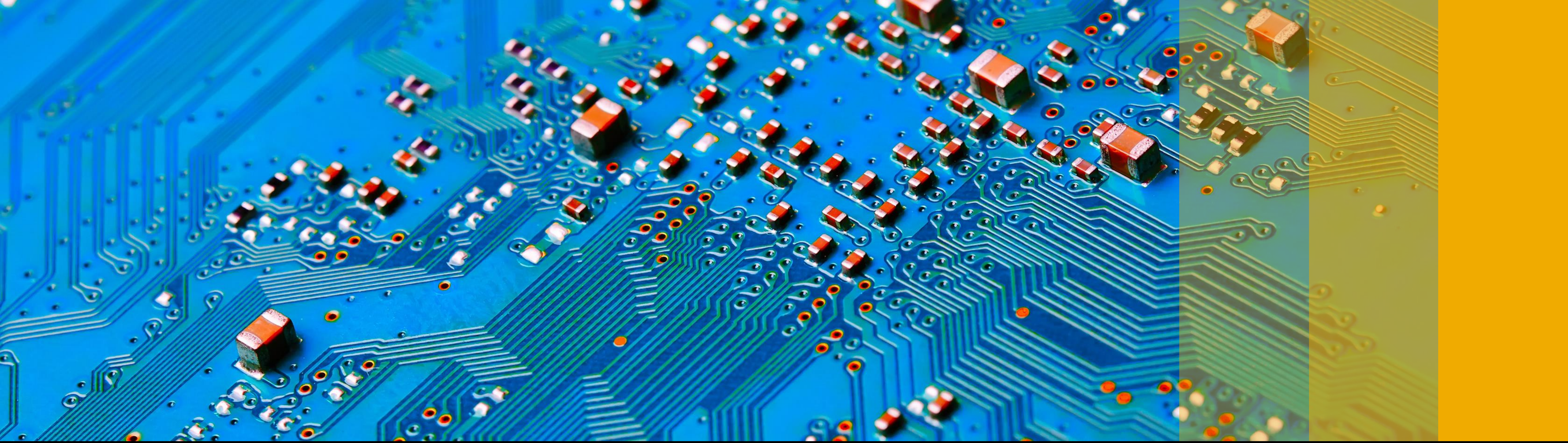


Leveraging Modern Hardware in SAP HANA
Present and Future

Dr.-Ing. Ismail Oukid (SAP)
January 21, 2019

PUBLIC

Run Simple

Agenda

Present

- Single Instruction Multiple Data (SIMD)
- Hardware Transactional Memory (HTM)
- Non-Volatile Memory (NVM)

Future

- Graphic-Processing Units (GPUs)
- Field-Programmable Gate Arrays (FPGAs)
- Software-Defined Memory Coherency

SIMD Single Instruction Multiple Data

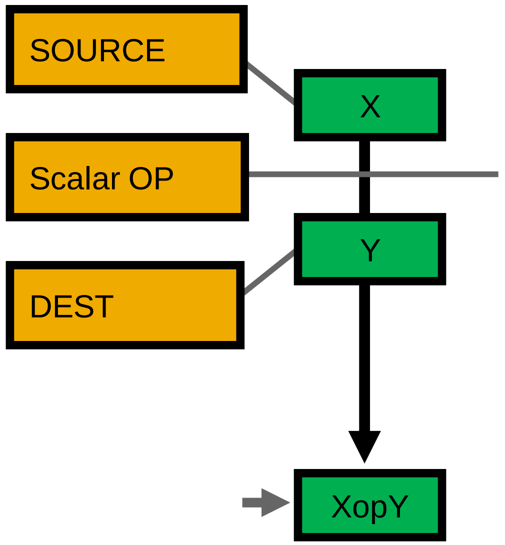
Single Instruction Multiple Data (SIMD)

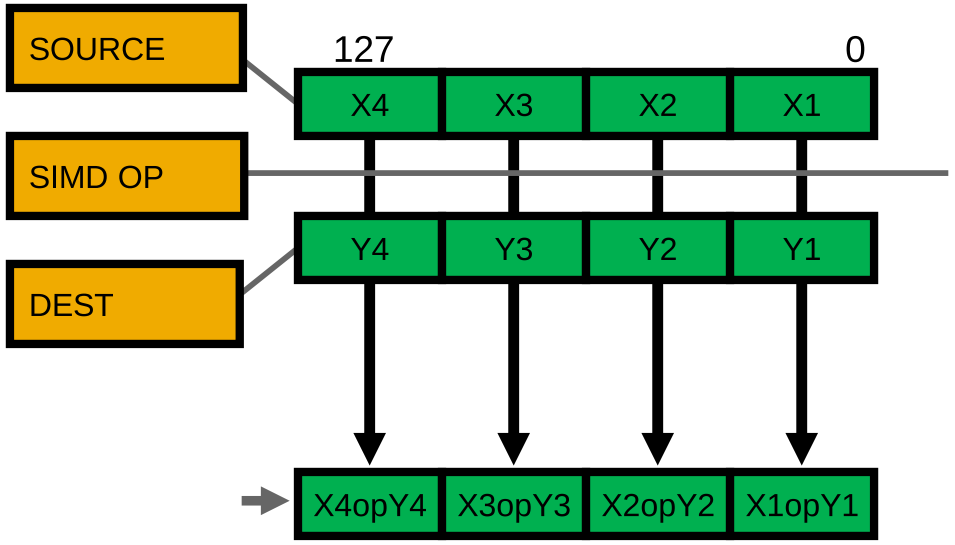
Scalar processing

- traditional mode
- one instruction produces one result

SIMD processing

- with Intel® SSE, AVX
- one instruction produces multiple results



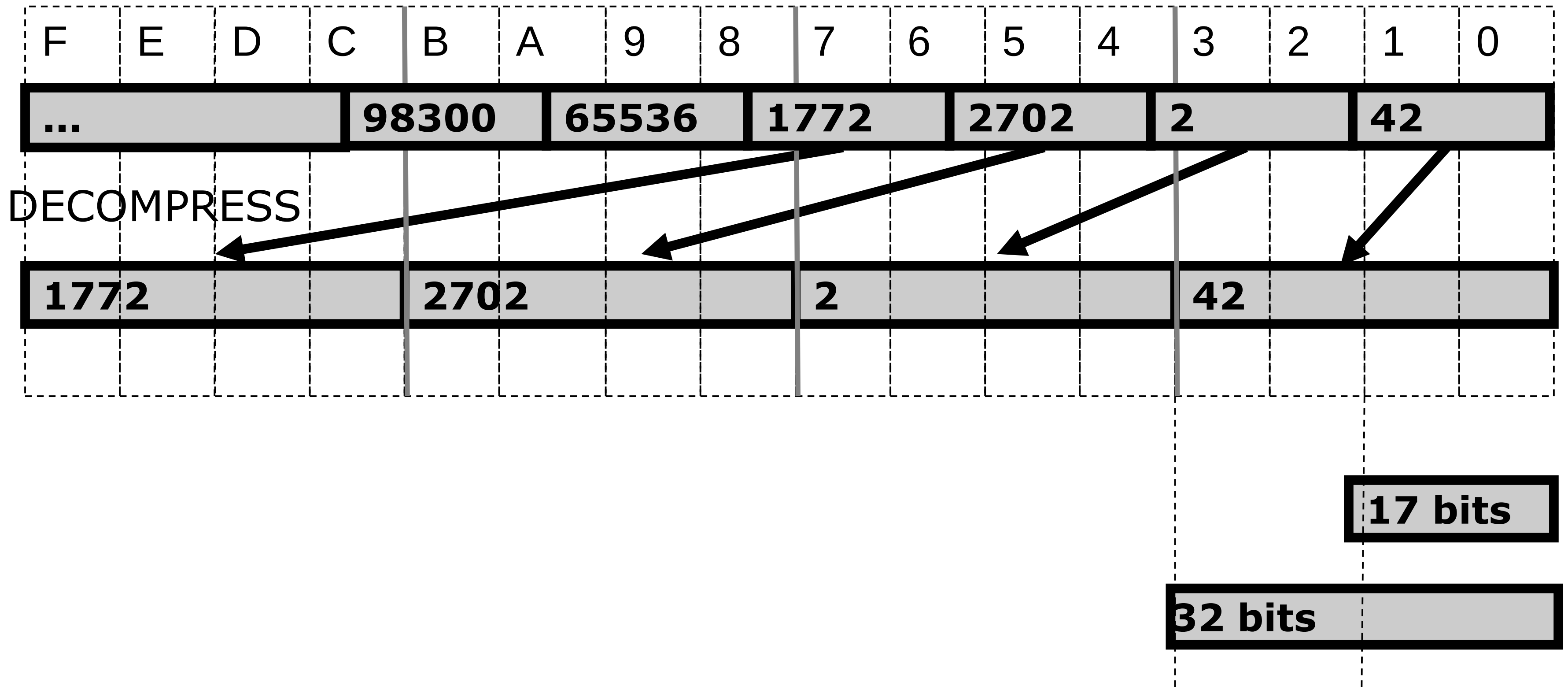


Use-Case: Pack Integers in Bit-Fields

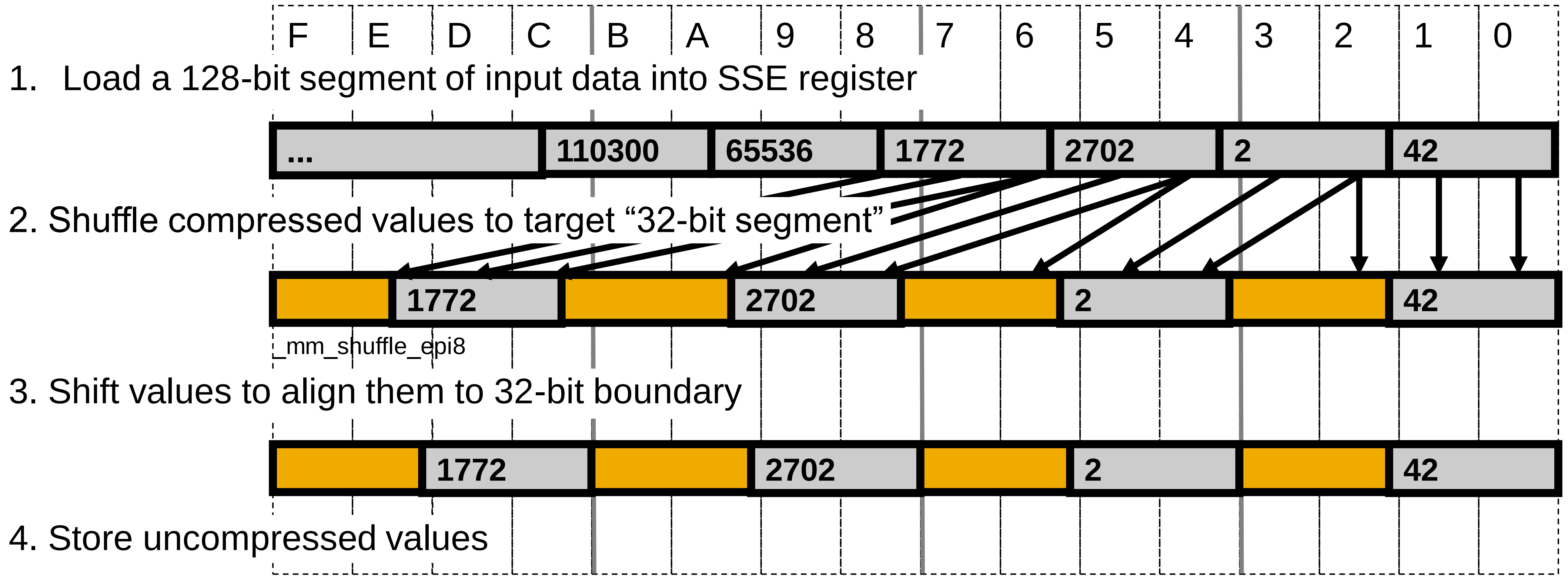
Example: Packed 17-bit fields

Integers in the range **[0, 100000]** need only **17 bits**

Idea: Store only 17 bits (saving 15 bits per value)

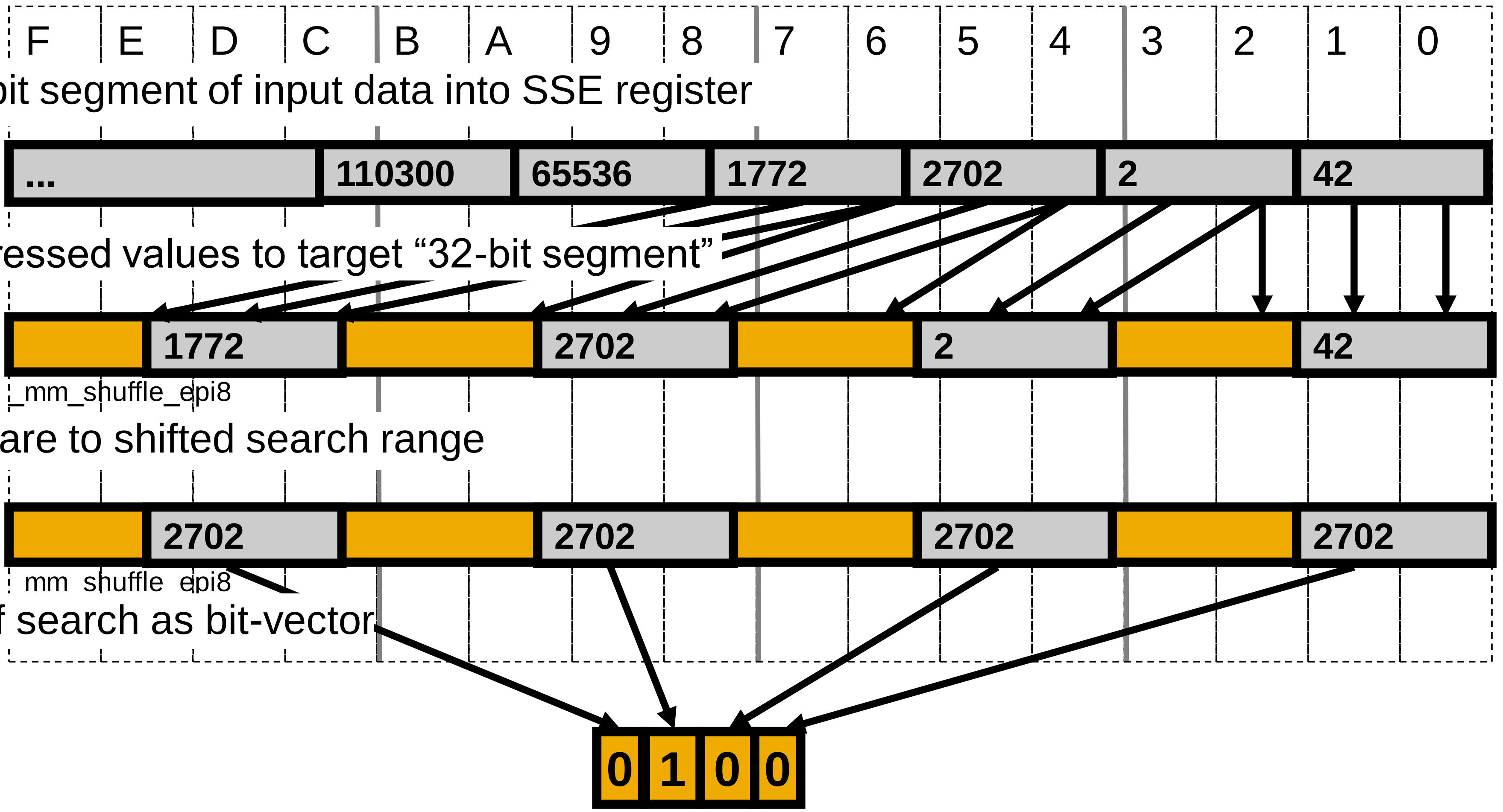


Decompress Unaligned Bit Fields (Example: Packed 17-Bit Fields)

