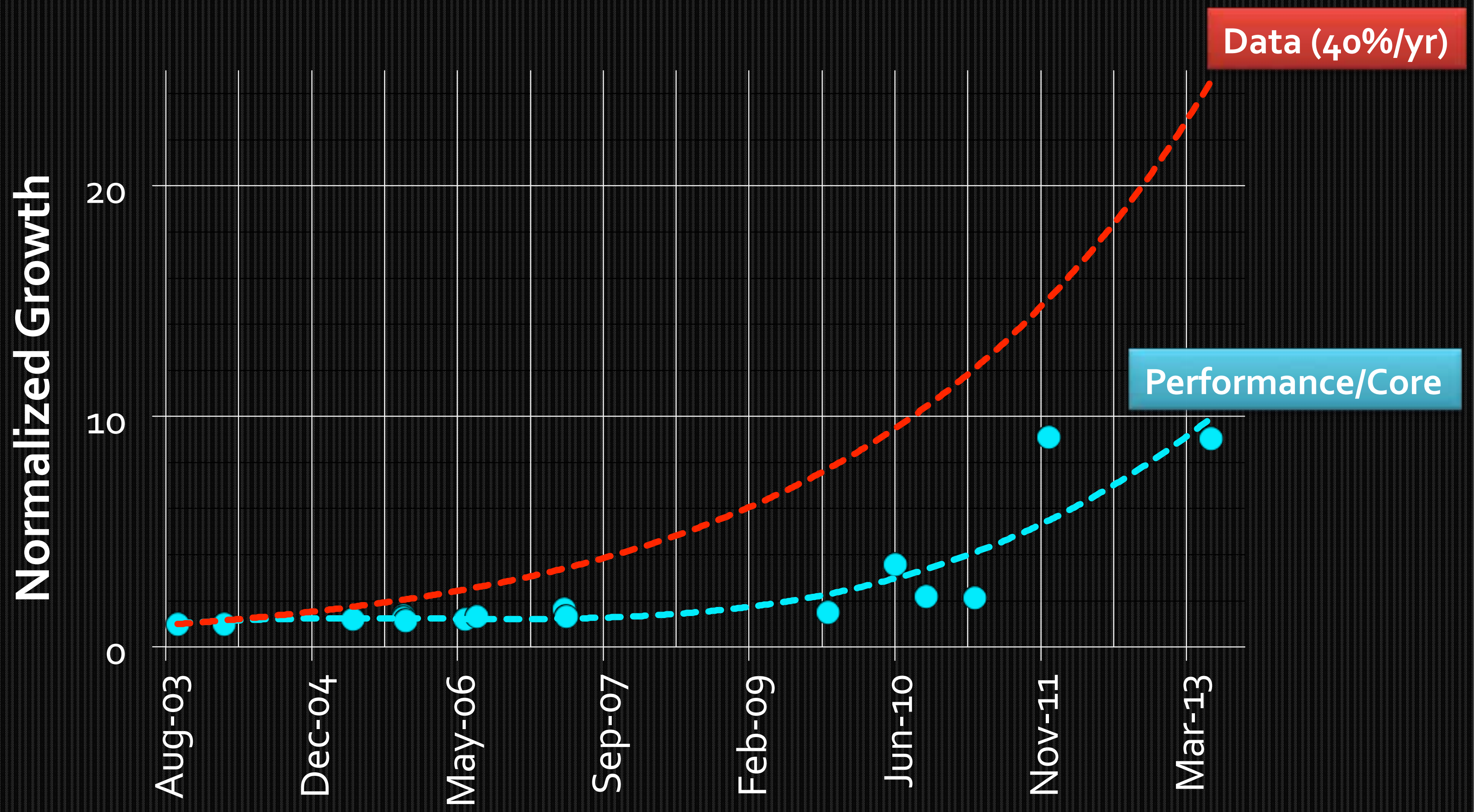
The Wisconsin Quickstep Project


Applications
Platforms
σ (select)
π (project)
γ (aggregate)
τ (sort)
⋈ (join)
∪ (union)
▷ (anti-join)
δ (bag->set)
− (minus)
Parallel DBMS • MapReduce • KV/Document Store
Streaming • Analytics • Operational • Graph