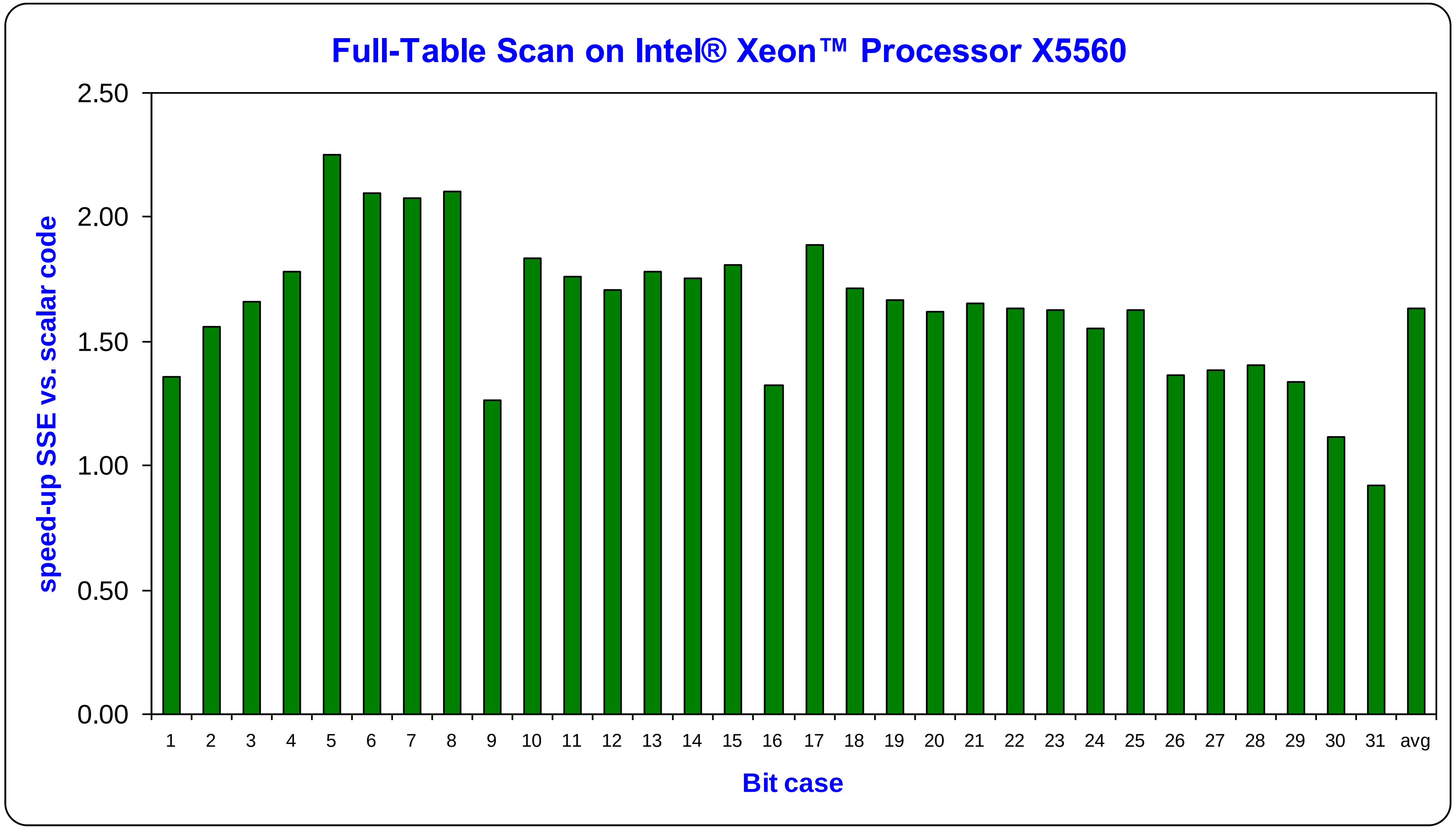

F
E
D
C
B
A
9
8
7
6
5
4
3
2
1
0
1. Load a 128-bit segment of input data into SSE register
...
110300
65536
1772
2702
2
42
2. Shuffle compressed values to target “32-bit segment”
1772
2702
2
42
_mm_shuffle_epi8
3. Shift values to align them to 32-bit boundary
1772
2702
2
42
4. Store uncompressed values

Search Unaligned Bit Fields (Example: Packed 17-Bit Fields)

1. Load a 128-bit segment of input data into SSE register
2. Shuffle compressed values to target "32-bit segment"
3. Parallel compare to shifted search range
4. Store result of search as bit-vector



Full-table Scan is 1.63x faster with SSE



Source: Willhalm et al. SIMD-scan: ultra fast in-memory table scan using on-chip vector processing units. *PVLDB 2009*

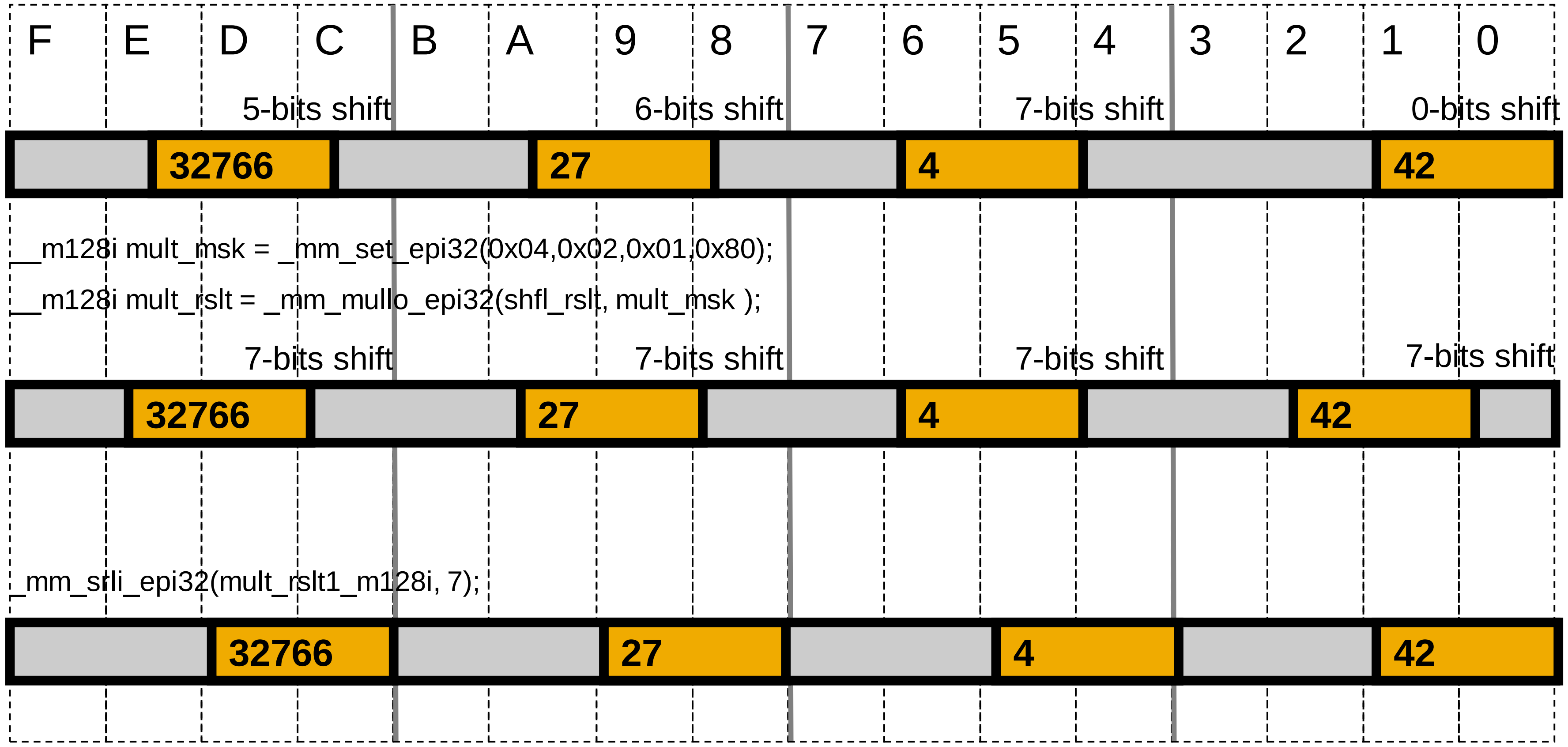
Problem: No “vector-vector shift” in SSE

Hint: How do you implement a fast multiplication by “2”?

Solution: Use multiplication for shifting

Multiply to shift left

Shift right



```
__m128i mult_msk = _mm_set_epi32(0x04,0x02,0x01,0x80);
__m128i mult_rslt = _mm_mullo_epi32(shfl_rslt, mult_msk );
```

```
_mm_srli_epi32(mult_rslt1_m128i, 7);
```

Unpacking of Bit-Fields with Intel® AVX2

Pseudo Code	SSE 4.1	AVX2

vector load v
from input array
	`movdqu`
	`x8(%r10,%rcx,1),%xmm6`
		`vmovdqu xmm8, xmmword ptr[rax+rcx*1+0x11]`
		`vinserti128 ymm9, ymm8, xmmword ptr`
		`[rax+rcx*1+0x19], 0x01`

byte shuffle v	`pshufb %xmm1,%xmm6`	`vpshufb ymm10, ymm9, ymm1`

vector shift v	`pmulld %xmm2,%xmm6`
	`psrld  $0xe,%xmm6`
		`vpsrlvd ymm11, ymm10, ymm0`

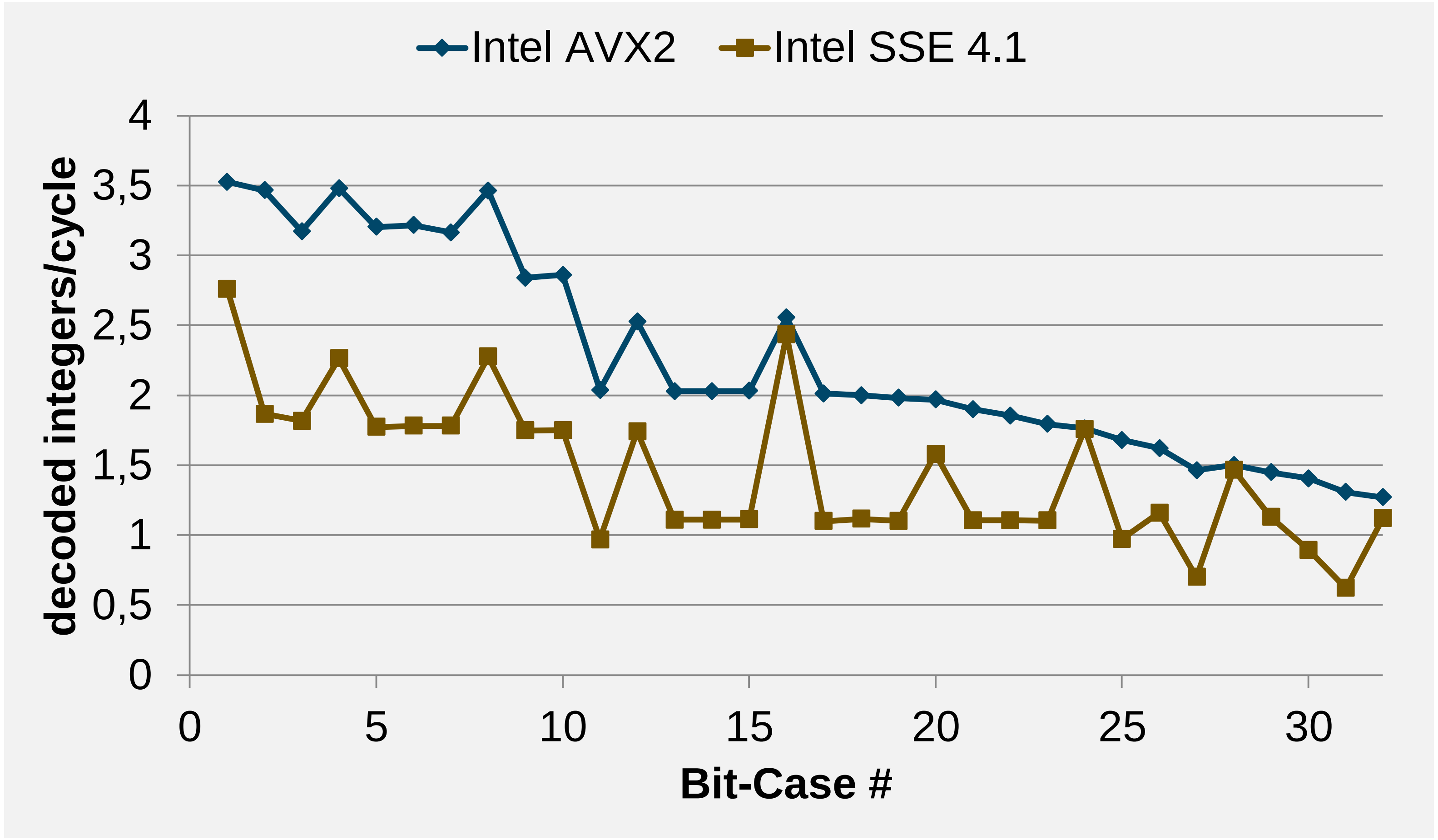
vector and v	`pand   %xmm0,%xmm6`	`vpand ymm12, ymm11, ymm2`
vector store v in output array	`movdqa %xmm6,0x10(%r8)`	`vmovdqu ymmword ptr [r8+0x20], ymm12`

New variable shift instruction

Double the number of data elements vs. Intel® SSE

Implementation takes advantage of new variable shift

Intel® AVX2 Unpacking - Performance



Bit-field unpacking runs up to 1.6x faster on average with Intel ® AVX2

Source: Willhalm et al. Vectorizing Database Column Scans with Complex Predicates. ADMS@VLDB 2014.

Intel® Advanced Vector Extensions 512

Expands register size to 512 bits

New mask registers:

- Results of comparisons
- Masking operations

New instructions:

- vpcompressd – extract selected DWORDs
- Scatter – distribute values

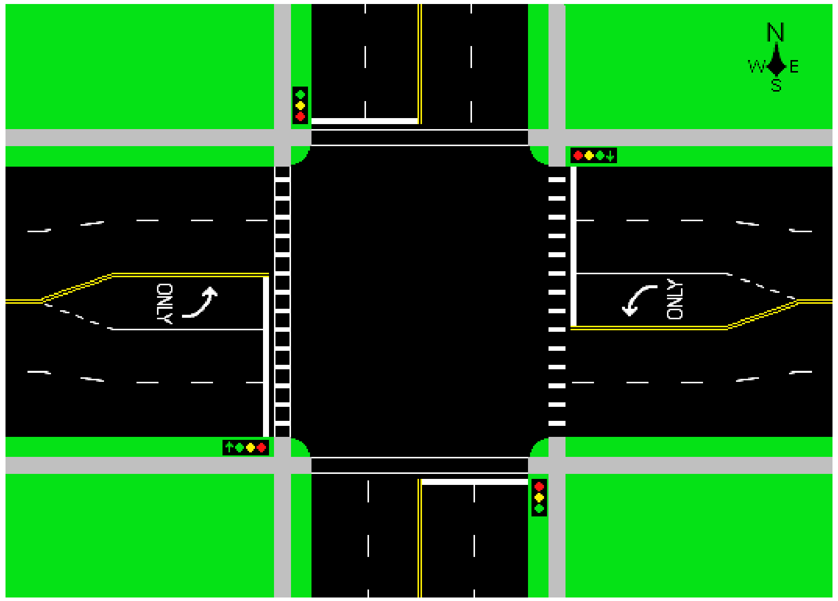
Using Mask Registers for Predication

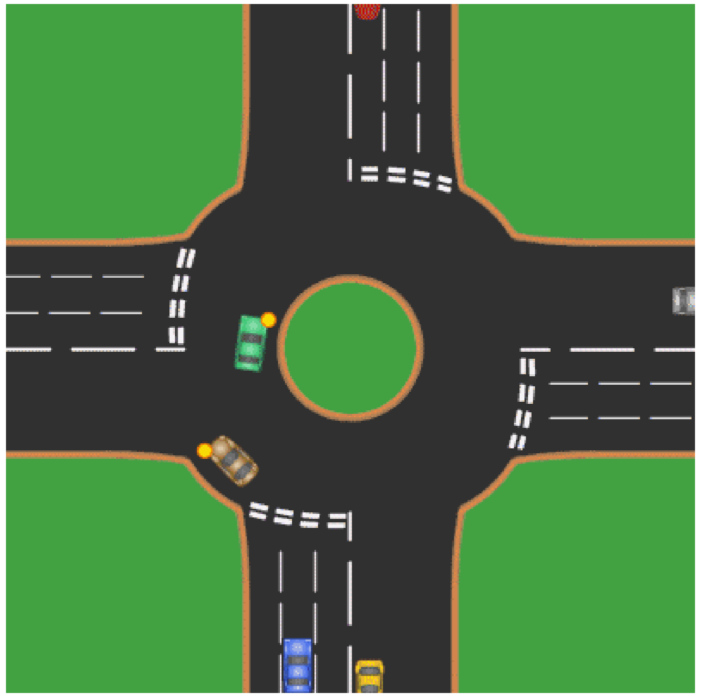
```
if (v5<v6) {v1 += v3;}

v5 = 0 4 7 8 3 9 2 0 6 3 8 9 4 5 0 1
v6 = 9 4 8 2 0 9 4 5 5 3 4 6 9 1 3 0
vpcmpd k7, k0, zmm5, zmm6, 0x6
k7 = 1 0 1 0 0 0 1 1 0 0 0 0 1 0 1 0

v3 = 5 6 7 8 5 6 7 8 5 6 7 8 5 6 7 8
v1 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
vaddpd v1, k7, v1, v3
v1 = 6 1 8 1 1 1 8 9 1 1 1 1 6 1 8 1
```

- Instructions operate only on selected elements
- Enables vectorization of “branchy” code, e.g. processing of NULL values

HTM Hardware Transactional Memory

Locks are blocking like traffic lights





Picture idea from Dave Boutcher

Serialize execution only when necessary

Intel® Transactional Synchronization Extensions

Transactionally execute lock-protected critical sections

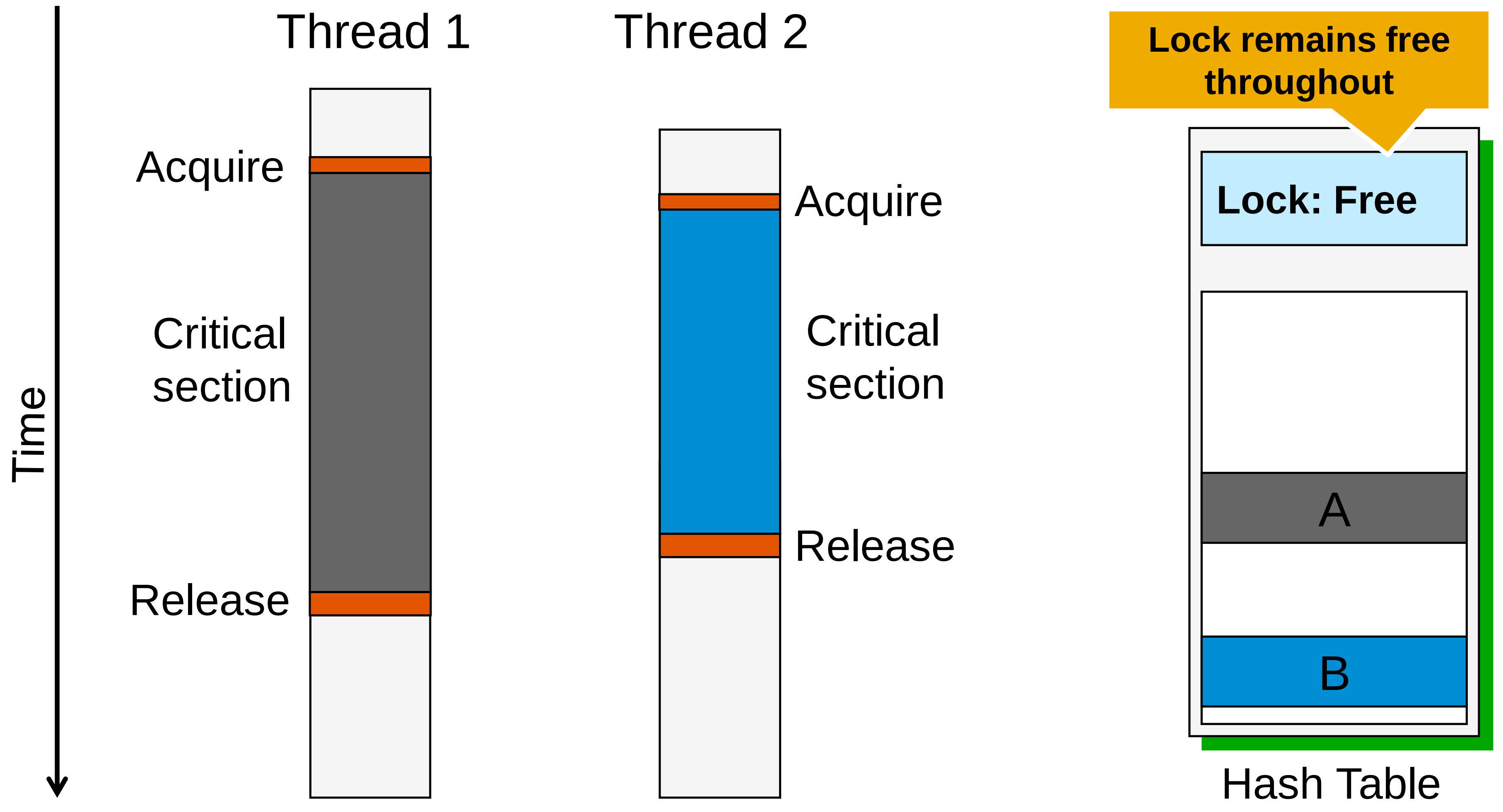
Execute without acquiring lock

- Expose hidden concurrency

Hardware manages transactional updates – All or None

- Other threads can’t observe intermediate transactional updates
- If lock elision cannot succeed, restart execution & acquire lock

Intel® Transactional Synchronization Extensions



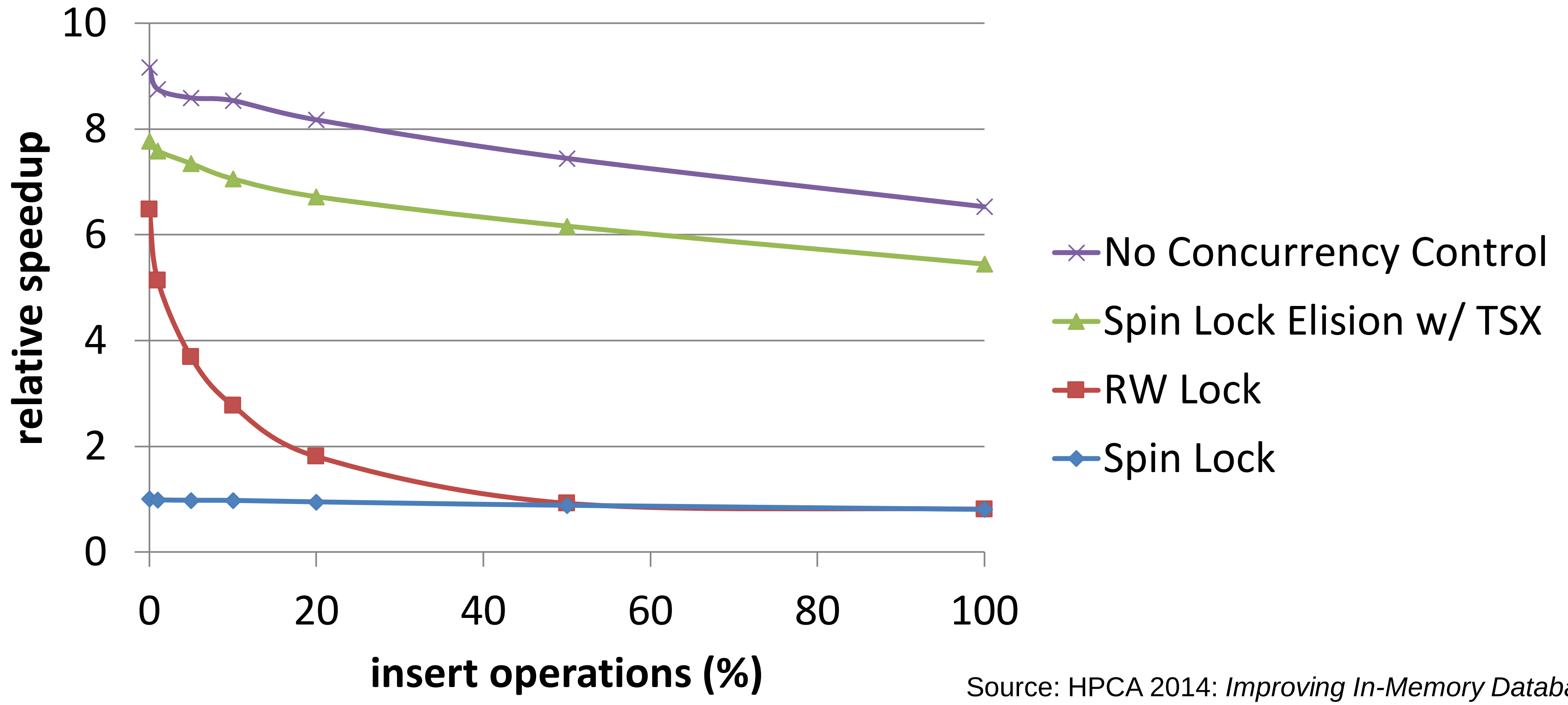
Transactions keep read and write sets in L1 cache

No Serialization and no communication if no data conflicts

Initial Analysis: B+Tree

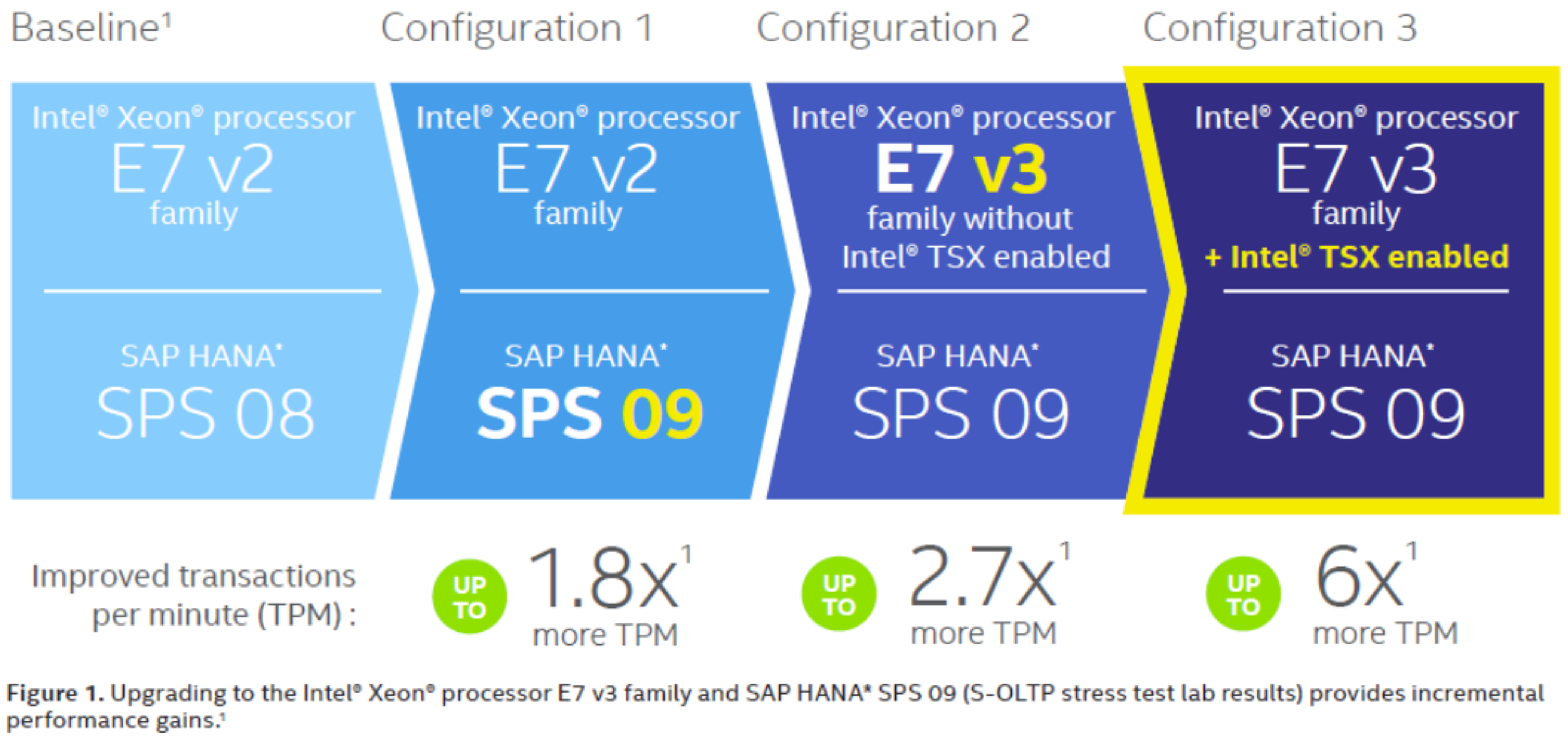
Intel TSX provides significant gains with no application changes

- Outperforms RW lock on read-only queries
- Significant gains with increasing inserts (6x for 50%)



Source: HPCA 2014: *Improving In-Memory Database Index Performance with Intel® Transactional Synchronization Extensions.* Tomas Karnagel et al.

Up to 2x Performance Boost in OLTP When Running SAP HANA with TSX



Baseline[1]	Configuration 1	Configuration 2	Configuration 3
Intel® Xeon® processor E7 v2 family	Intel® Xeon® processor E7 v2 family	Intel® Xeon® processor E7 v3 family without Intel® TSX enabled	Intel® Xeon® processor E7 v3 family + Intel® TSX enabled
SAP HANA* SPS 08	SAP HANA* SPS 09	SAP HANA* SPS 09	SAP HANA* SPS 09
Improved transactions per minute (TPM) :	UP TO 1.8x[1] more TPM	UP TO 2.7x[1] more TPM	UP TO 6x[1] more TPM

Figure 1. Upgrading to the Intel® Xeon® processor E7 v3 family and SAP HANA* SPS 09 (S-OLTP stress test lab results) provides incremental performance gains.[1]

Source: http://www.intel.com/content/dam/www/public/us/en/documents/solution-briefs/sap-hana-real-time-analytics-solution-brief.pdf

Disclaimer: Performance estimates were obtained prior to implementation of recent software patches and firmware updates intended to address exploits referred to as "Spectre" and "Meltdown." Implementation of these updates may make these results inapplicable to your device or system.

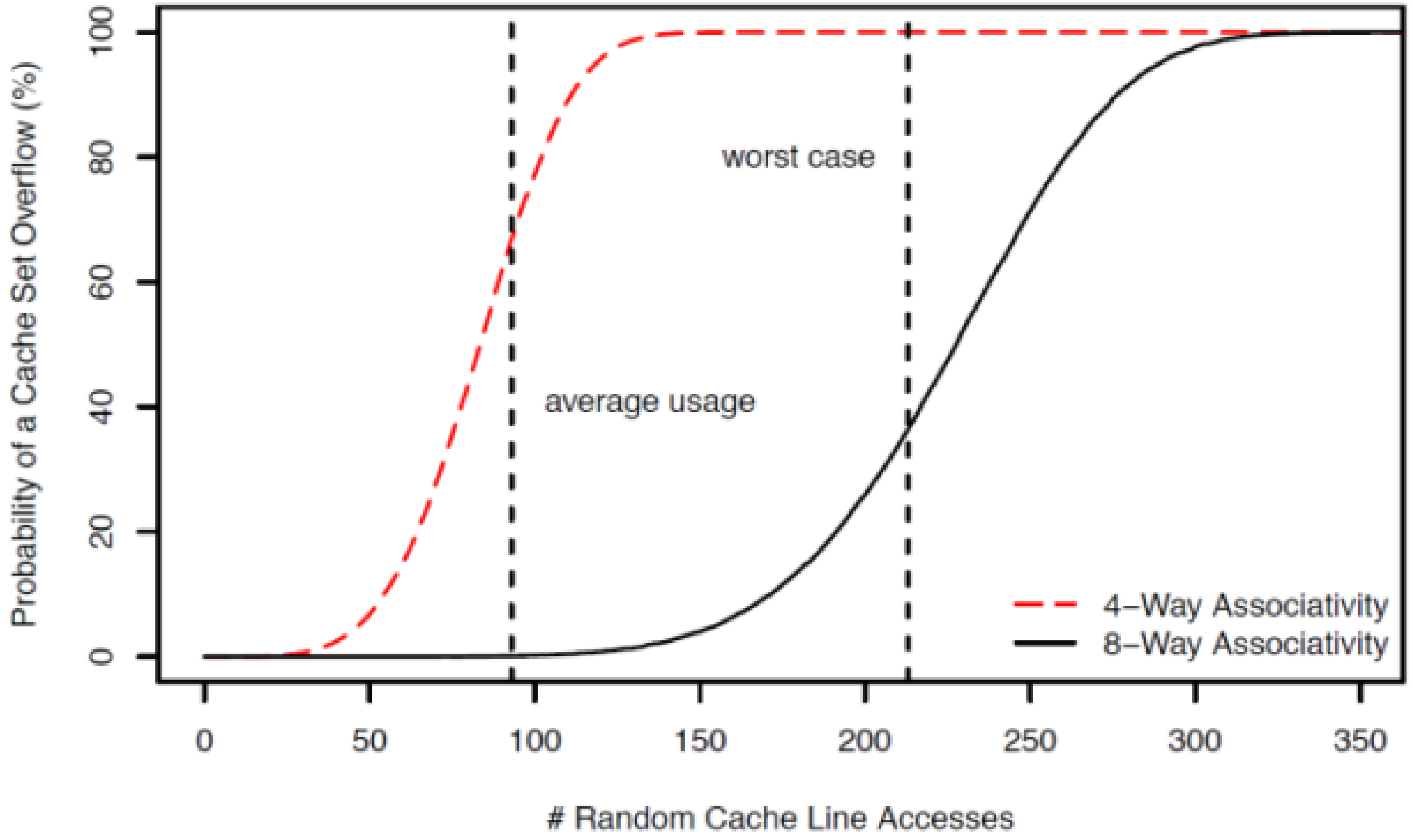
Analysis: TSX Aborts in Delta Storage Index

Capacity Aborts

- Algorithmic level
  - Node/Leaf Search Scan
  - Causes random lookups
- Cache Associativity Limits
  - Aborts typically before cache size limits
  - Hyper-threads share the L1 cache
- Dictionary contributes to larger footprint

Data Conflicts

- Single dictionary
- Global memory allocator

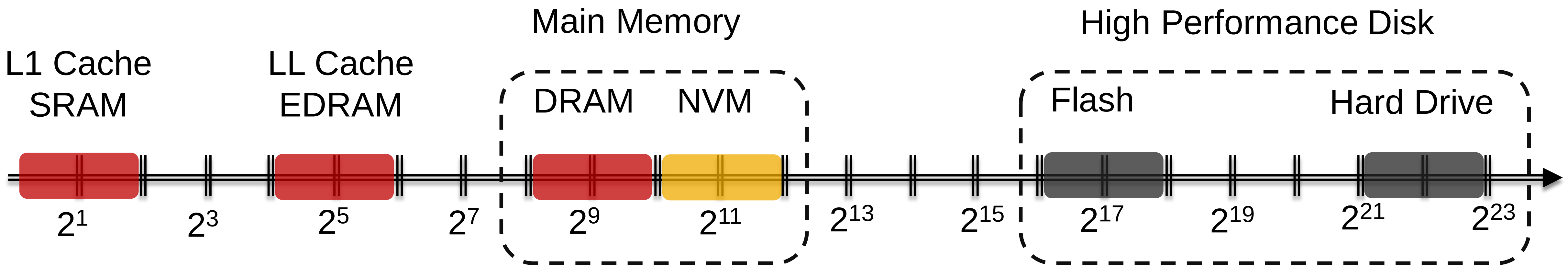


Source: Karnagel et al. *Improving In-Memory Database Index Performance with Intel® Transactional Synchronization Extensions. In* HPCA 2014.

Analysis leading to improvements in future HW generations

NVM Non-Volatile Memory

Non-Volatile Memory



Access Latency in Cycles for a 4 GHz Processor [1]

NVM writes slower than reads

Limited write endurance

NVM is a merging point between memory and storage

[1] Qureshi et al. “Phase change memory: From devices to systems”. Synthesis Lectures of Computer Science, 2011.