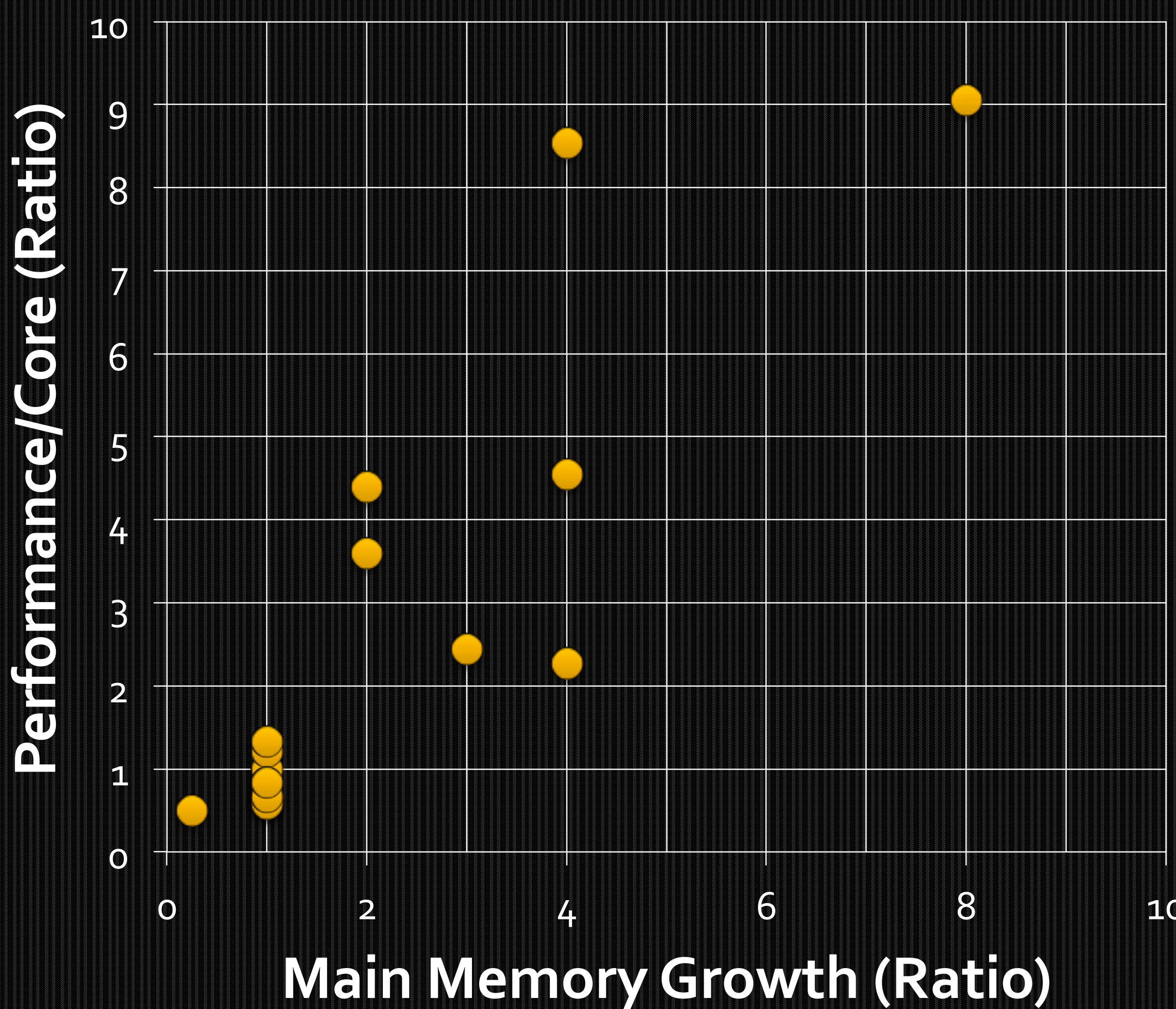
Disruptive hardware trends



TPC-H: Big-3 Vendors, 3TB scale

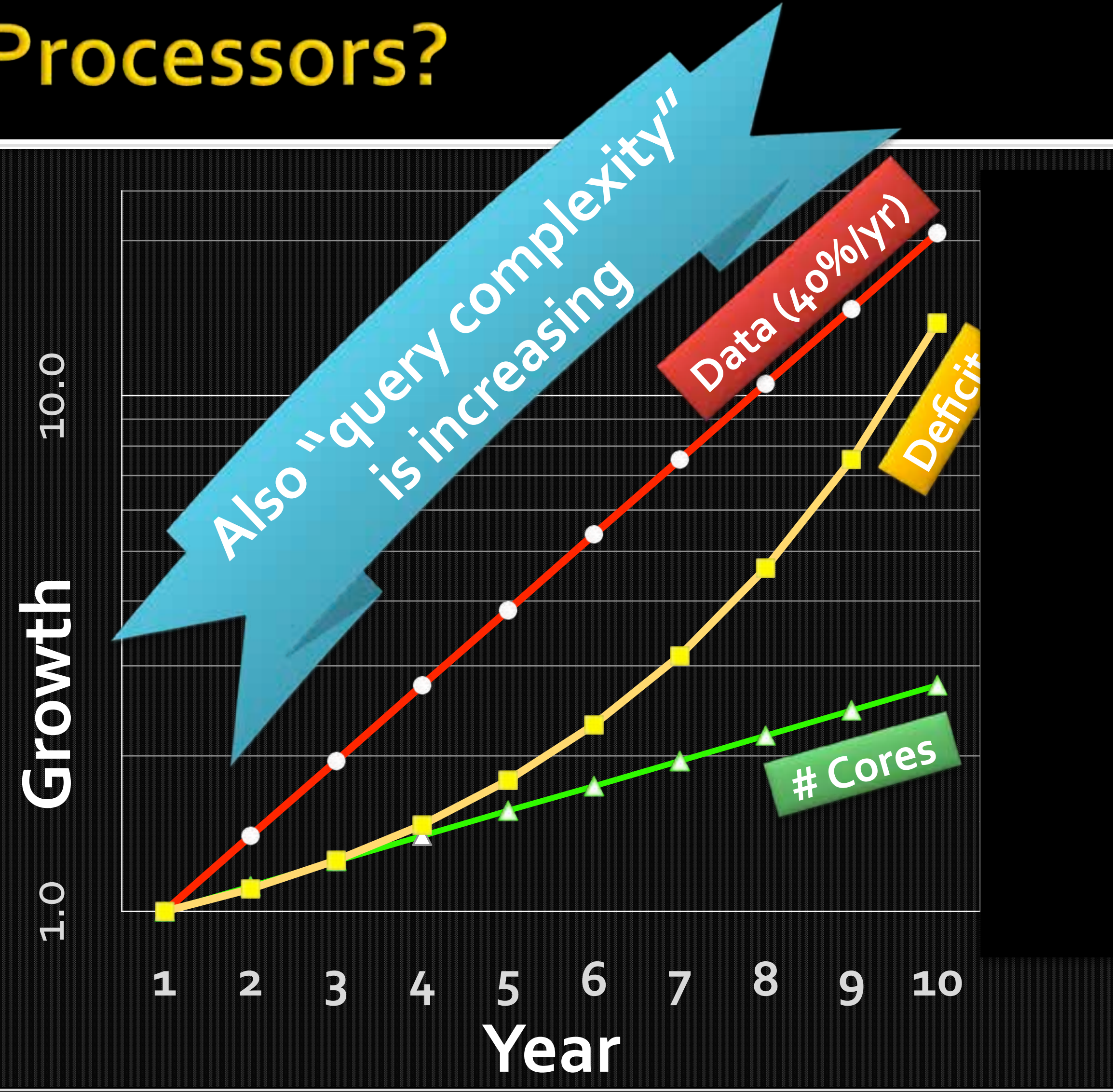


TPC-H: Growth driven by moving data to main memory



What's Next for Processors?

- Future processor design?
  1. Keep adding cores (~40% per generation)
  2. Heterogeneous cores
  3. Programmable functional units
- But, systems must stay within a power budget
- Data growth continues unabated



Need to do more with less.

Quickstep

Goal	• Run data analytics @ hardware speeds
Short-term	• Run @ the speed of hardware today
Long-term	• Hardware-software co-design for data kernels

Li and Patel, SIGMOD'13

Scan: A Key Data Processing Kernel

What?

- Scan a column of a table applying some predicate

Why?

- A key primitive in database
- "The" critical kernel in main memory analytic systems

How?

- Conserve memory bandwidth: **BitWeaving** the data
- Use every bit of data that is brought to the processer efficiently using **intra-cycle parallelism**

Blog: http://bigfastdata.blogspot.com

Li and Patel, SIGMOD'13

Focus on Column Scan (can be generalized)

Traditional Row Store

shipdate	...	discount	quantity
Mar-12-2013		5%	5
Jan-08-2013		2%	4
Apr-29-2013		10%	3
May-14-2013		0%	6
...	...	...	...
Feb-28-2013		5%	0

One big file

Column Store

shipdate
Mar-12-2013
Jan-08-2013
Apr-29-2013
May-14-2013
...
Feb-28-2013

File: 1

discount
5%
2%
10%
0%
...
5%

File: n-1

quantity
5
4
3
6
...
0

File: n

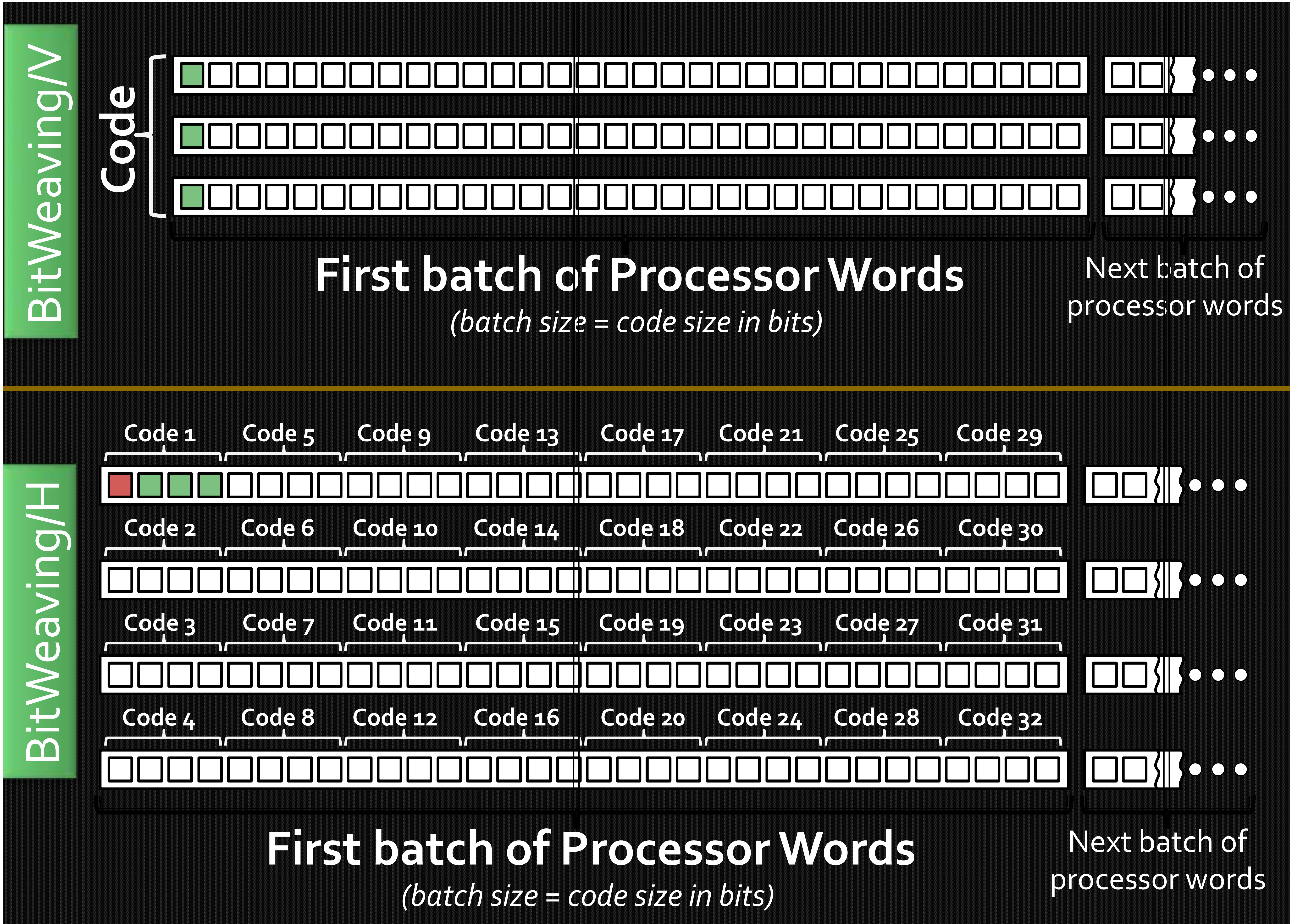
16 bits

Order-preserving compression

Column Codes:

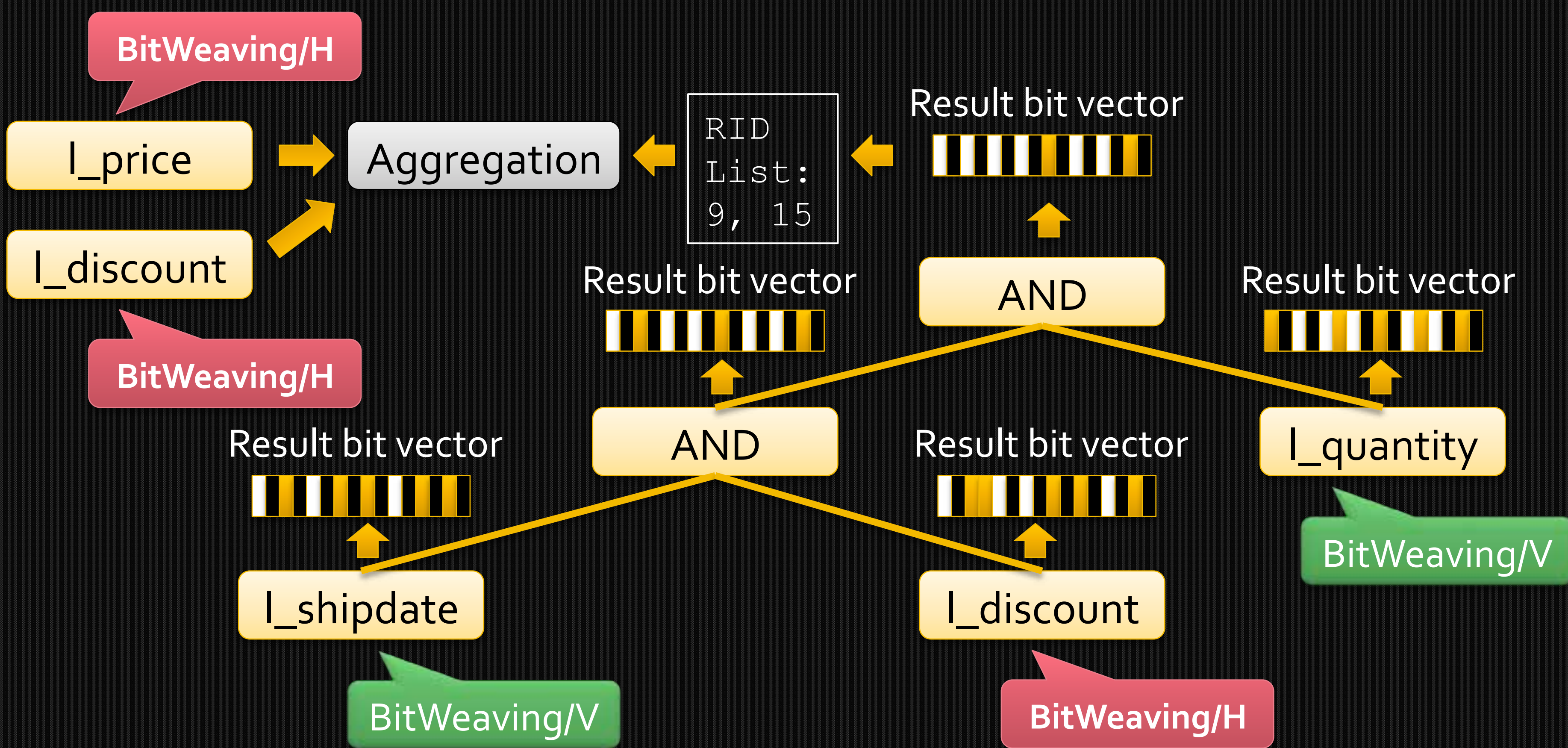
5	4	3	6	2	7	1	0	...

3 bits


BitWeaving/V
Code
First batch of Processor Words
(batch size = code size in bits)
Next batch of processor words
BitWeaving/H
Code 1
Code 5
Code 9
Code 13
Code 17
Code 21
Code 25
Code 29
Code 2
Code 6
Code 10
Code 14
Code 18
Code 22
Code 26
Code 30
Code 3
Code 7
Code 11
Code 15
Code 19
Code 23
Code 27
Code 31
Code 4
Code 8
Code 12
Code 16
Code 20
Code 24
Code 28
Code 32
First batch of Processor Words
(batch size = code size in bits)
Next batch of processor words

Framework – Example

```
SELECT SUM(l_discount * l_price) FROM lineitem
WHERE l_shipdate BETWEEN Date AND Date + 1 year
  AND l_discount BETWEEN Discount – 0.01 AND Discount + 0.01
  AND l_quantity < Quantity
```



BitWeaving/V

Column Codes:

	10	12	3	6	9	7	1	0
Word 1	1	1	0	0	1	0	0	0
Word 2	0	1	0	1	0	1	0	0
Word 3	1	0	1	1	0	1	0	0
Word 4	0	0	1	0	1	1	1	0

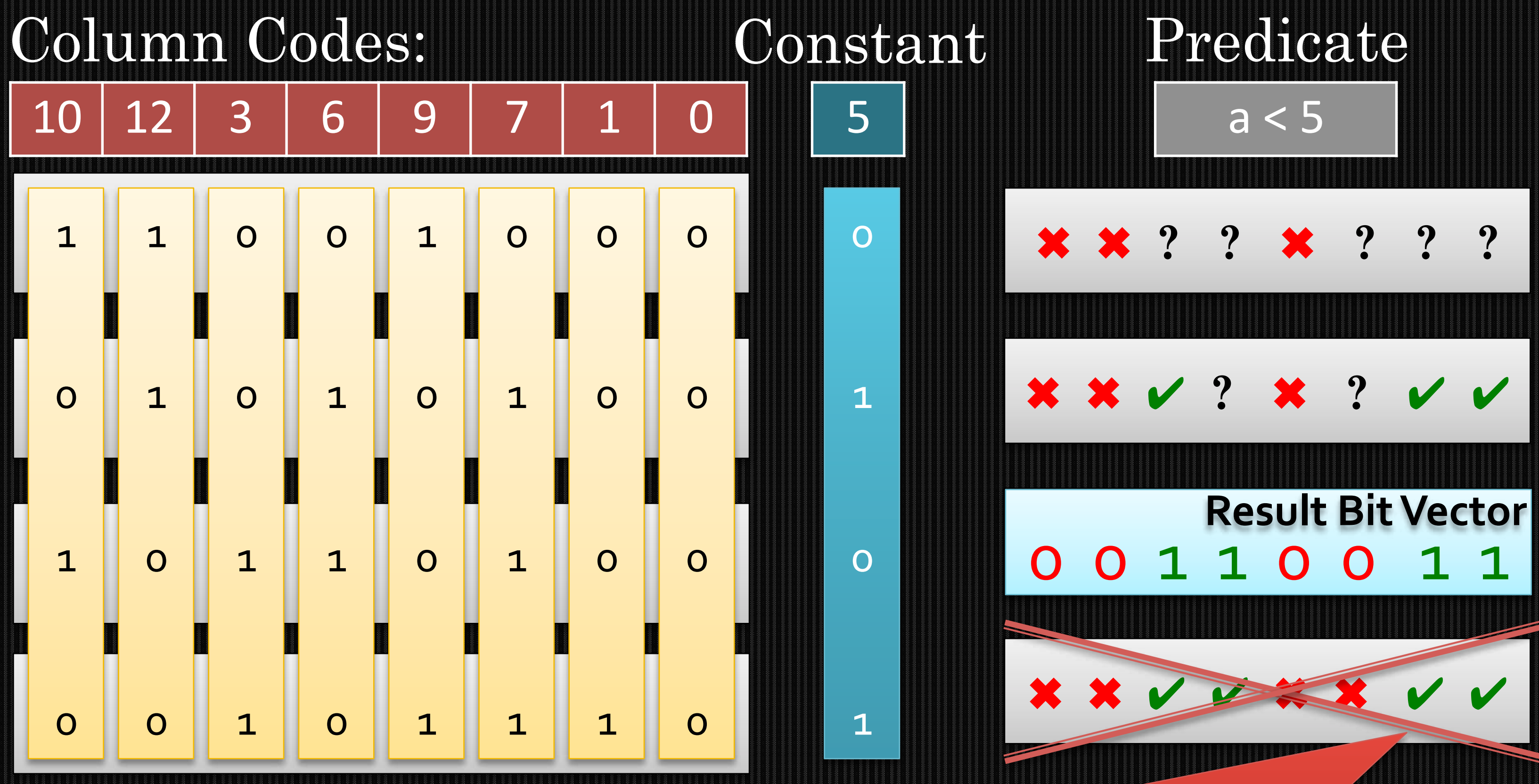
The first (most significant) bits of 8 consecutive codes

The second bits of 8 consecutive codes

The third bits of 8 consecutive codes

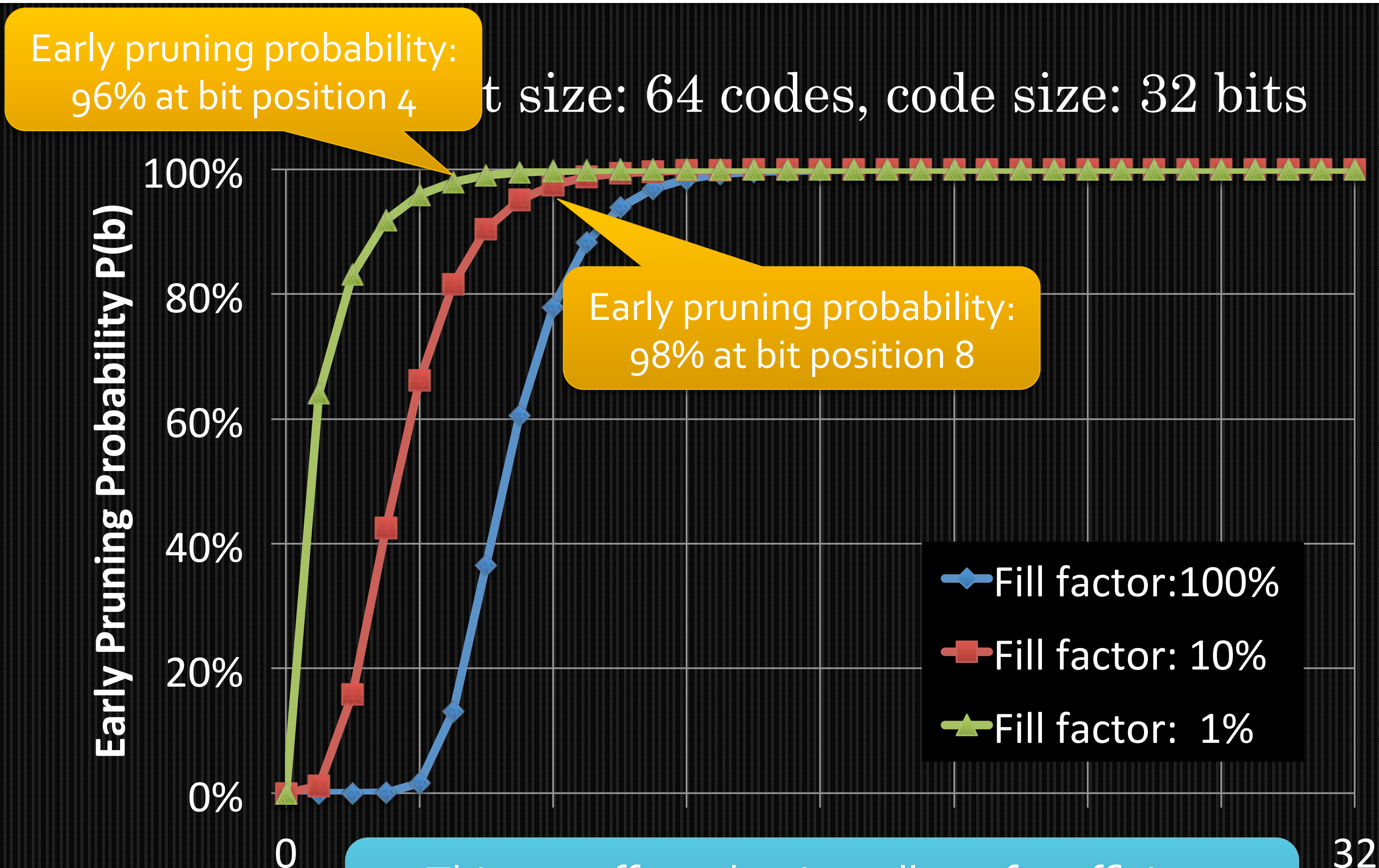
The last (least significant) bits of 8 consecutive codes