Non-Volatile Memory

Scale-up systems are constrained by the scalability limits of DRAM

→ NVM as potential remedy

Opportunities:

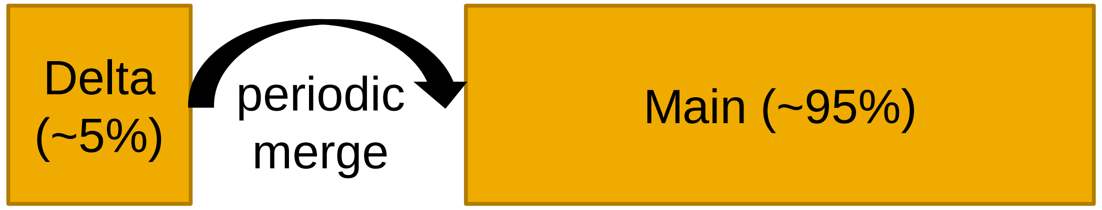
- Increased scalability
  - Larger memory modules means more memory available per server
- Significant cost savings
  - NVM will be cheaper than DRAM
- Improved recovery times

Challenges:

- Higher (than DRAM) latency impacting performance
- New technology, standards still evolving…
  - Means slow, phased implementation with increased complexity and uncertain timelines

Non-Volatile Memory Adoption in SAP HANA

- HANA is architected as a twin store
  - A large, read-only **main** part
  - A smaller, mutable **delta** part that is periodically merged into the main part



- Current NVM adoption in SAP HANA

→ **The main part is persisted in and directly accessed from NVM**

- **Pros**:

→ Faster restart time

→ Reduced TCO

→ Larger main memory

→ Write latency and endurance do not matter, only read latency does

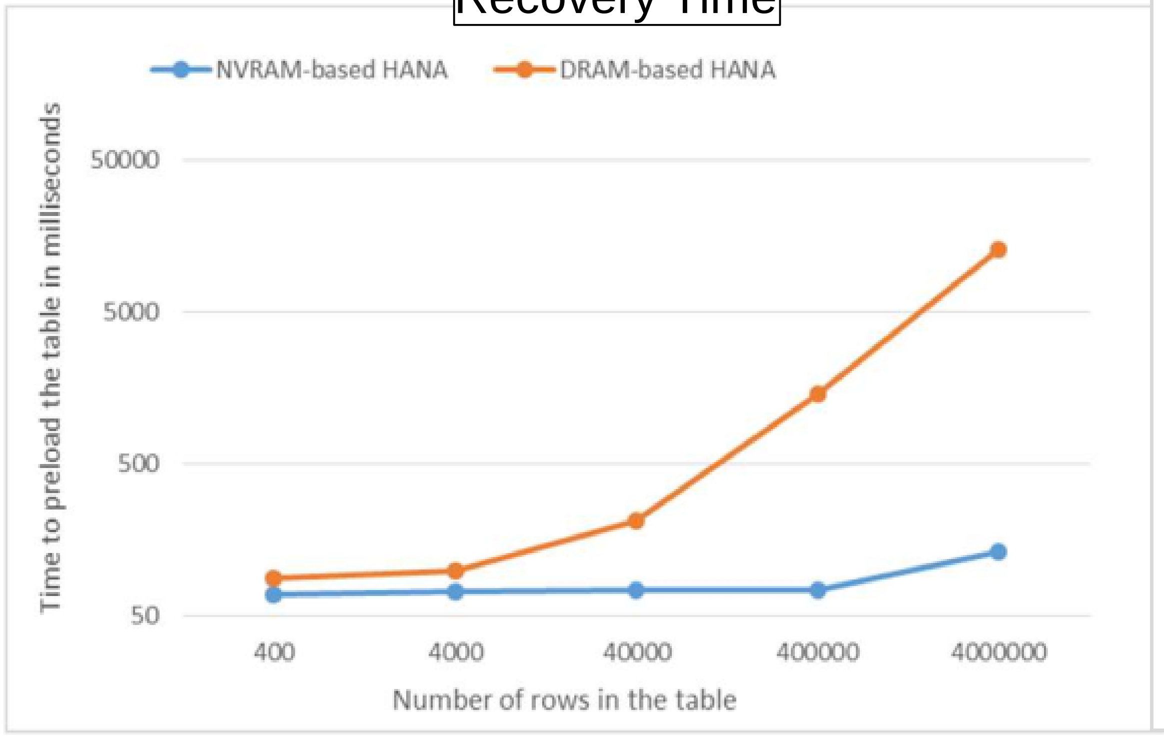
- **Cons**:

→ Performance penalty due to the higher latency of NVM

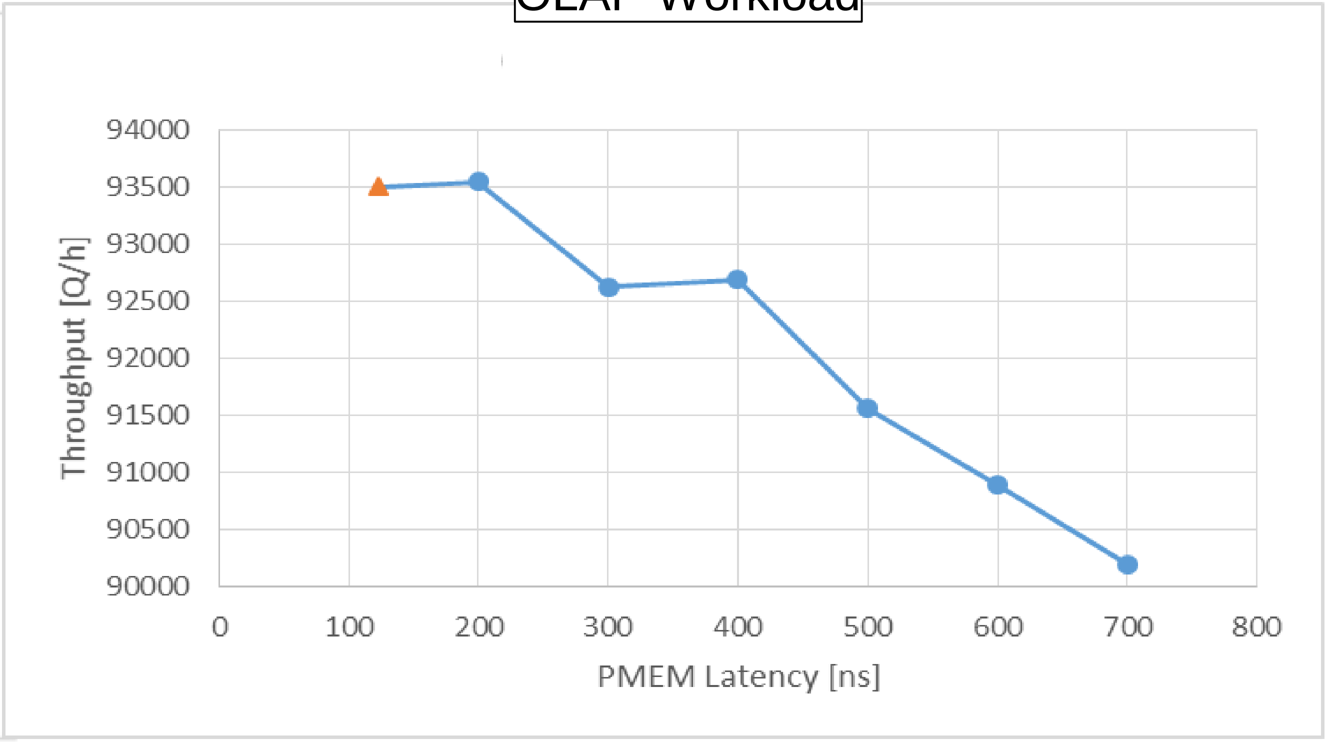
Leveraging Intel DC PM for SAP HANA: Performance analysis

Source: Mihnea et al. SAP HANA Adoption of Non-Volatile Memory. VLDB 2017.

Recovery Time



OLAP Workload



Promising results from a prototype using HW emulation:

- Significant improvements in restart time → **>100x** improvement measured
- Acceptable performance impact of higher latencies (**0-15%**)

Why is Restart Time Important?

Logging, warehousing, processing information: lifeline of companies

Information availability depends on database availability (9’s)

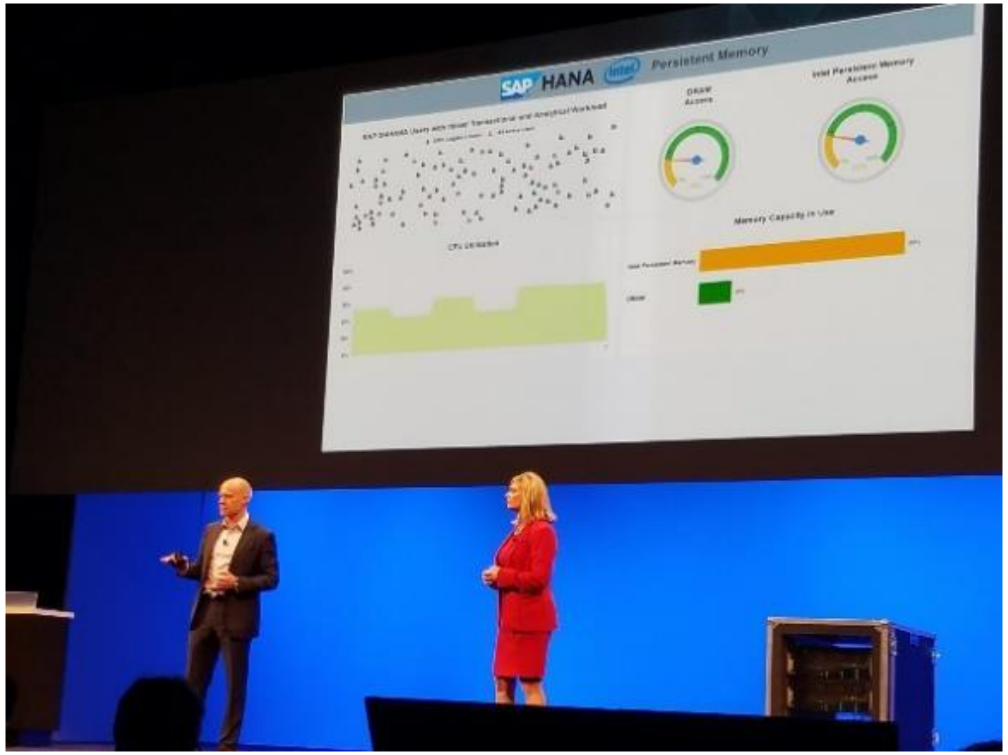
Minimize restart time to improve database availability

Availability	Annual Downtime
97%	11 days
98%	7 days
99%	3 days 15 hrs
99.9%	8 hrs 48 min
99.99%	53 min
99.999%	5 min
99.9999%	32 sec

- Each restart for an IMDB can take up to 1 hour to load TBs of data to memory.
- Dell study shows millions of dollars lost per hour due to downtime**
- Existing HA solutions increase the price exponentially for every nine

**http://tanejagroup.com/files/Compellent_TG_Opinion_5_Nines_Sept_20121.pdf

World‘s first Intel Persistent Memory Demo at Sapphire 2017

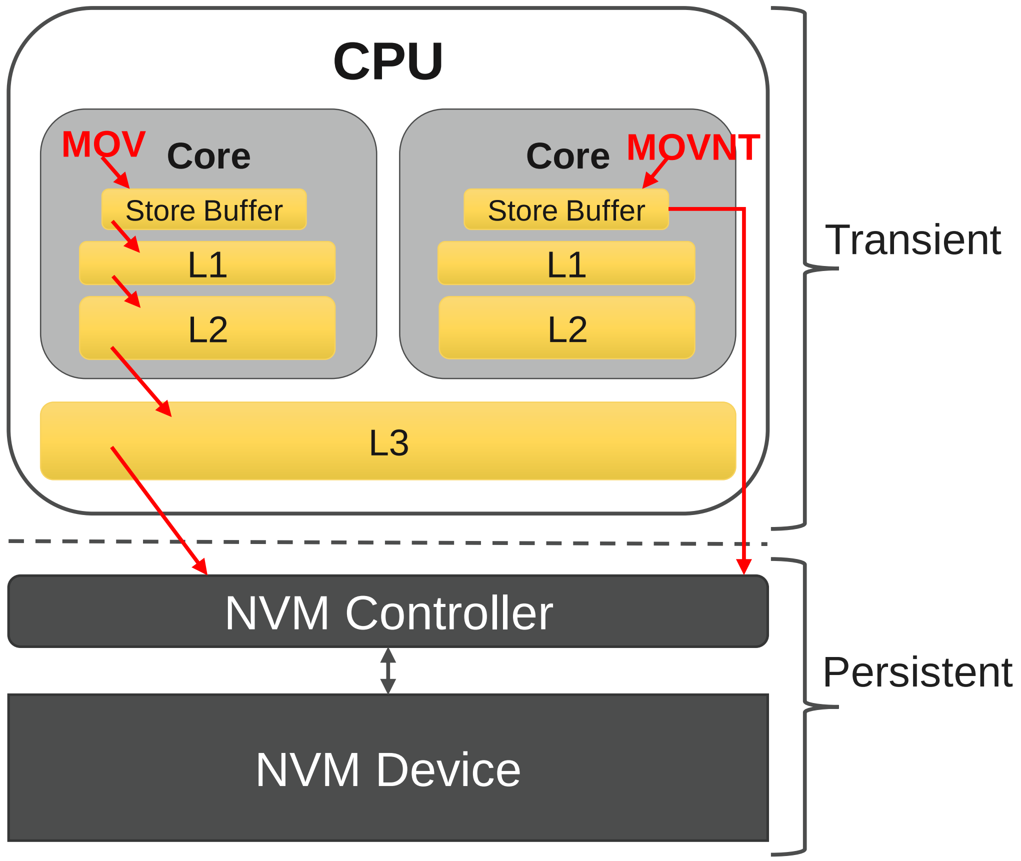


Next step: Mutable data structures on NVM

NVM Programming Challenges

- Database developers are used to:
  - Ordering operations at the logical level (e.g., write undo log, then update primary data)
  - Fully controlling when data is made persistent (e.g., log durability must precede data durability)

- NVM invalidates these assumptions:
  - Little control over when data is made persistent
  - Writes need to be ordered at the system level
  - New failure scenarios

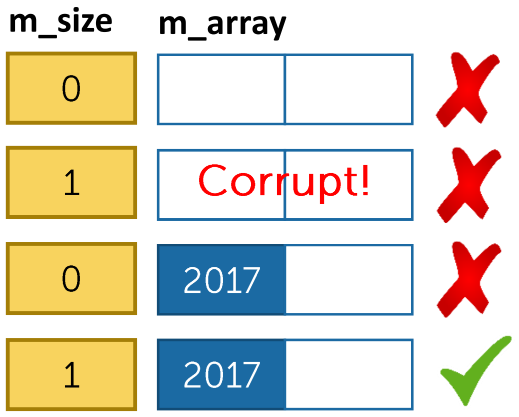


How to persist data in NVM?

Example: Array Append Operation

```
void push_back(int val){
    m_array[m_size] = val;
    sfence();
    clwb(&m_array[m_size]);
    sfence();
    m_size++;
    sfence();
    clwb(&m_size);
    sfence();
}
```

What is in NVM?