BitWeaving/V - early pruning



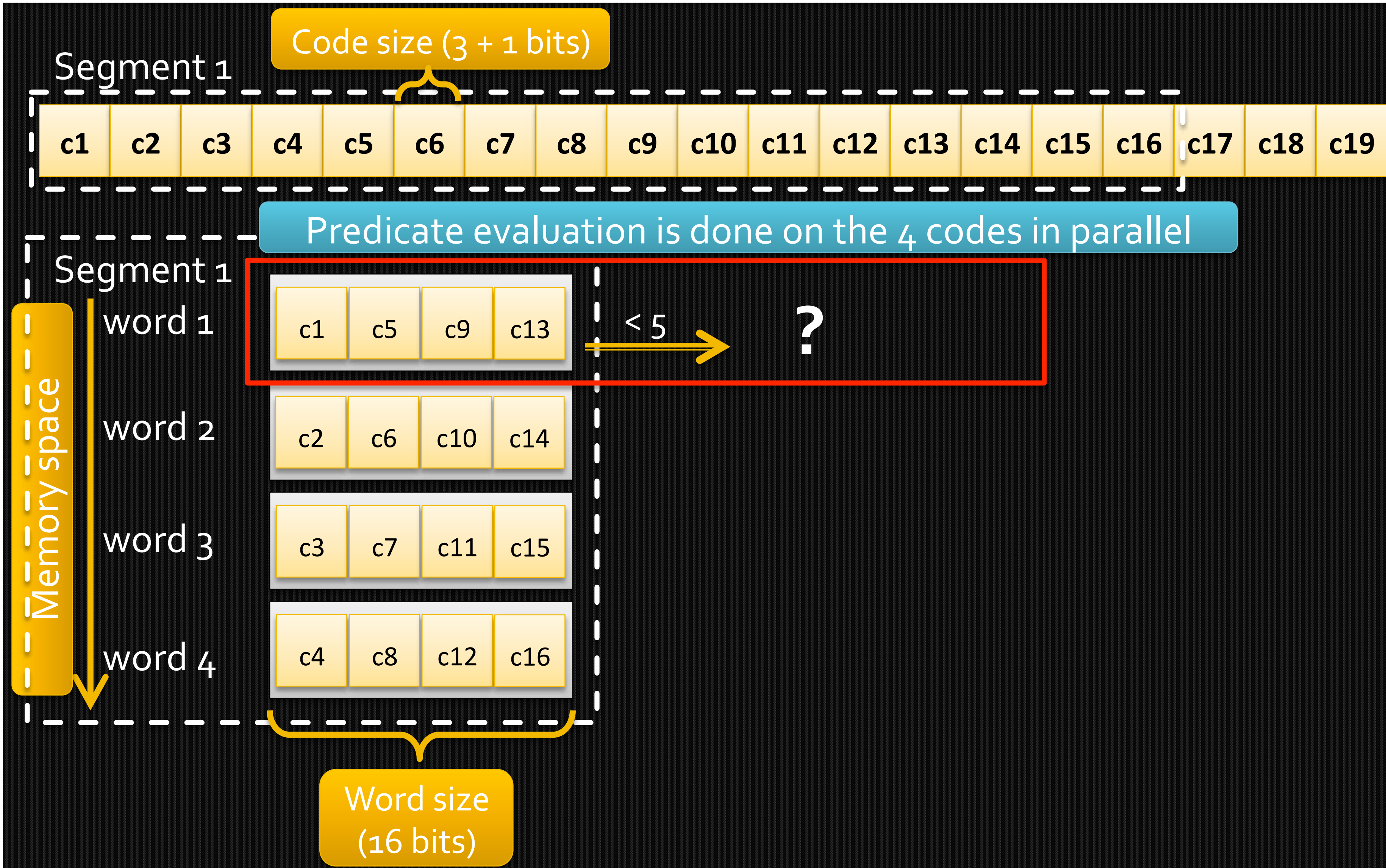
Early Pruning: terminate the predicate evaluation on a segment, when all results have been determined.

BitWeaving/V - Early Pruning Model



This cut-off mechanism allows for efficient evaluation of conjunction/disjunction predicates

BitWeaving/H - Example



BitWeaving/H: Less Than Predicate

Uses only 3 instructions! Without the delimiter, we would need ~12 instructions…

$$X = (c_1 c_5 c_9 c_{13})$$

$$Y = (5555)$$

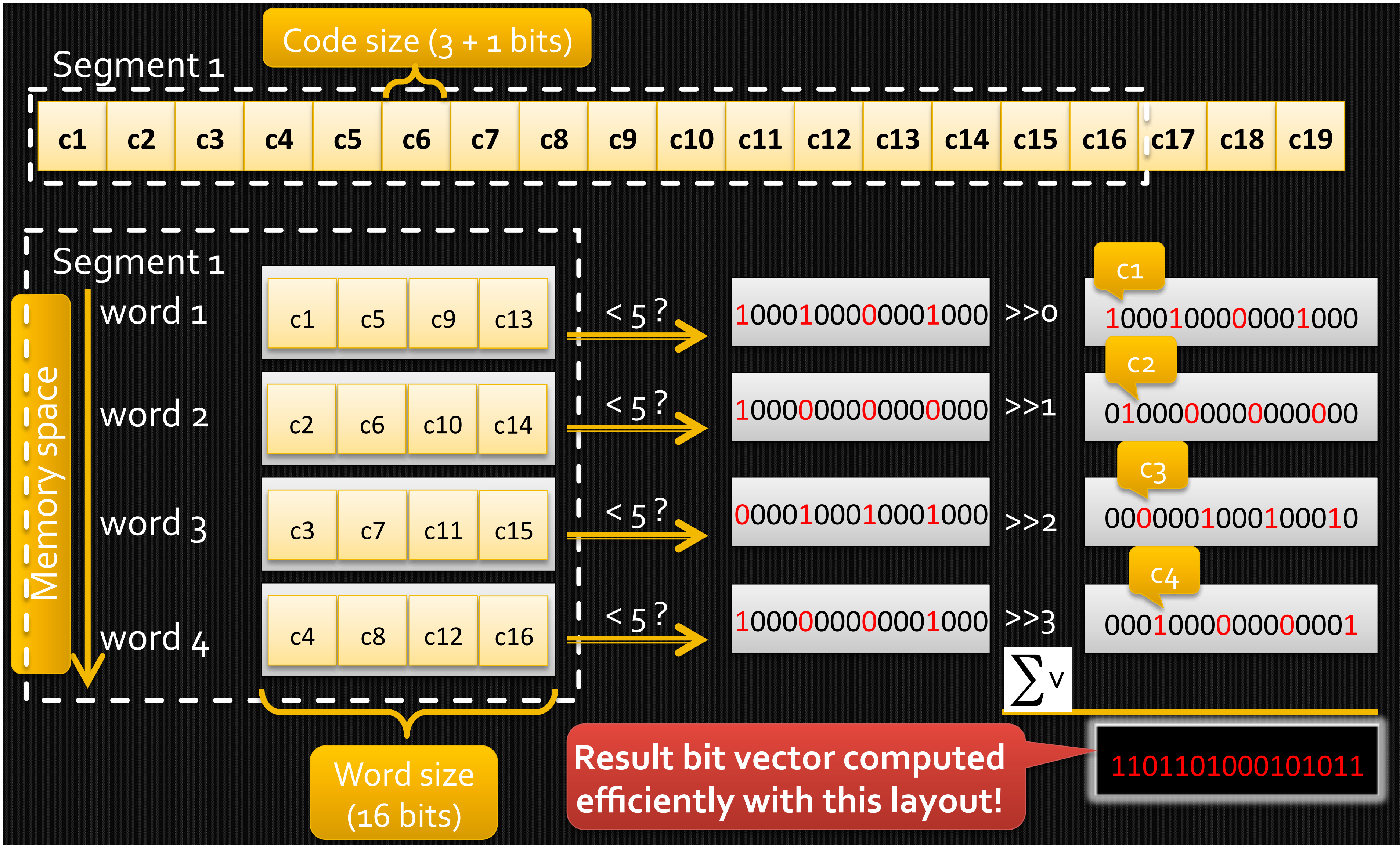
$$(Y + (X \oplus M1)) \wedge M2$$

$$M1 = 0111\ 0111\ 0111\ 0111$$
$$M2 = 1000\ 1000\ 1000\ 1000$$

	c5=7	c9=6	c13=2
0001	0111	0110	0010
0101	0101	0101	0101
1000	0000	0000	1000

Works for arbitrary code sizes & word sizes!

BitWeaving/H - Example



Evaluation

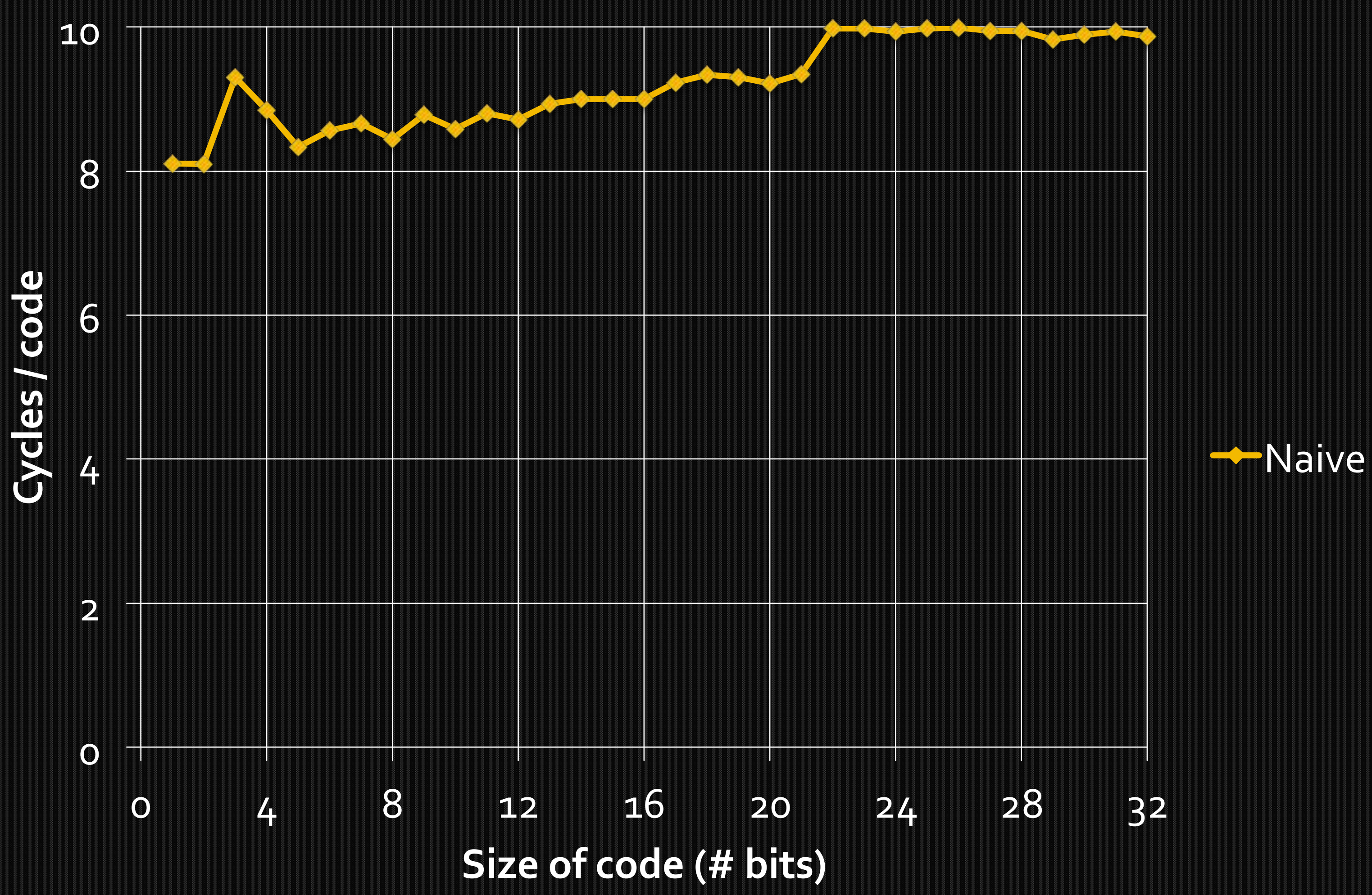
SYSTEM

- Intel Xeon X5650
  - 64 bits ALU
  - 128 bits SIMD
  - 12MB L3 Cache
- 24GB memory
- Single threaded execution

WORKLOAD

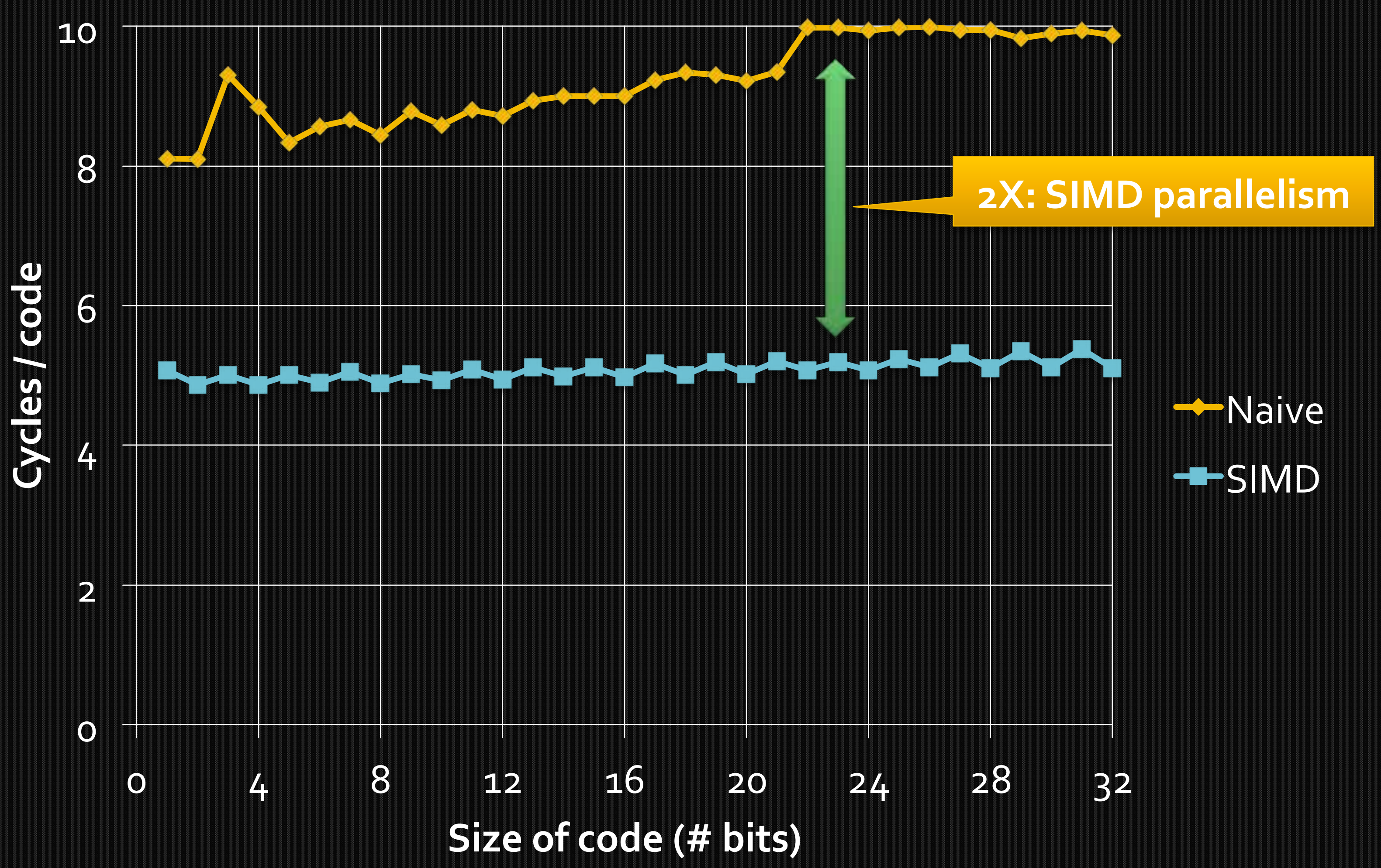
1. Synthetic
   - `SELECT COUNT(*) FROM R WHERE R.a < C`
   - 1 billion tuples
   - Uniform distribution
   - Selectivity: 10%
2. TPC-H @ SF=10
   - scan only with materialized join results