Pros:
- Low-level optimizations possible

Cons:
- Programmer must reason about the application state
  → Harder to use and error prone

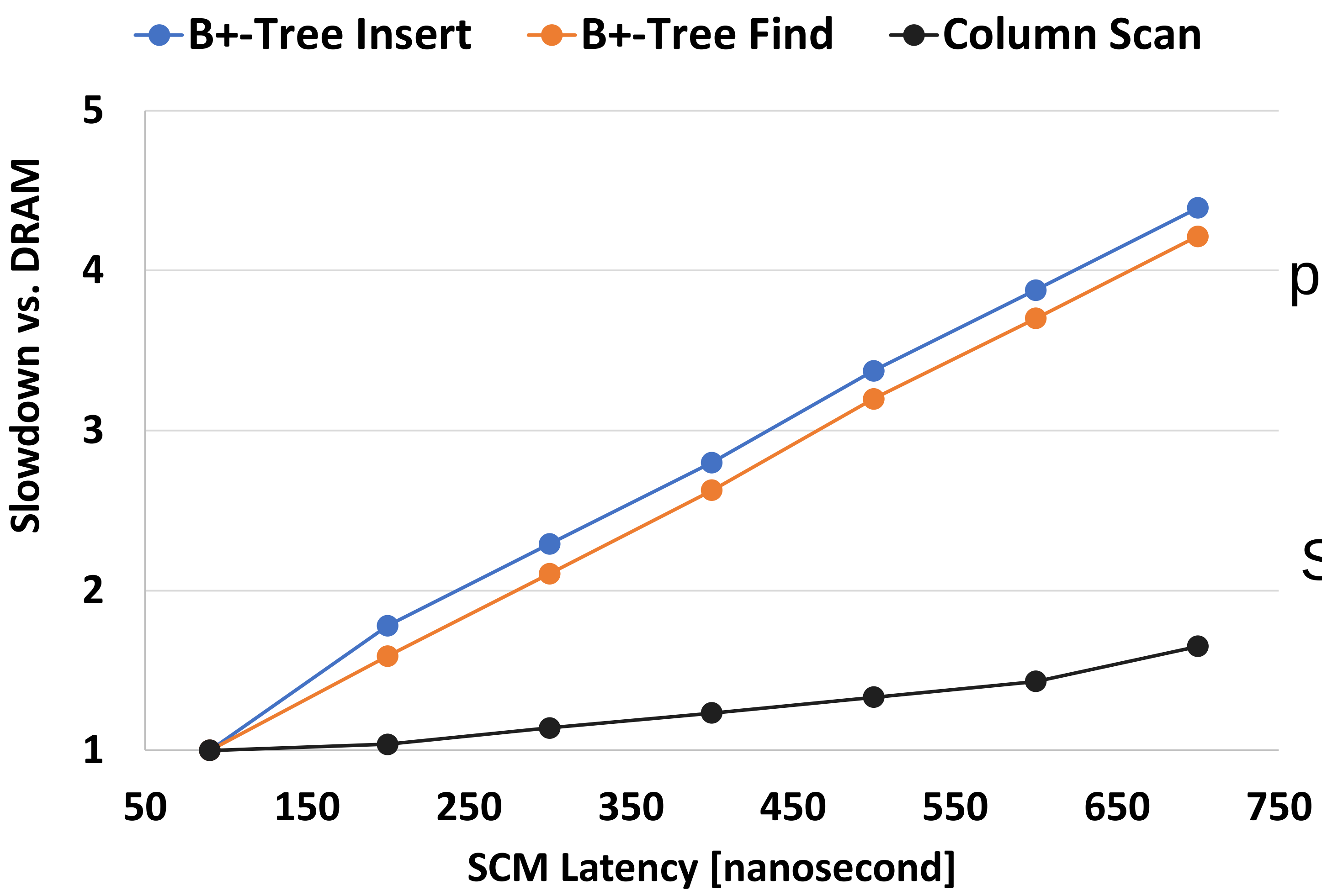
```
void push_back(int val){
    TXBEGIN {
        m_array[m_size] = val;
        m_size++;
    } TXEND
}
```

à la software transactional memory

Pros:
- Easy to use and to reason about

Cons:
- Overhead due to systematic logging
- Low-level optimizations not possible

NVM Performance Implications



Random memory access patterns significantly affected

Sequential memory access patterns hardly affected

Need to keep DRAM next to NVM

NVM-Based Data Structures: State-of-the-Art

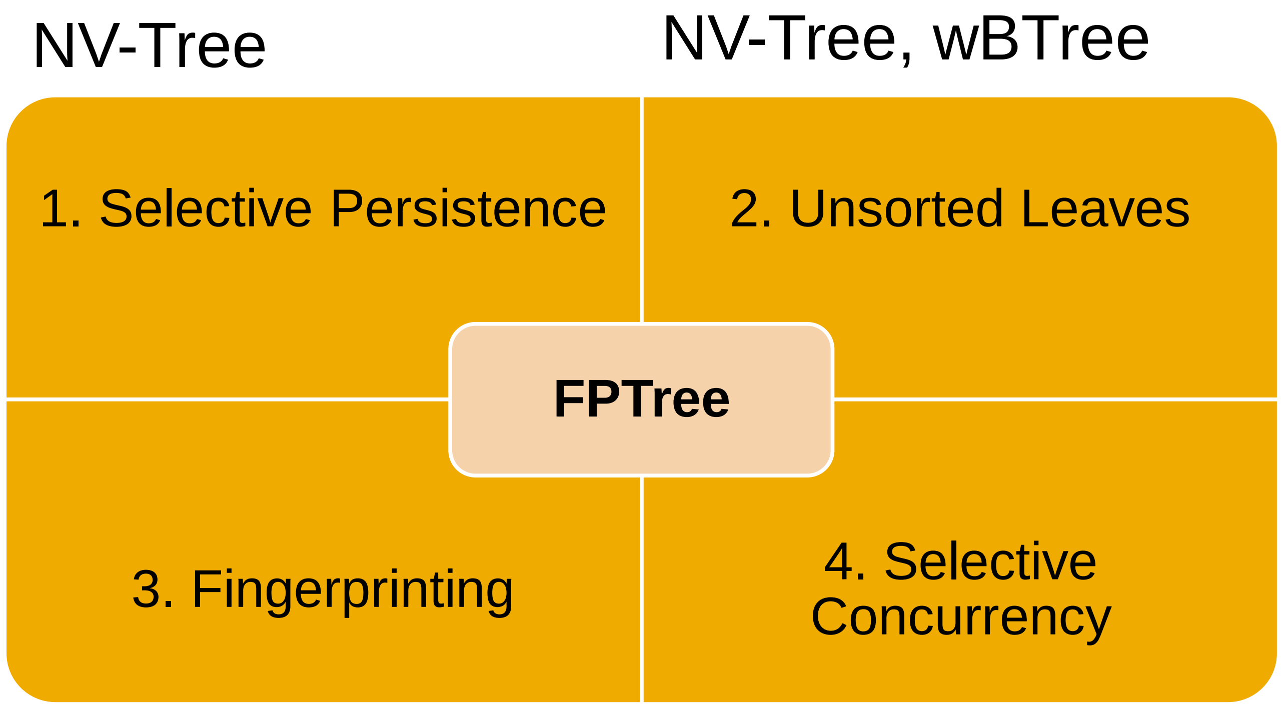
Literature focuses mostly on tree-based data structures
→ Failure-atomic updates
→ Reduce NVM writes

Literature timeline

CDDS-Tree (FAST‘11) → wB-Tree (VLDB‘15) → NV-Tree (FAST‘15) → **FPTree (SIGMOD‘16)** → WORT (FAST‘17) → HiKV (ATC‘17) → BzTree (VLDB‘18)

FPTree Design Goals

- Persistence
- Fast Recovery
- Near DRAM-Performance
- High scalability

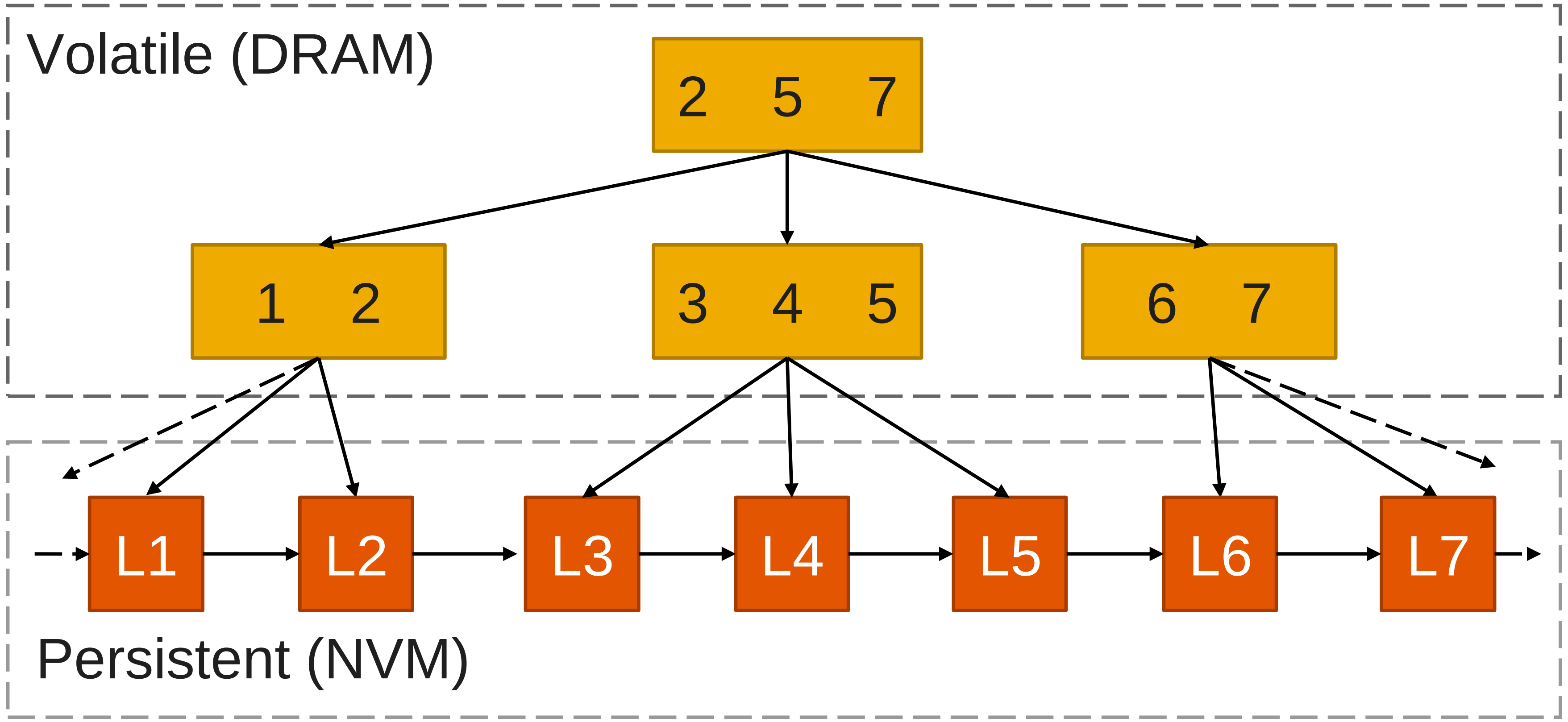


Source: Oukid et al. *FPTree: A Hybrid SCM-DRAM Persistent and Concurrent B-Tree for Storage-Class Memory*. In SIGMOD 2016.

1. Selective Persistence

Inner nodes in DRAM for better performance

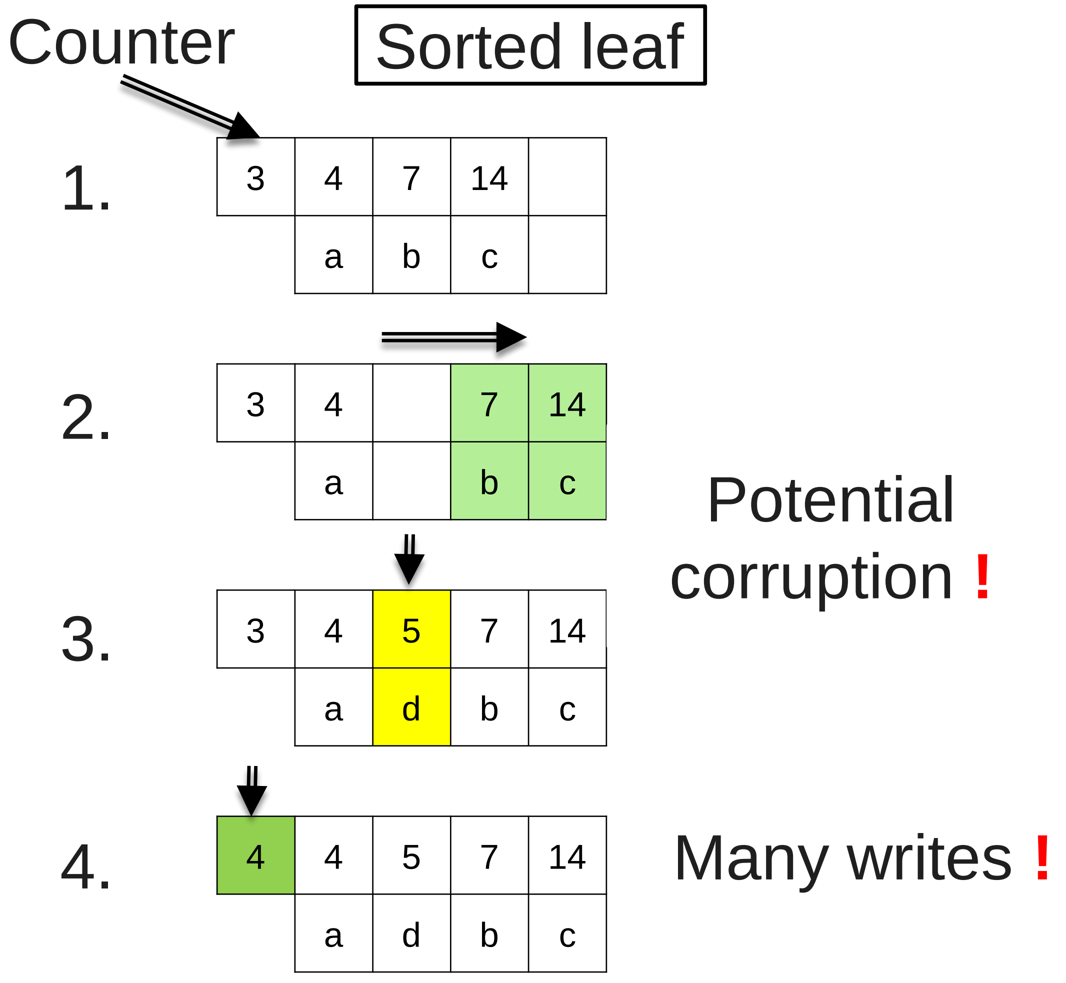
Leaves in NVM to ensure durability



Inner nodes rebuilt from leaves upon recovery in O(#entries)

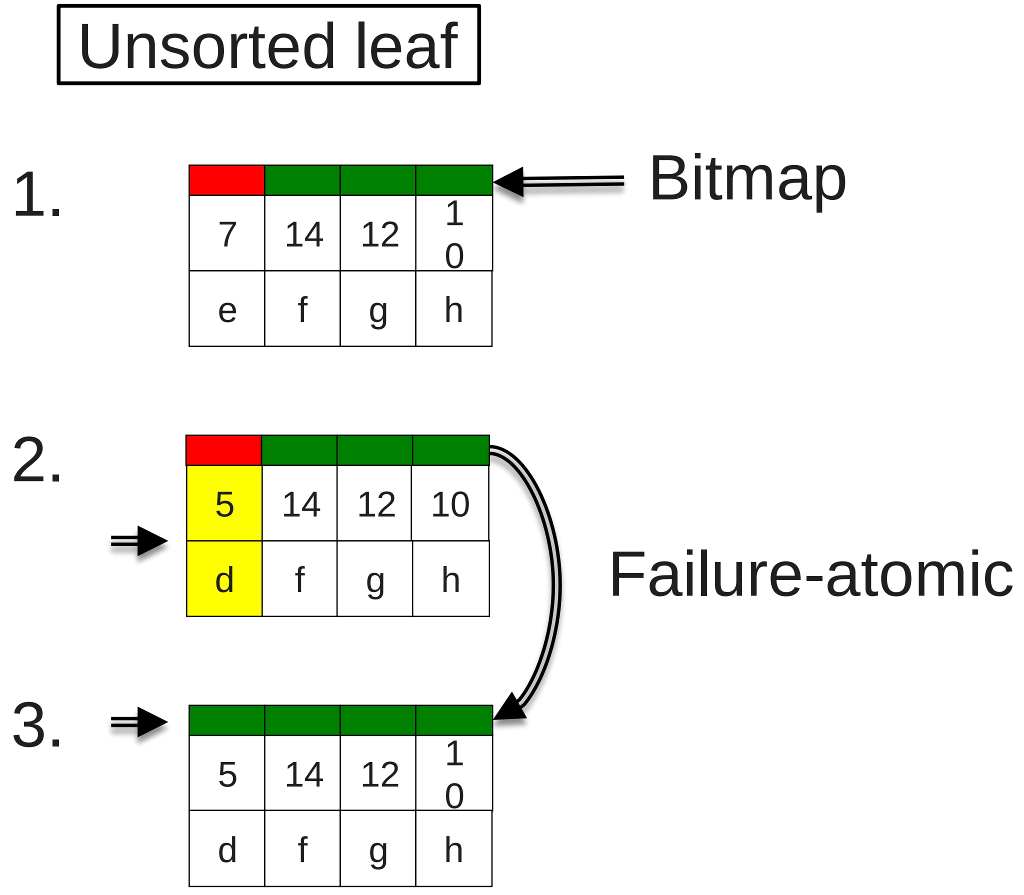
Recovery is up to 100x faster than a full rebuild

2. Unsorted Leaves



Potential corruption !

Many writes !