Evaluation: Micro-benchmark



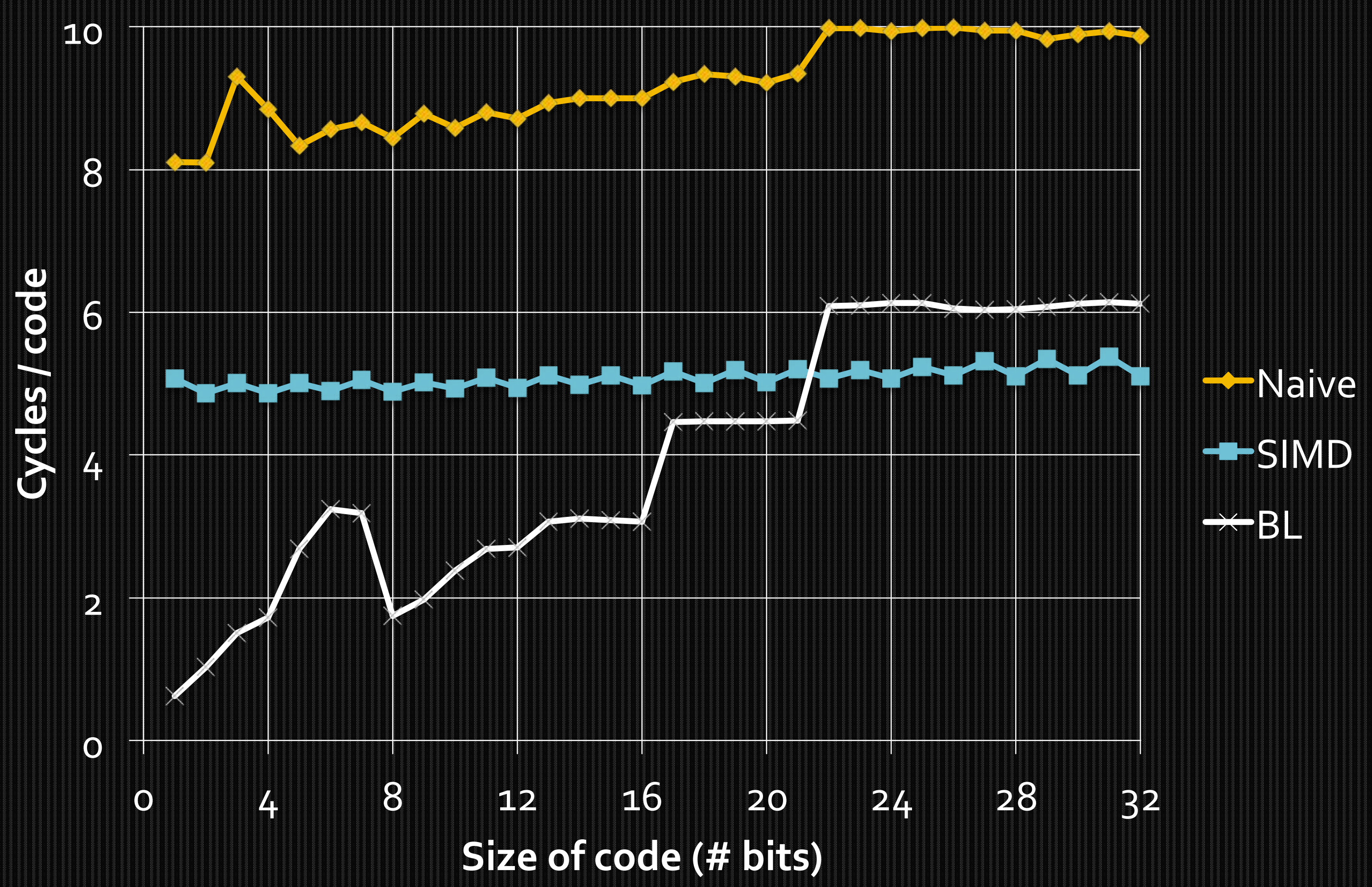
SIMD Paper: T. Willhalm, N. Popovici, Y. Boshmaf, H. Plattner, A. Zeier, and J. Schaffner.
SIMD-Scan: Ultra fast in-memory table scan using on-chip vector processing units. PVLDB'09

Evaluation: Micro-benchmark

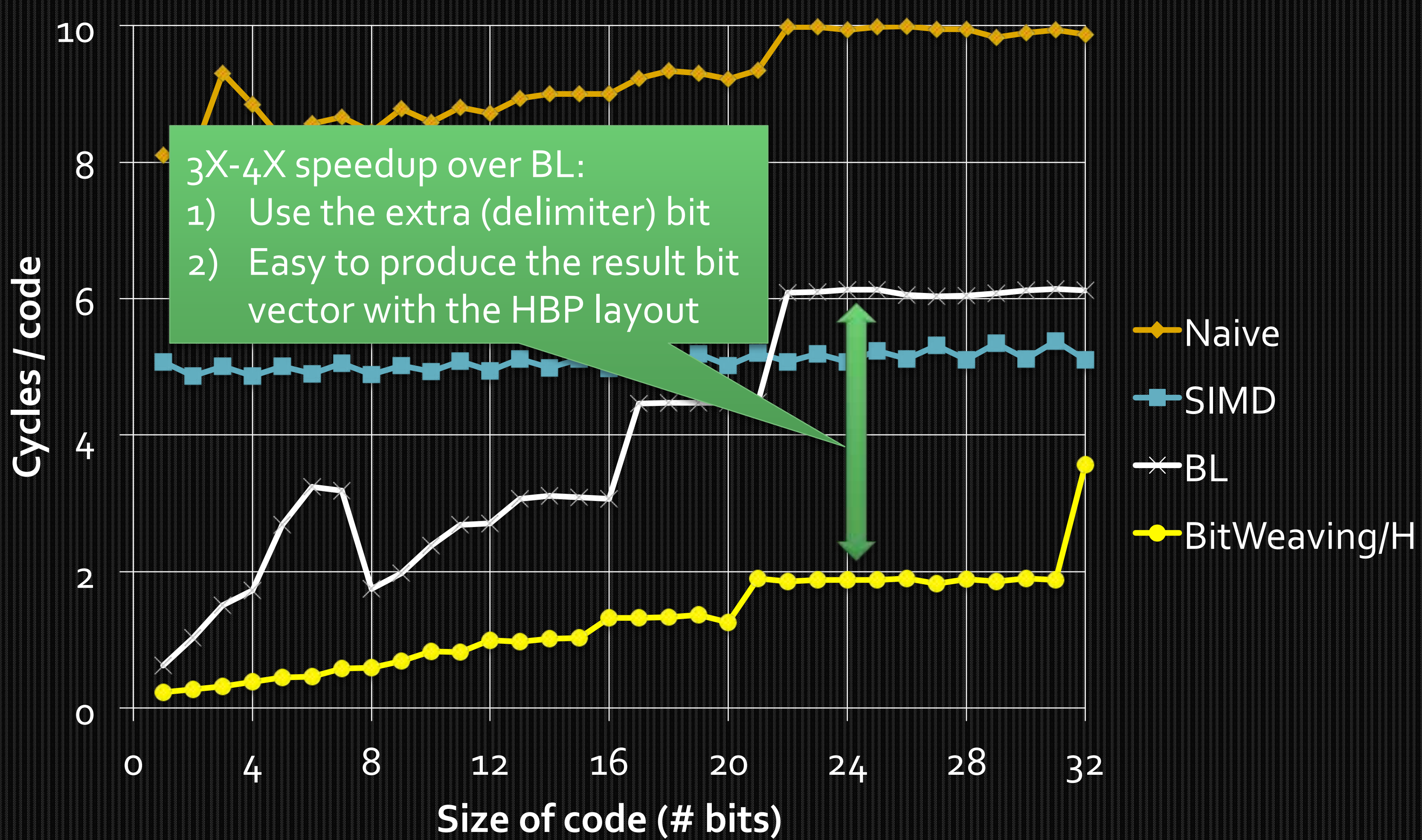


Blink Paper: R. Johnson, V. Raman, R. Sidle, and G. Swart. Row-wise parallel predicate evaluation. VLDB'08

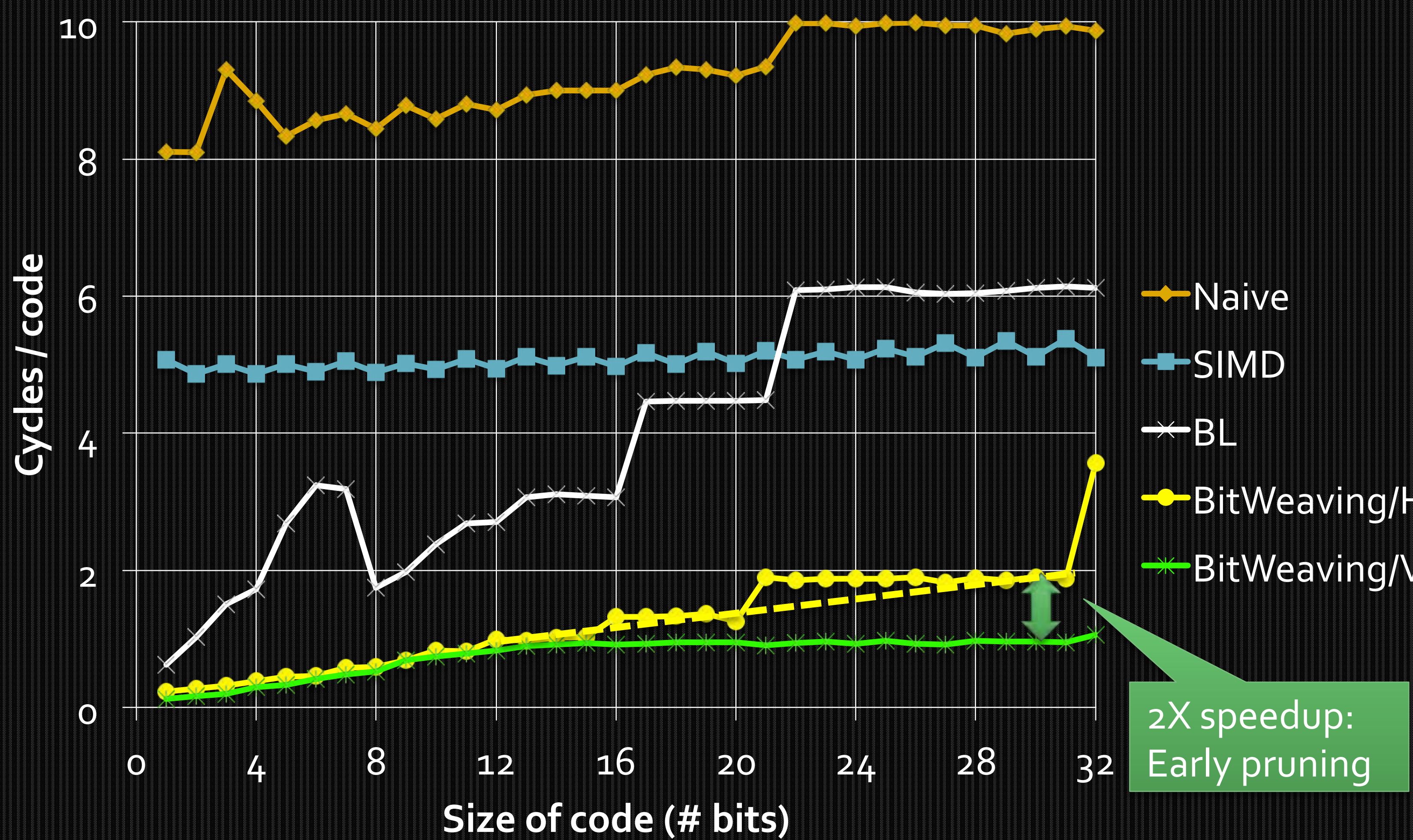
Evaluation: Micro-benchmark



Evaluation: Micro-benchmark



Evaluation: Micro-benchmark



Many more experiments in the paper

WideTable

Customer

cid	cname	gender	address
1	Andy	M	100 Main st.
2	Kate	F	20 10th blvd.
3	Bob	M	300 5th ave.

⋈

Product

pid	pname
1	Milk
2	Coffee
3	Tea

⋈

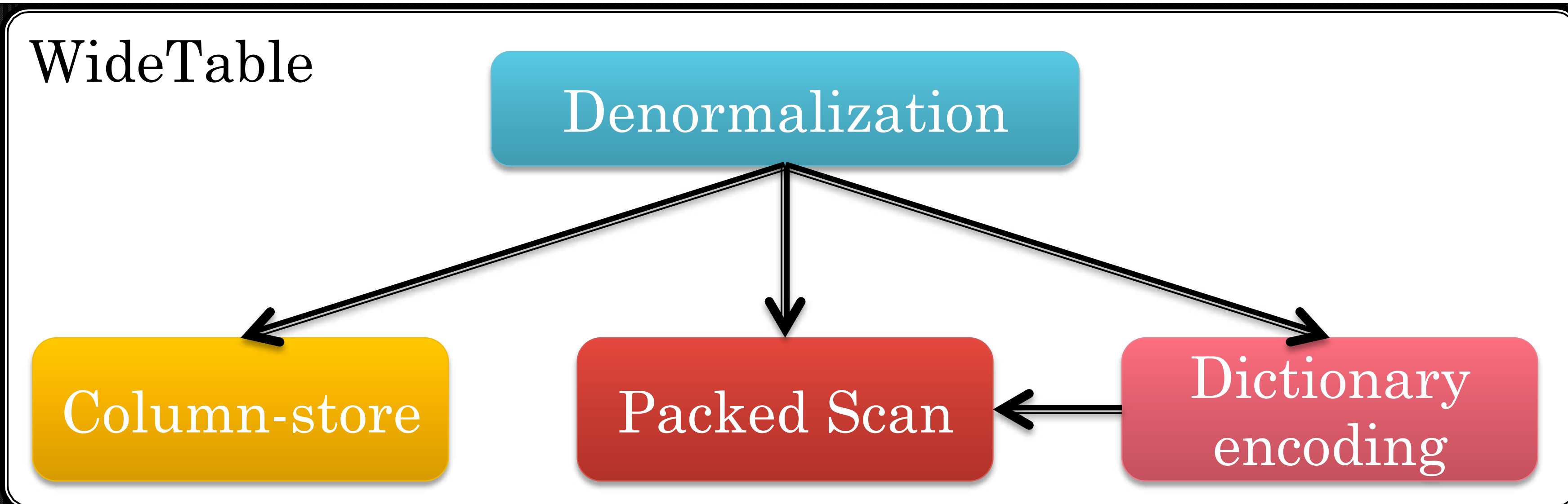
Buy

cid	pid	status
1	2	S
2	2	F
3	3	S
1	2	S

cid	cname	gender	address	pid	pname	status
1	Andy	M	100 Main st.	2	Coffee	S
2	Kate	F	20 10th blvd.	2	Coffee	F
3	Bob	M	300 5th ave.	3	Tea	S
1	Andy	M	100 Main st.	2	Coffee	S
NULL	NULL	NULL	NULL	1	Milk	NULL

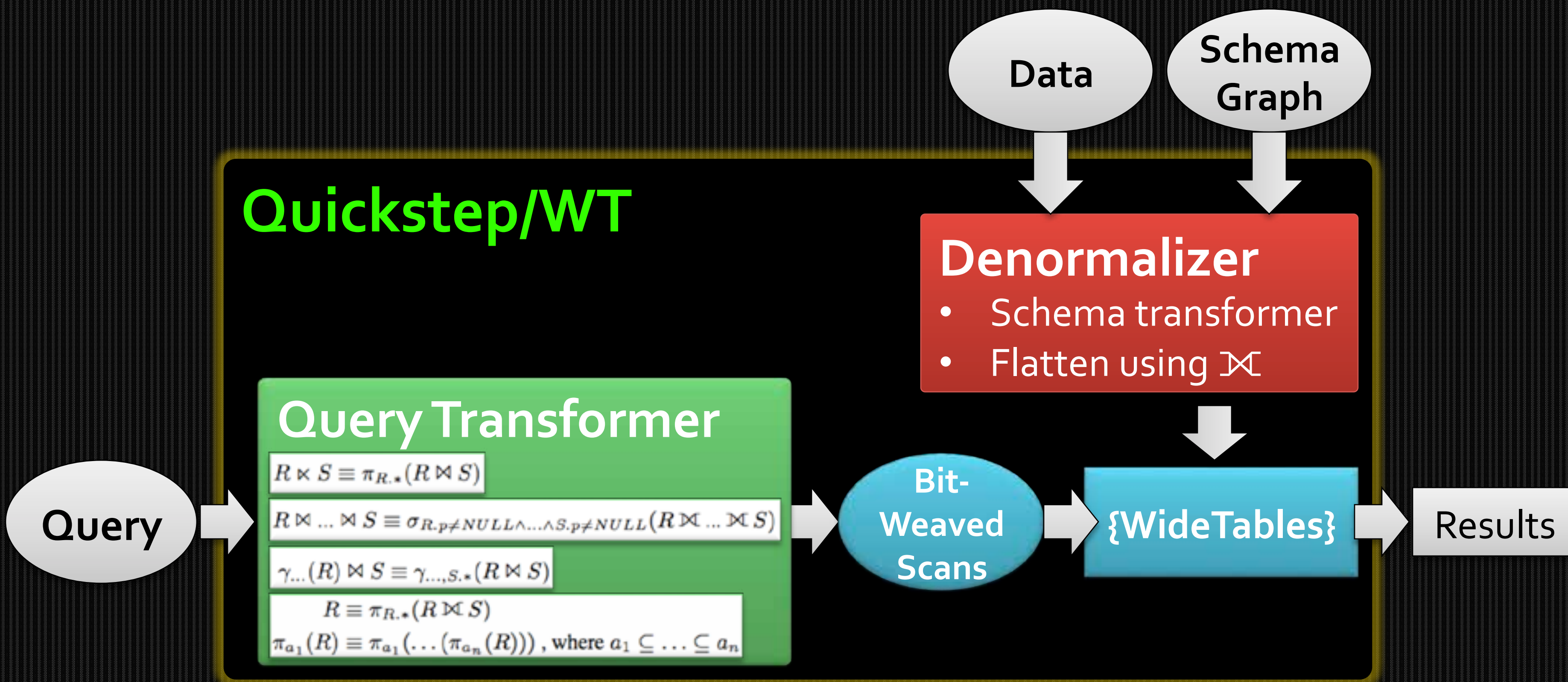
WideTable

WideTable



Now we can run analytical workloads (e.g. TPC-H) using simple BitWeaved scans

WideTable Techniques



Schema Graph

Schema Graph **Schema Tree**

WideTable = (Region ⋈ Nation ⋈ Customer) ⋈ (Region ⋈ Nation ⋈ Product ⋈ Buy)