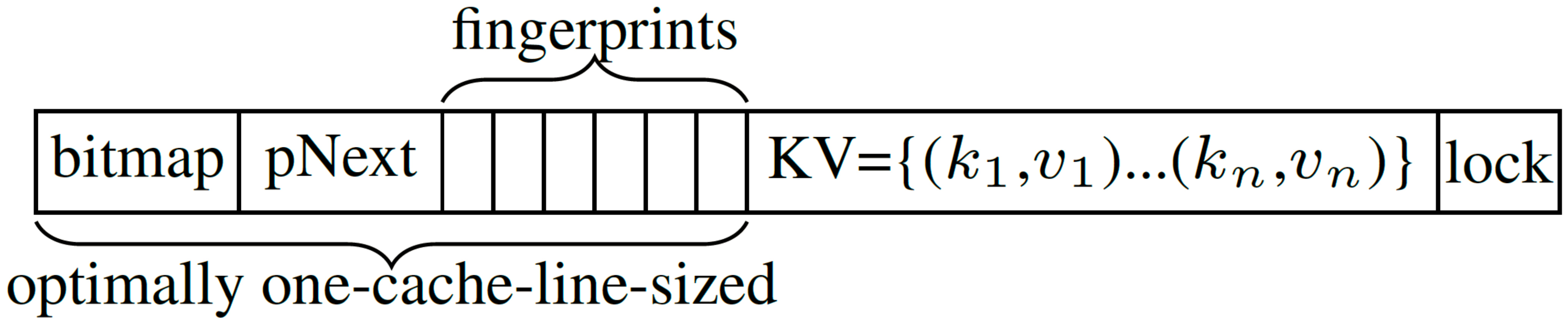


Failure-atomicity + fewer writes

But…linear search instead of binary search

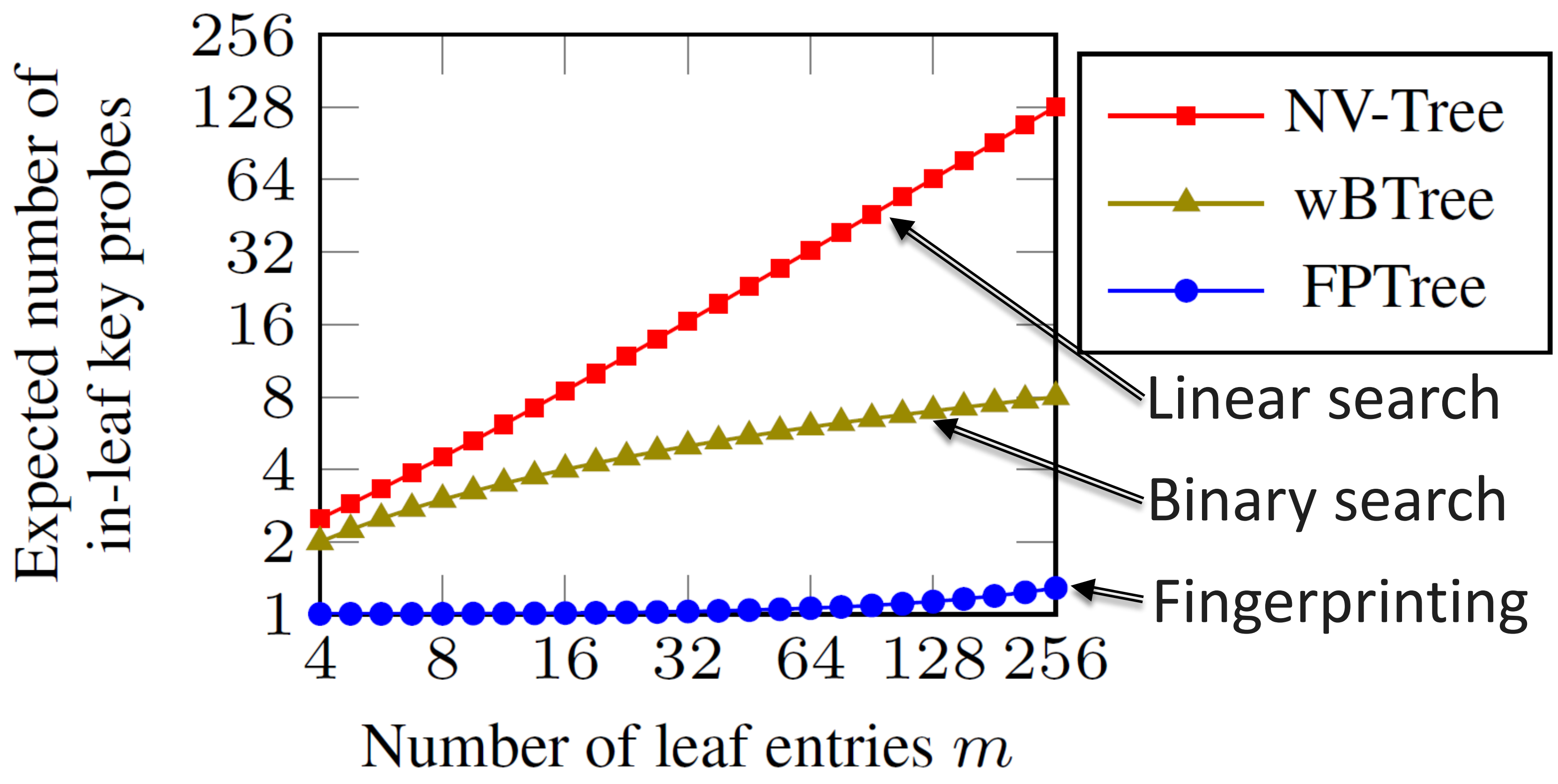
3. Fingerprinting

A fingerprint is a 1-byte hash of a key



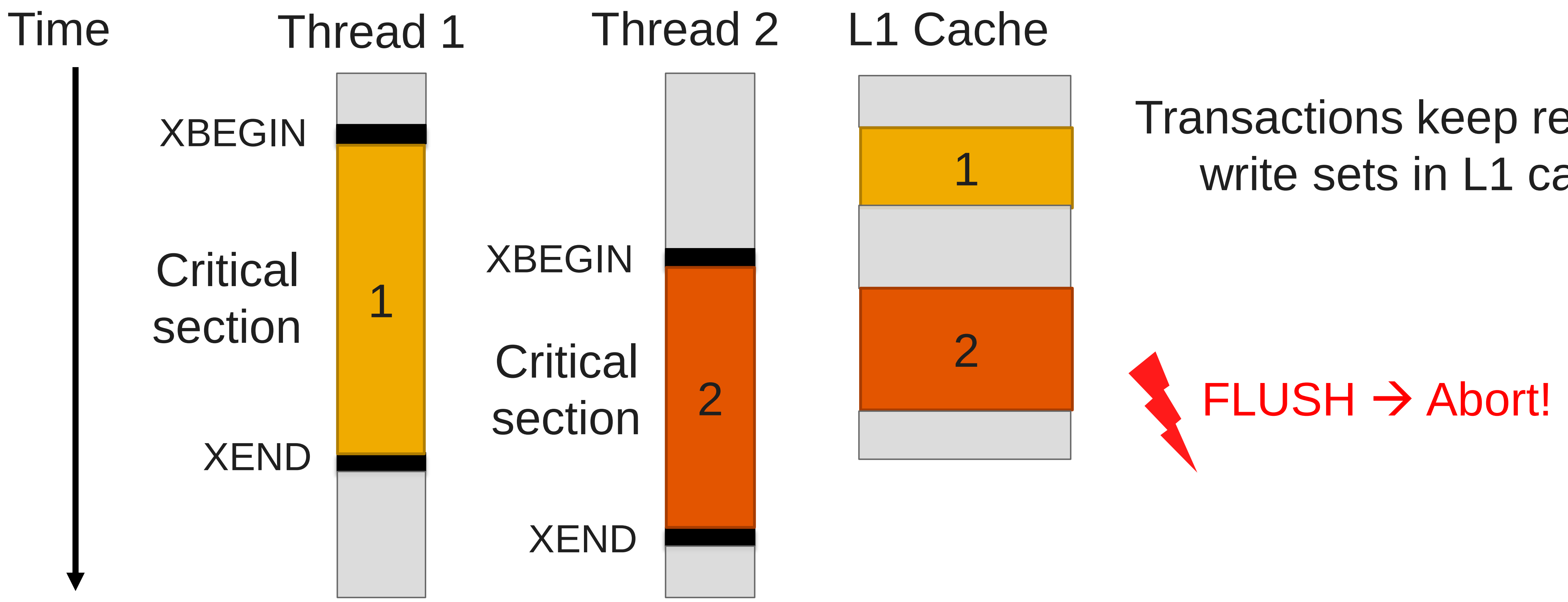
Fingerprints act as a filter to limit the number of key probes

3. Fingerprinting



Expected number of probed keys is one for leaf sizes up to 64

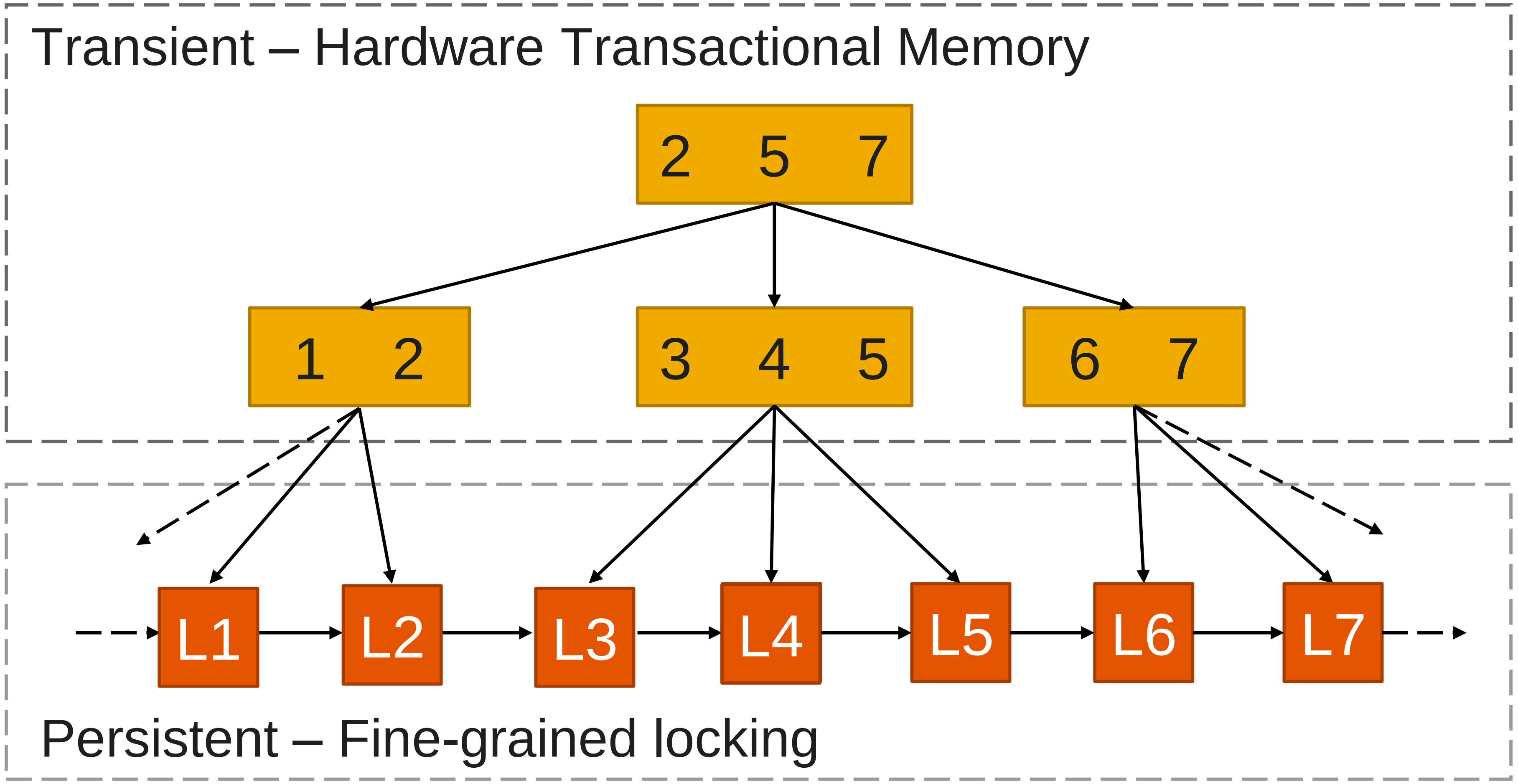
Hardware Transactional Memory (HTM)



Transactions keep read and write sets in L1 cache

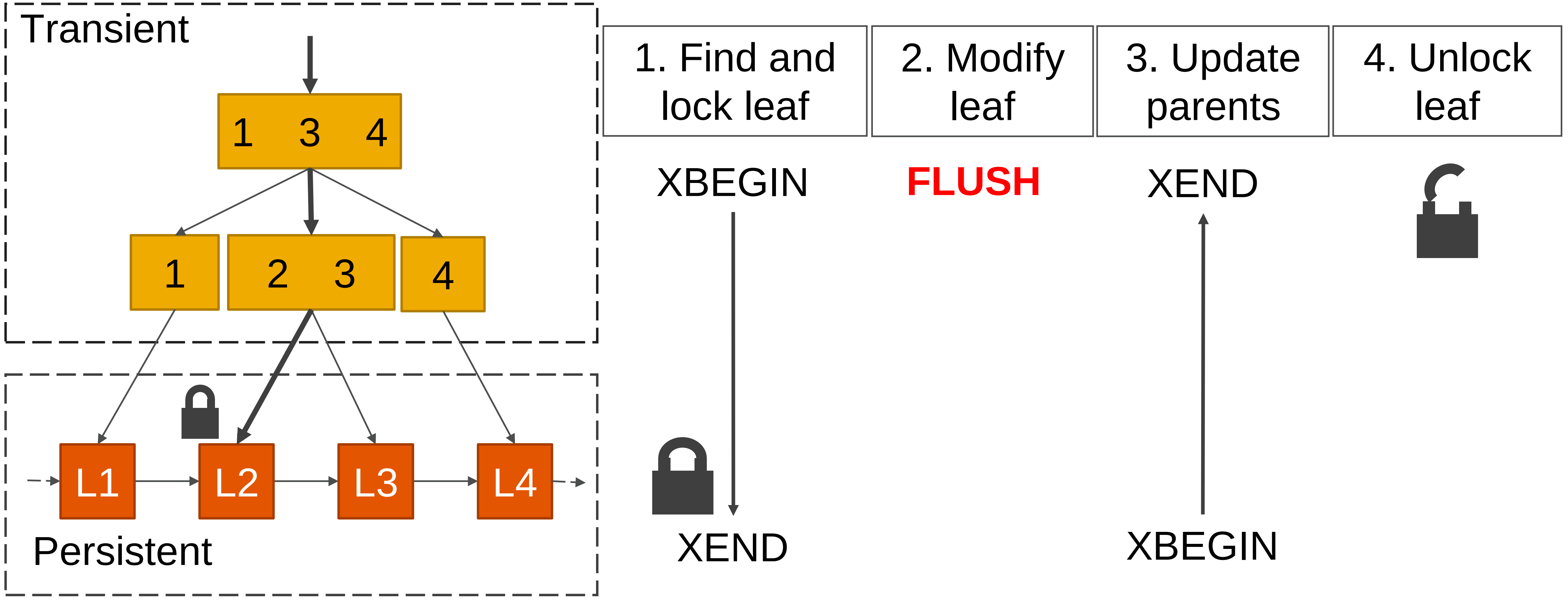
HTM and NVM are apparently incompatible

4. Selective Concurrency



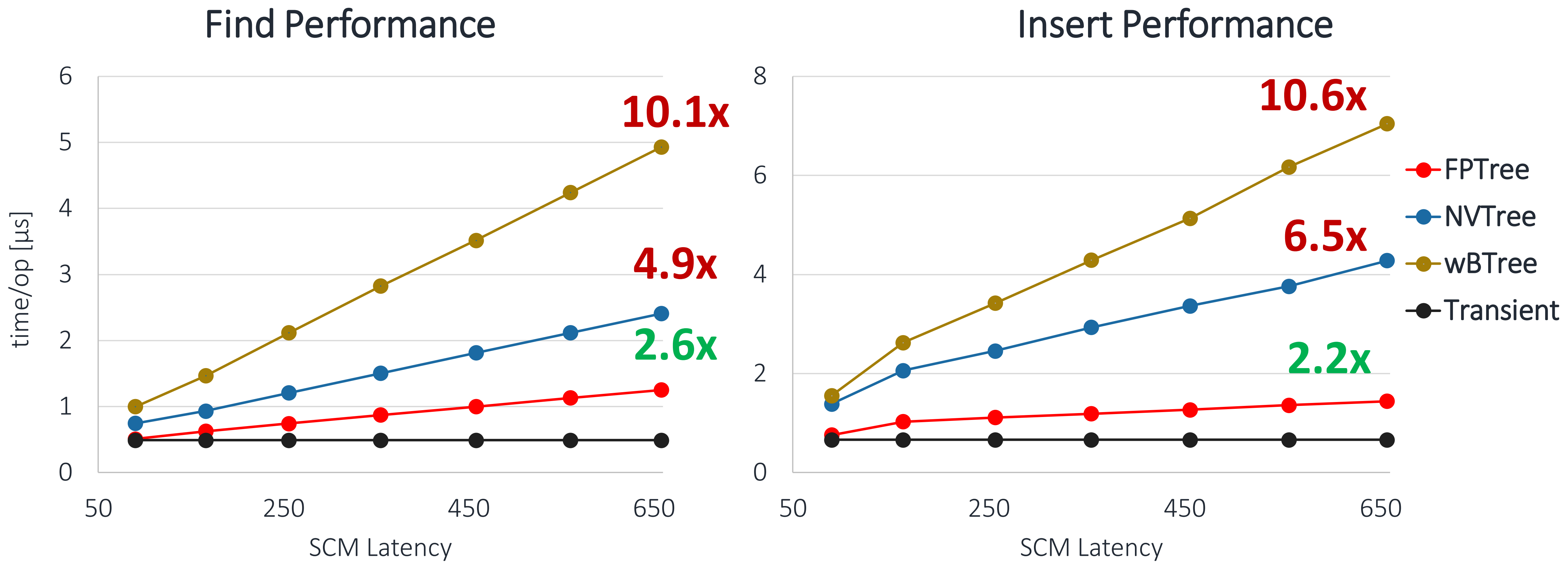
Transient – Hardware Transactional Memory
2 5 7
1 2
3 4 5
6 7
L1
L2
L3
L4
L5
L6
L7
Persistent – Fine-grained locking