SMW = {WideTables}
e.g. for TPC-H, SMW={lineItemWT, ordersWT, partsuppWT, customerW

TPC-H Queries

TPC-H Queries	Joins	Nested Queries	Non-FK joins	WideTable
Q1, Q6				LineitemWT
Q3, Q5, Q7-Q10, Q12, Q14, Q19	×			LineitemWT
Q4, Q15, Q17, Q18, Q20	×	×		LineitemWT
Q21	×	×	×	---
Q2, Q11, Q16	×	×		PartsuppWT
Q13	×			OrdersWT
Q22	×	×		OrdersWT

Evaluation

SYSTEM

- Intel Xeon E5-2620 × 2
- 2.0 GHz
- 12 cores / 24 threads
- 15MB L3 Cache
- 32G. 1600MHz DDR3

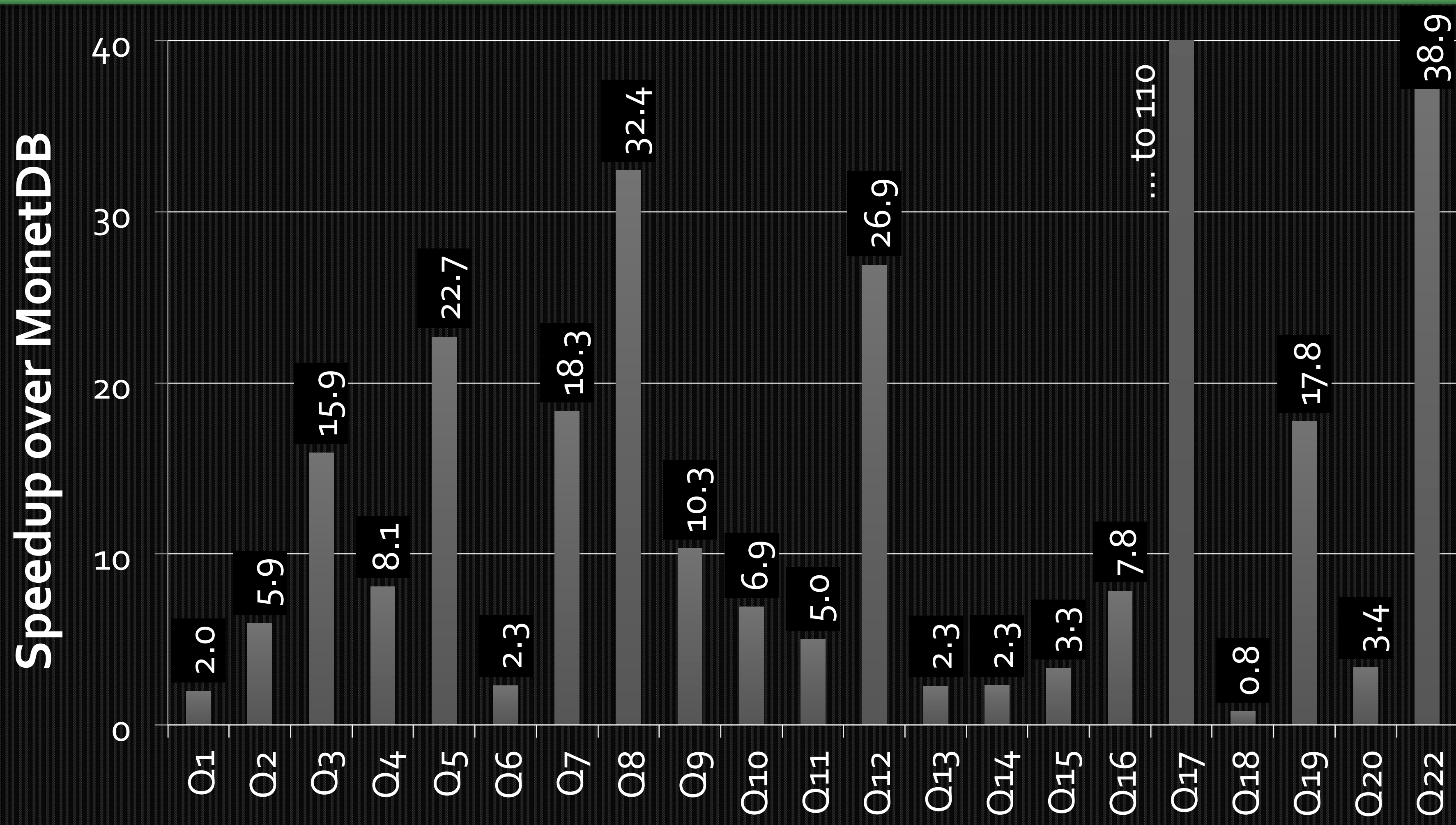
BENCHMARK

- SF: 10 (~10GB)
- SMW =

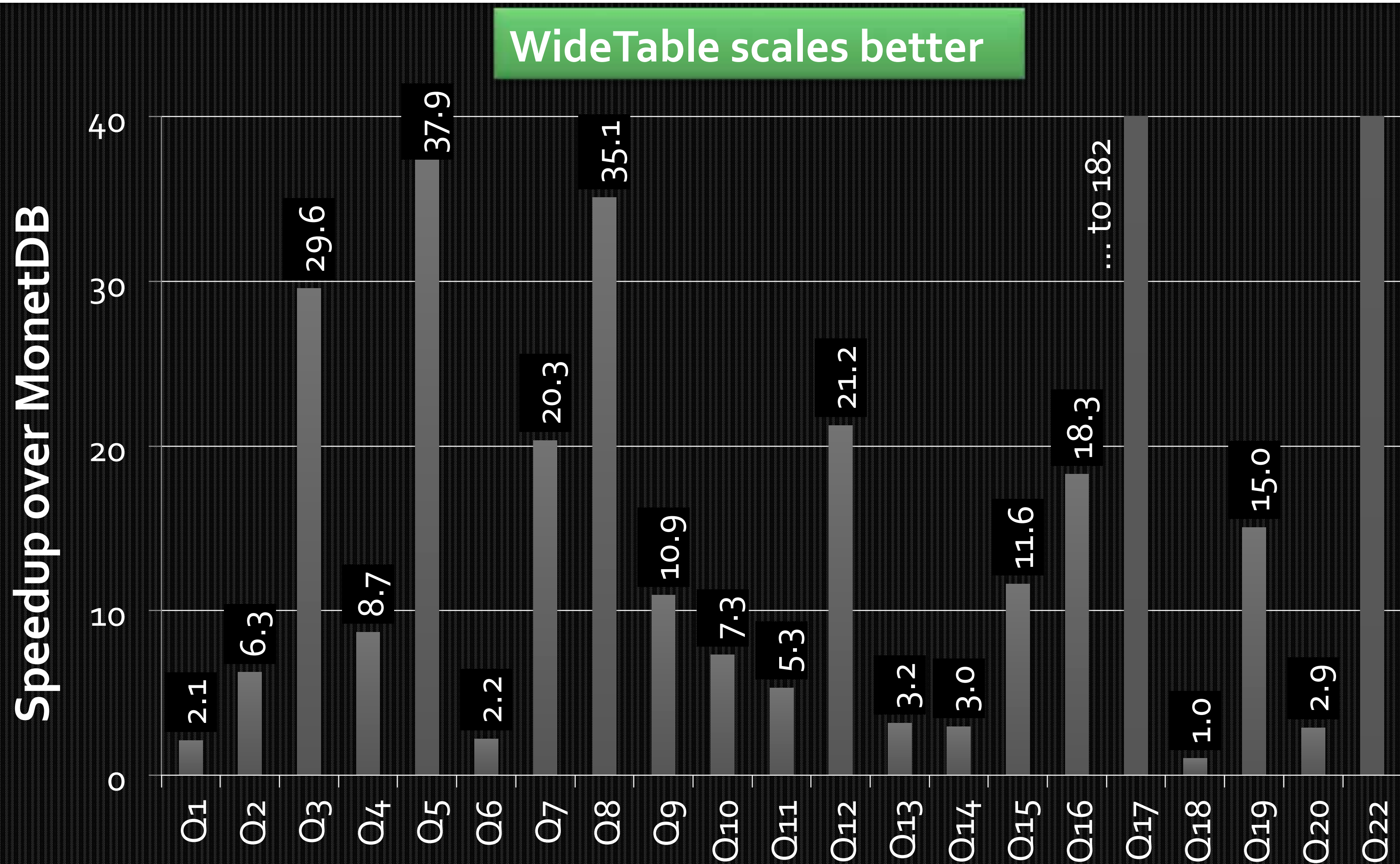
lineItemWT	5.4 GB
ordersWT	0.7 GB
partsuppWT	0.2 GB
customerWT	0.05 GB
dictionaries	0.8 GB
filter columns	1.3 GB
TOTAL	8.5GB

Speedup over MonetDB: Single Thread

WideTable over 10X faster than MonetDB for about half of the 21 queries



Speedup over MonetDB: 12 Threads



Conclusions and Future Work

Transformative architectural changes at all levels (CPU, memory subsystem, I/O subsystem) is underway

Need to rethink data processing kernels

- Run @ current bare metal speed

Need to think of hardware software co-design

Thanks!

Craig Chasseur | Anusha Dasarakothapalli | Harshad Deshmukh | Jing Fan | Yinan Li | James Paton | Navneet Potti | Sangmin Shin | Qiang Zeng

The Quickstep Team

NSF

Google | Microsoft | ORACLE | SAMSUNG

Funding