4. Selective Concurrency: Insertion



Selective Concurrency solves the incompatibility of HTM and NVM

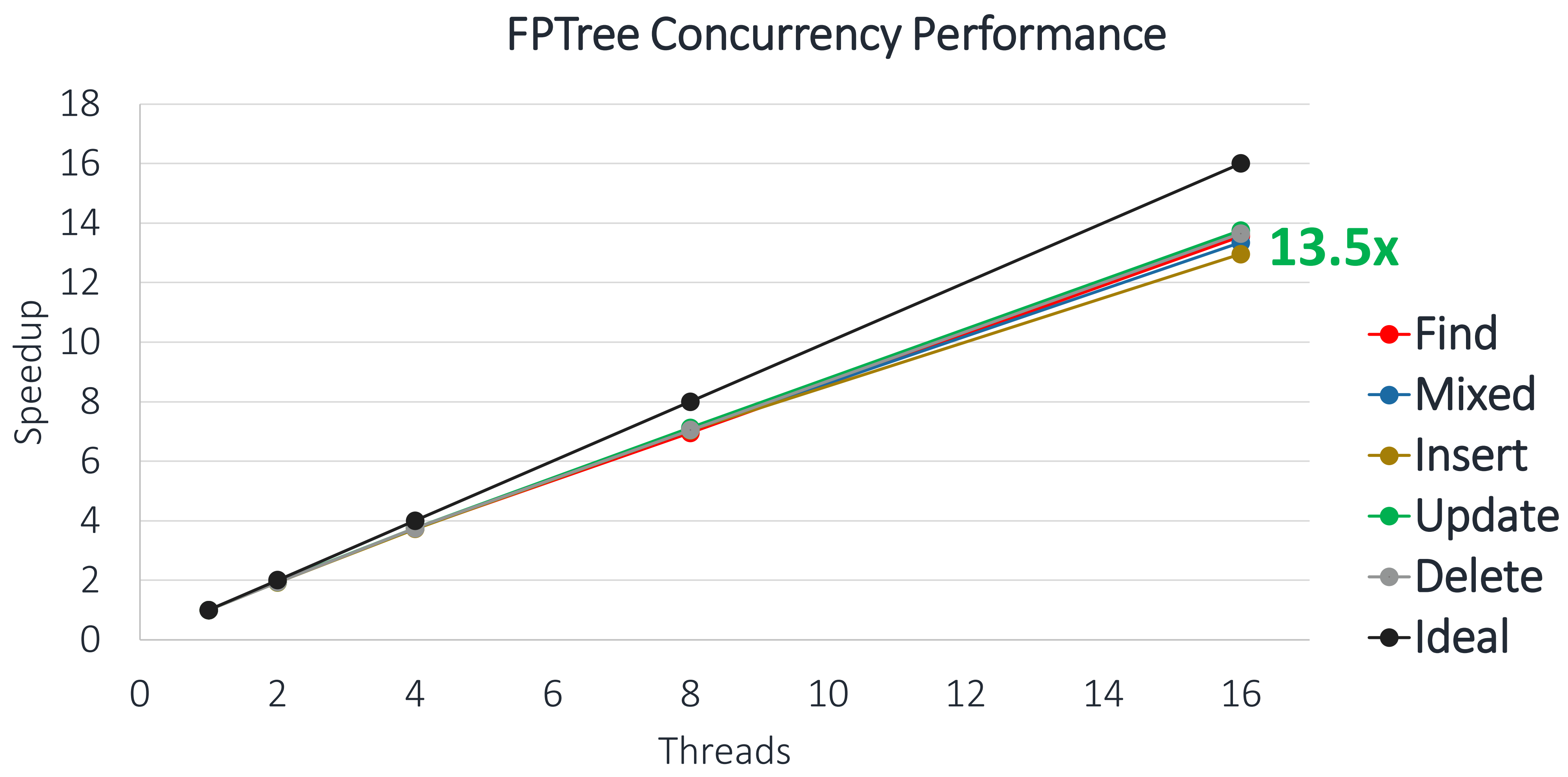
FPTree Performance Evaluation: Single-Threaded



The FPTree is competitive with a Transient B+-Tree

Only ~3% of data in DRAM

FPTree Performance Evaluation: Multi-Threaded



The FPTree scales nearly linearly

GPU Graphic Processing Unit

GPUs: The New Supercomputers

Slide by: Dr. Norman May

	Nvidia V100 (2017)	IBM ASCI White (2000)
Number of Processor Cores	3584	8192 (512 nodes x 16 IBM Power3)
Double-Precision Performance	7.5 TeraFLOPS	7.2 TeraFLOPS
NVIDIA NVLink™ v2 Interconnect Bandwidth	2x150 GB/s	N/A
PCIe x16 Interconnect Bandwidth	2x16 GB/s	N/A
Memory Capacity	16 GB	**6 TB DRAM** (Power 3 with up to **16 MB L2 cache**)
Max. overall data transfer speed	900 GB/s	?
Weight	450 gramm	106 tons
Energy consumption	**300W**	**3 MW**

https://www.nvidia.com/en-us/data-center/tesla-v100/

https://www.top500.org/featured/systems/asci-white-lawrence-livermore-national-laboratory/

GPUs and DBMSs: A Story of Negative Research Results

Slide by: Dr. Norman May

	Why past attempts failed? (2007)	What has changed? (2018)
Memory	• < 1GB of memory too small for DBMS workload • Allocation: host vs. device memory	• 16 GB of memory maybe enough for hot tables • Memory allocation unchanged
Bandwidth CPU ⇔ GPU	• Good Bandwidth to GDRAM, but • PCIe v2 with 2x0.5 GB/s per lane as bottleneck	• PCIe v4 with 2x1.9 GB/s per lane still too slow • Nvidia NVLink v2 with up to 2x150 GB/s!
Programming Model	• Hard to debug and profile	• Improved tool support, e.g. debugging and profiling • Matured libraries

Need to revisit negative GPU research results

Query Processing on GPUs

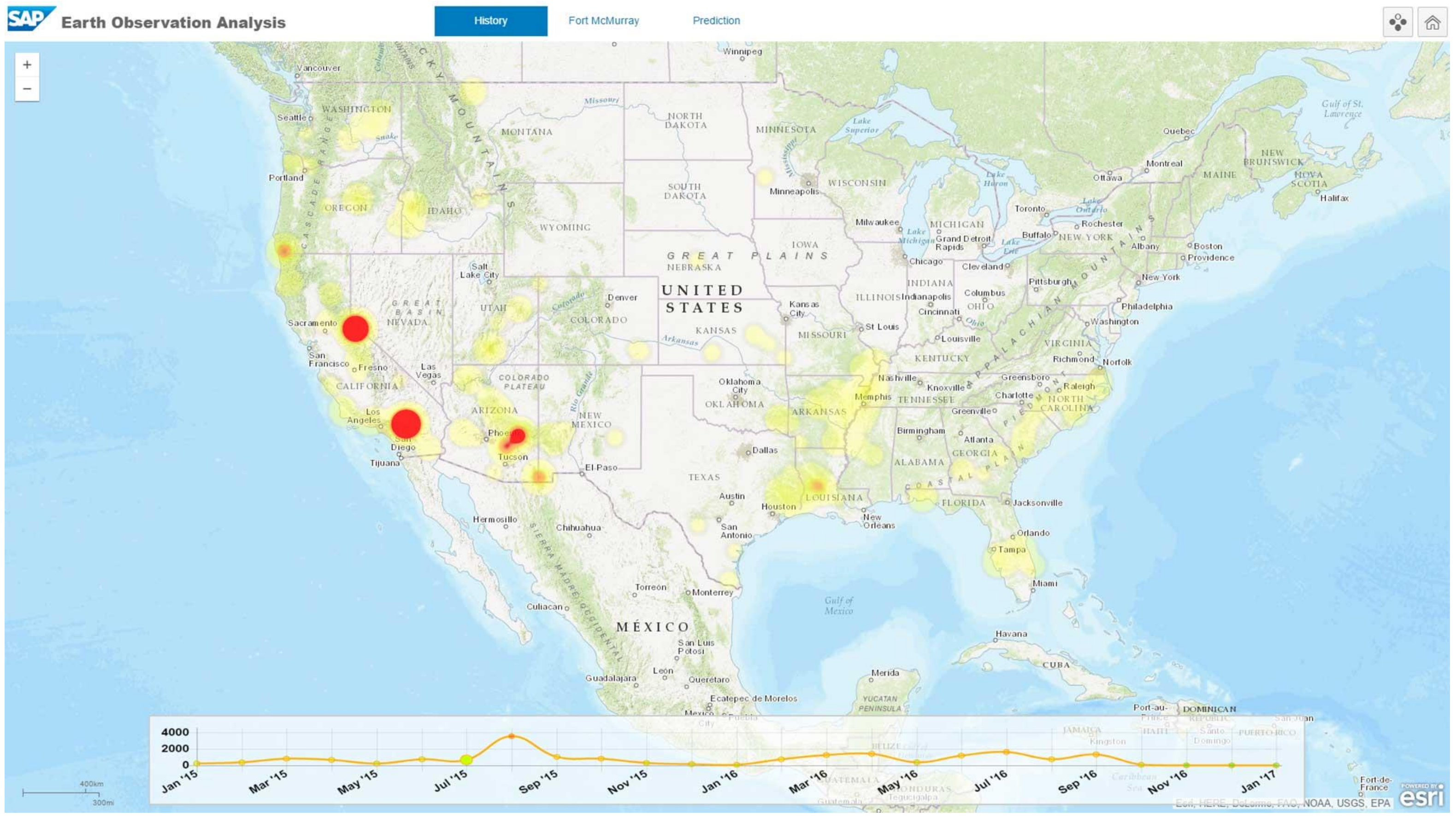
Slide by: Dr. Norman May

Query Engine

- GPU memory still tiny
- Research not promising:
  - Significant programming effort
  - Code duplication for relatively small gain
  - OLTP on the GPU?
  - Most DBMS code not GPU-friendly
- But promising examples in streaming and analytic scenarios, e.g. MapD, Kinetica.

Geospatial Processing

- Classical expensive UDFs, often natural mapping to GPU operations
- Geo-Clustering
- Typically Graphical User Interfaces



https://devportal.yaas.io/services/earthobservationanalysis/latest/

Beyond Query Processing

Slide by: Dr. Norman May

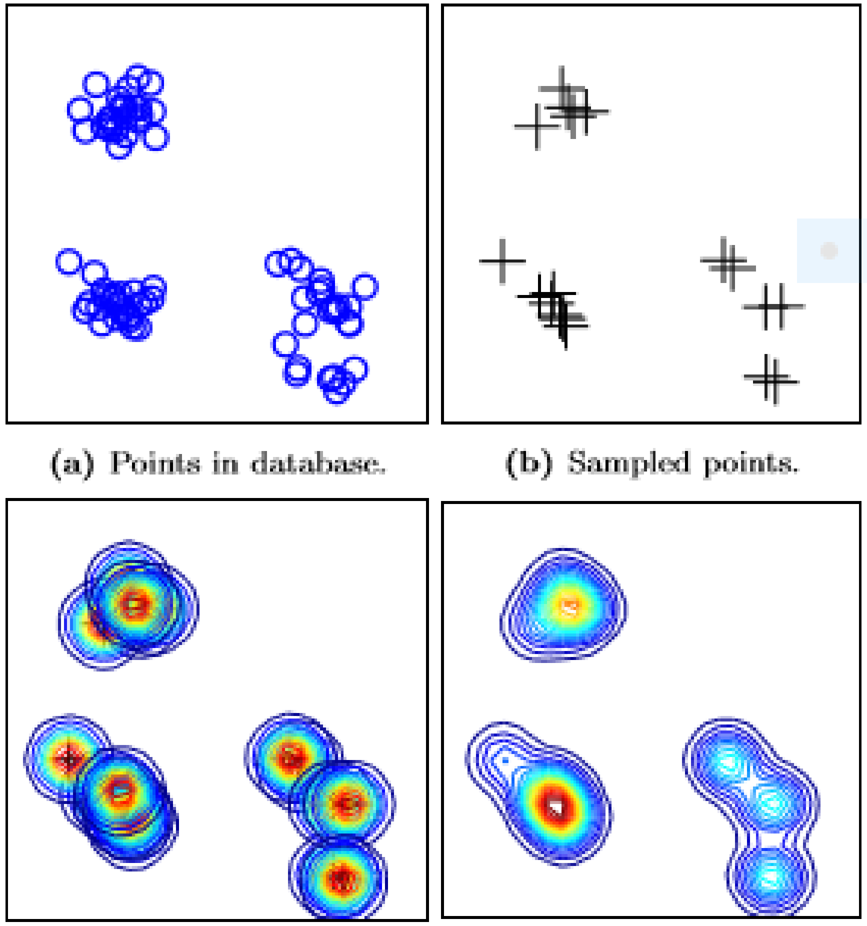
Cardinality Estimation

- Small data that rarely change, e.g. samples, sketches, histograms
- Asynchronous processing often acceptable
- Some methods are very compute intensive and GPU-friendly, e.g. Maximum Entropy, Kernel Density Models

Machine Learning Extension in the DBMS Kernel

- HANA has Machine Learning / Data Mining capabilities built in
  – Predictive Analytics Library (Clustering, etc.)
  – Tensor-Flow Integration

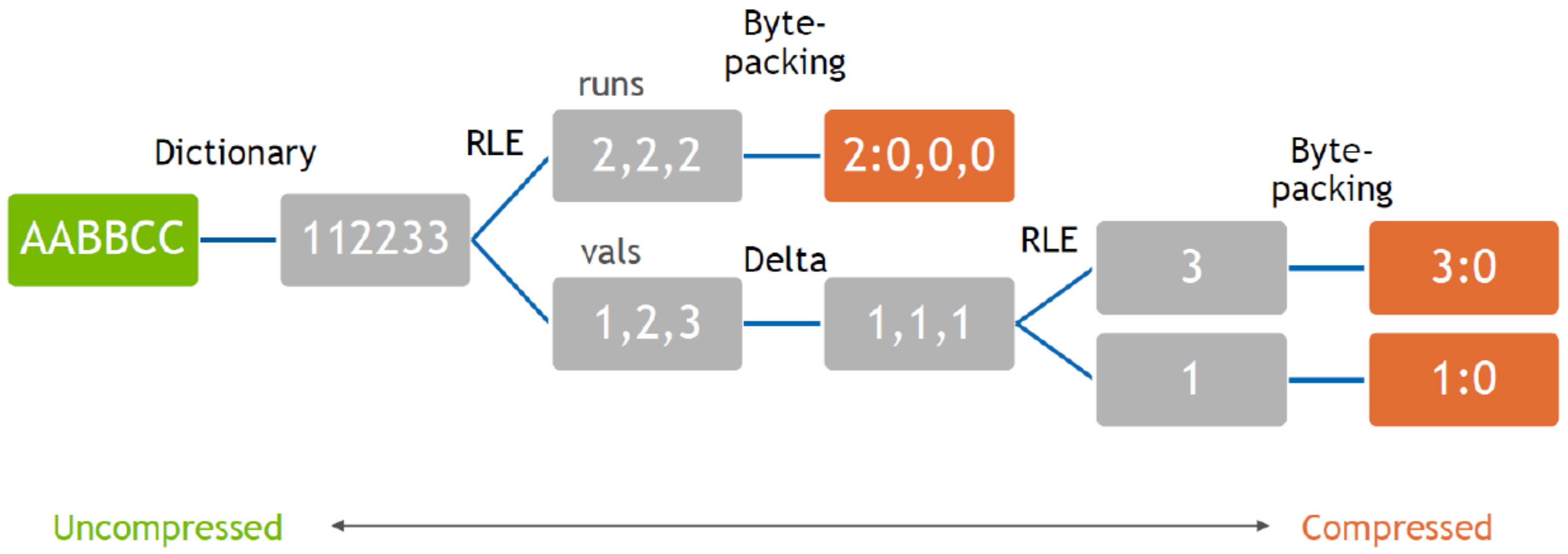
GPU application domain is expanding



Source: http://dl.acm.org/citation.cfm?id=2749438

Pipelined Compression

Source: Felipe Aramburu (**Blazing DB**) & Nikolay Sakharnykh (**Nvidia**). *Breaking the Speed of Interconnect with Compression for Database Applications.* http://on-demand-gtc.gputechconf.com/gtc-quicklink/2VMqF



TPC-H *l_orderkey* compression rates with NVCOMP:

- RLE + Byte-packing = **3.2x**
- RLE + Delta + RLE + Byte-packing = **10.6x → 2x** faster data load
- TPC-H Q4 and Q21 **~10x** and **~6x** faster on GPU than on CPU (details in presentation video)

FPGA Field-Programmable Gate Array

Broad Architecture Possibilities

FPGA as Coprocessor

- On-chip
- Connected via PCIe
- Connected via UPI
- Connected via Network

Networking

- FPGA on NIC
- NIC on FPGA
- FPGA on Network Switch

Storage

- FPGA on SSD
  - e.g., Samsung's KV SSD: https://www.samsung.com/semiconductor/global.semi.static/Samsung_Key_Value_SSD_enables_High_Performance_Scaling-0.pdf
- FPGA as storage node*
- FPGA as memory node*

*FPGAs may have ARM cores

Major Roadblock: Programmability

Our Pathfinding Efforts so Far Using OpenCL

Three ways an FPGA as coprocessor can be beneficial

- Accelerate an existing workload
- Offload operations to FPGA to relieve CPU resources without loss in performance
- Given a time budget, do something better than the CPU, e.g. better compression ratio

String Predicate Evaluation on FPGA

- Create a configurable select engine that is faster than a plain CPU implementation
- 80% of SAP HANA columns contain strings
  → Focus on string operators e.g. SQL LIKE

String Compression on FPGA

- HANA uses extensively compression
- Goal: either accelerate or improve the compression ratio without loss in speed

FPGAs have huge potential, but compelling use cases are hard to find

The Intel / Altera OpenCL Experience

The Good

- The Compiler (Quartus) gives useful warnings (although misses important ones)
- Compilation reports help identify bottlenecks
- Intel / Altera are very **responsive** to our feedback

The Bad

- Data has to be packed and streamed chunk by chunk to the FPGA
- Clear separation between host and device code
- Limited streaming capabilities (pipes not adequate)
- Limited debugging possibilities on the device (mainly printf)
- Obscure language semantics (e.g. how to organize load/store units, etc.)
- **No definitive how-to guide**. Knowledge only acquired through (a lot of) experience.

Hardware Limitations

- Host must actively transport data to device, thereby consuming CPU resources
- Not possible to create more than one pipe per direction

Desired Properties

- Unified host-device virtual memory
  - Direct, zero-copy access to host memory from the FPGA with NUMA-like latencies
- Software-tunable hardware prefetching from host to FPGA
- Programming language with device code similar to host code (e.g. CUDA, SYCL?)
- Tool supportability, industry-wide standardization, compiler maturity

SDMC Software-Defined Memory Coherency

Motivation

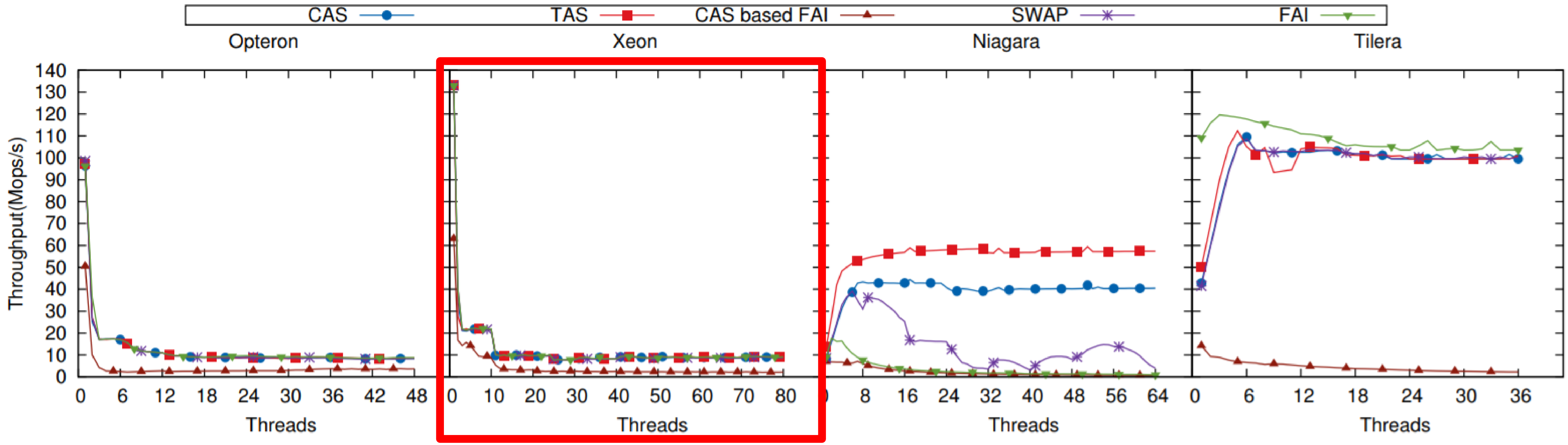


Figure 4: Throughput of different atomic operations on a single memory location.

Source: Tudor et al. *Everything you always wanted to know about synchronization but were afraid to ask*. SOSP'13

Atomic Instructions Do Not Scale

Why is CC so Expensive?

HW always tracks coherency state

- Not always necessary, e.g. read-only

HW enforces coherency on cache line basis

- HW efficiently handles CC for cache lines, but size of “piece” of data varies

Example: HPE UV300

- Cache-coherent domain is huge (4-32 sockets and up to 48 TB DRAM)
- Increased memory latencies due to snooping
- Atomic instructions prohibitively expensive at the slightest contention

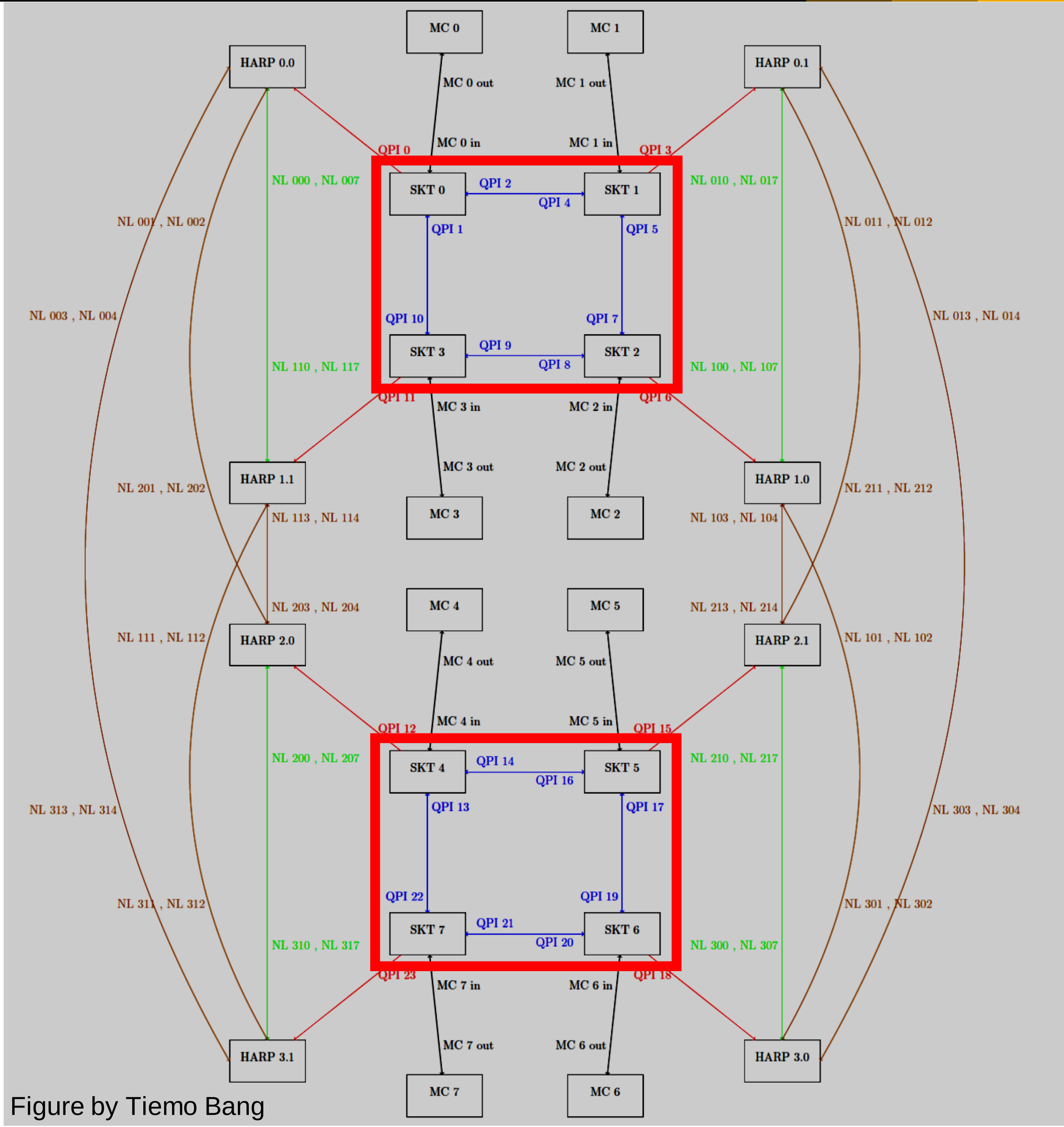
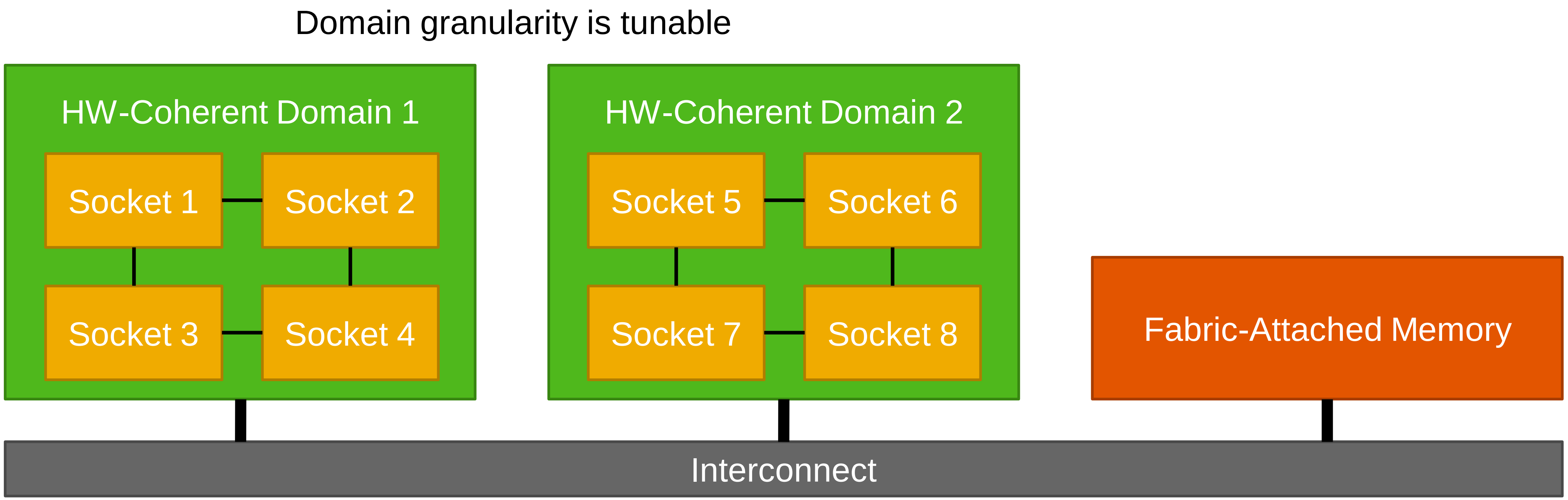


Figure by Tiemo Bang

Key Idea: Split the System Into Smaller Cache-Coherent Domains



- HW manages coherency within a domain, SW manages coherency across domains
- No or partial hardware coherency across domains → expected latency improvement

Scale up system with scale out semantics

Prototype based on HPE Superdome Flex*

HPE announced a prototype for Software-Defined Scalable Memory based on Superflex Dome

A step towards “real” memory-centric computing with fabric-attached memory (FAM)

*https://h20195.www2.hpe.com/v2/gethtml.aspx?docname=a00026242enw

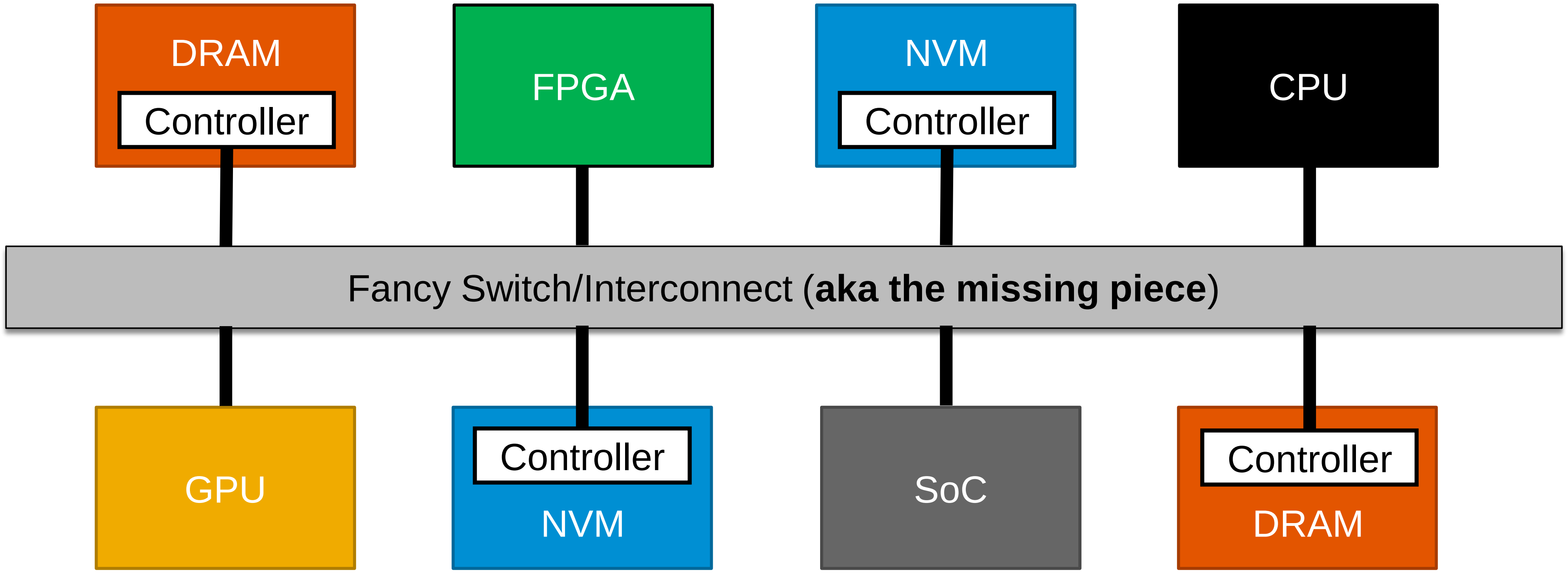
HPE BOOTS UP SANDBOX OF THE MACHINE FOR EARLY USERS

June 21, 2018 Jeffrey Burt

https://www.nextplatform.com/2018/06/21/hpe-boots-up-sandbox-of-the-machine-for-early-users/

Gen-Z – Towards Software-Composable Hardware

- Pool of media modules (DRAM, NVM, GPU, FPGA, etc.) and SoCs
- Remote resources exposed/accessible as local resources



Conclusion

- **SIMD and HTM**
  - → Well established technologies with proven benefits. Still evolving, thereby unlocking novel use cases.

- **Non-Volatile Memory**
  - → Programming model and toolchain available (PMDK)
  - → Hardware availability is imminent!

- **GPUs**
  - → Need to revisit negative results in light of new hardware advancements
  - → GPU application domain is expanding

- **FPGAs**
  - → Programmability is a major issue, OpenCL toolchain not mature
  - → The potential is huge: Smart NICs & Storage, encryption, compression, QP, storage/memory node, etc.

- **Software-Defined Memory Coherency**
  - → A merging point between scale-up and scale-out systems
  - → The first step towards software-composable hardware

Thank you.

Contact information:

Dr.-Ing. Ismail Oukid
Ismail.Oukid@sap.com

We are hiring!

- **Full-time positions**
- **Student jobs (master/bachelor theses, internships, working student)**

Email us at: students-hana@sap.com

Run Simple

* Benchmark Details

SAP BW Enhanced Mixed Load (BW-EML) Standard Application Benchmark with a total of 1,000,000,000 records. Dell PowerEdge R930, 4 processors / 72 cores / 144 threads, Intel Xeon Processor E7-8890 v3, 2.50 GHz, 64 KB L1 cache and 256 KB L2 cache per core, 45 MB L3 cache per processor, 1536 GB main memory . Certification #: 2015015
http://global.sap.com/solutions/benchmark/pdf/Cert15015.pdf

SAP BW Enhanced Mixed Load (BW-EML) Standard Application Benchmark with a total of 1,000,000,000 records. HP DL580 G7, 4 processor / 40 cores / 80 threads, Intel Xeon Processor E7-4870, 2.40 GHz, 64 KB L1 cache and 256 KB L2 cache per core, 30 MB L3 cache per processor, 512 GB main memory. Certification #: 2013027
http://global.sap.com/solutions/benchmark/pdf/Cert13027.pdf

SAP BW Advanced Mixed Load (BW-AML) Standard Application Benchmark, 2B initial records. Fujitsu PRIMERGY RX4770 M3, 4 processors / 96 cores / 192 threads, Intel Xeon Processor E7-8890 v4, 2.20 GHz, 64 KB L1 cache and 256 KB L2 cache per core, 60 MB L3 cache per processor, 1024 GB main memory. Certification #: 2017012
https://www.sap.com/documents/2017/06/8e11832a-c27c-0010-82c7-eda71af511fa.html

SAP BW Advanced Mixed Load (BW-AML) Standard Application Benchmark, 2B initial records. Fujitsu PRIMERGY RX4770 M2, 4 processors / 72 cores / 144 threads, Intel Xeon Processor E7-8890 v3, 2.50 GHz, 64 KB L1 cache and 256 KB L2 cache per core, 45 MB L3 cache per processor, 1536 GB main memory. Certification #: 2016049
http://global.sap.com/solutions/benchmark/pdf/Cert16040.pdf

SAP* BW edition for SAP HANA* Standard Application Benchmark* @ 1.3 billion (1.3B) initial records result published at http://global.sap.com/solutions/benchmark as of 11 July 2017 Huawei FusionServer RH5885H V3, 4 processor / 96 cores / 192 threads, Intel Xeon Processor E7-8890 v4, 2.20 GHz, 64 KB L1 cache and 256 KB L2 cache per core, 60 MB L3 cache per processor, 2048 GB main memory. Certification #: 2017004
https://www.sap.com/documents/2017/02/ac8d3332-a77c-0010-82c7-eda71af511fa.html

SAP* BW edition for SAP HANA* Standard Application Benchmark* @ 1.3 billion (1.3B) initial records result published at http://global.sap.com/solutions/benchmark as of 11 July 2017. 4x Intel® Xeon® Platinum 8180 processor (112 cores/224 threads) on HPE CS500 (DL560 Gen10) with 3072 GB total memory on SUSE* Linux Enterprise Server 12 using SAP HANA 1.0, SAP NetWeaver 7.50. Benchmark: SAP BW for SAP HANA @ 1.3B initial records, Source: Certification #: 2017025:
http://www.sap.com/solution/benchmark/appbm/netweaver.sap-bw-edition-for-sap-hana-standard-application.html.