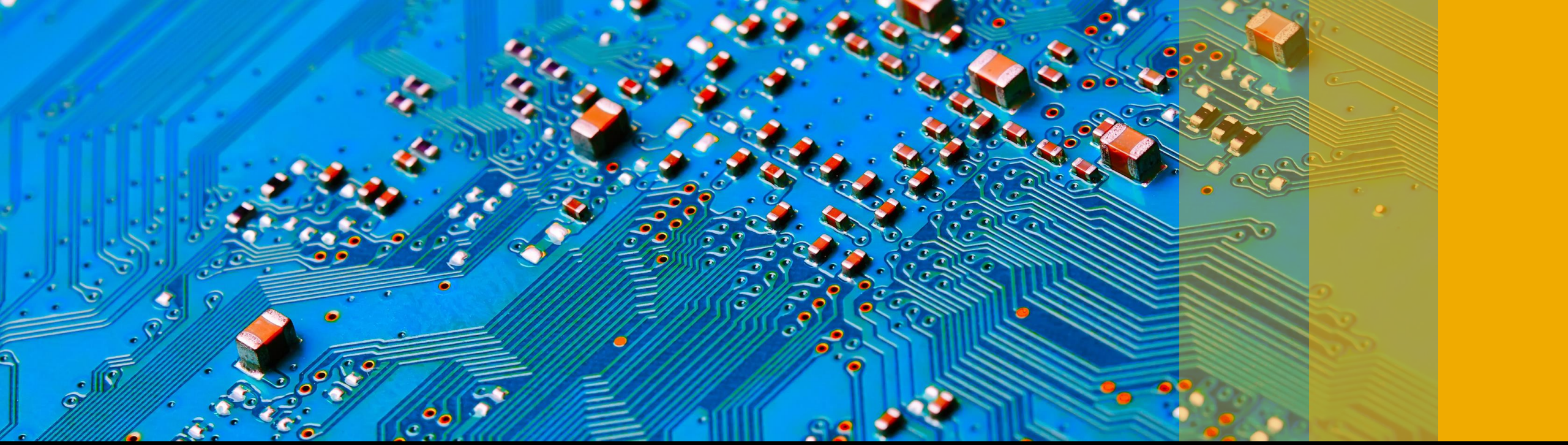


Leveraging Modern Hardware in SAP HANA

How SAP and Intel collaborate to Lead the Edge

Dr. Ismail Oukid (SAP), Dr. Thomas Willhalm (Intel)
January 22, 2018

PUBLIC

SAP Run Simple

intel

Agenda

- Simple Instruction Multiple Data
- Hardware Transactional Memory
- Non-Volatile Memory
- Outlook

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.

SIMD Simple Instruction Multiple Data

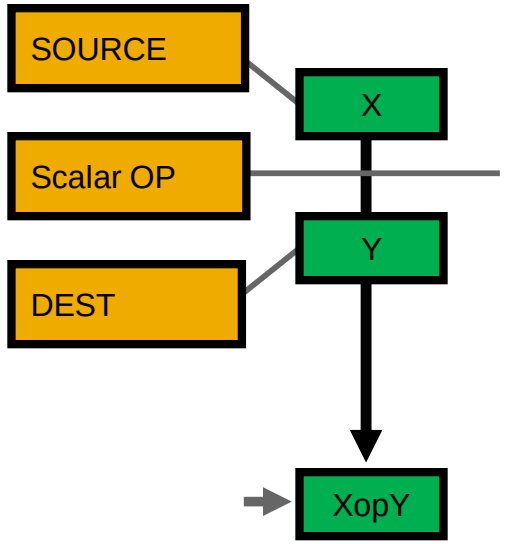
Simple Instruction Multiple Data (SIMD)

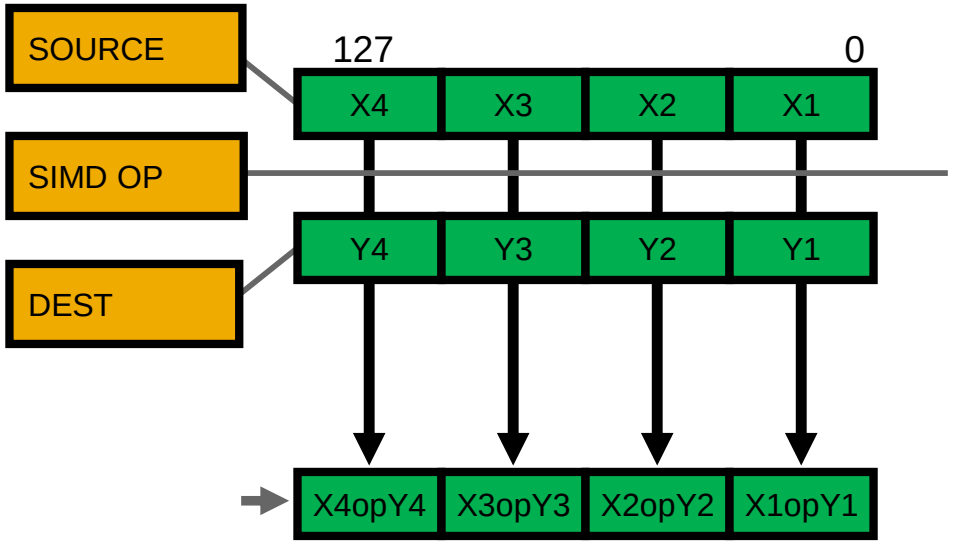
Scalar processing

- traditional mode
- one instruction produces one result

SIMD processing

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



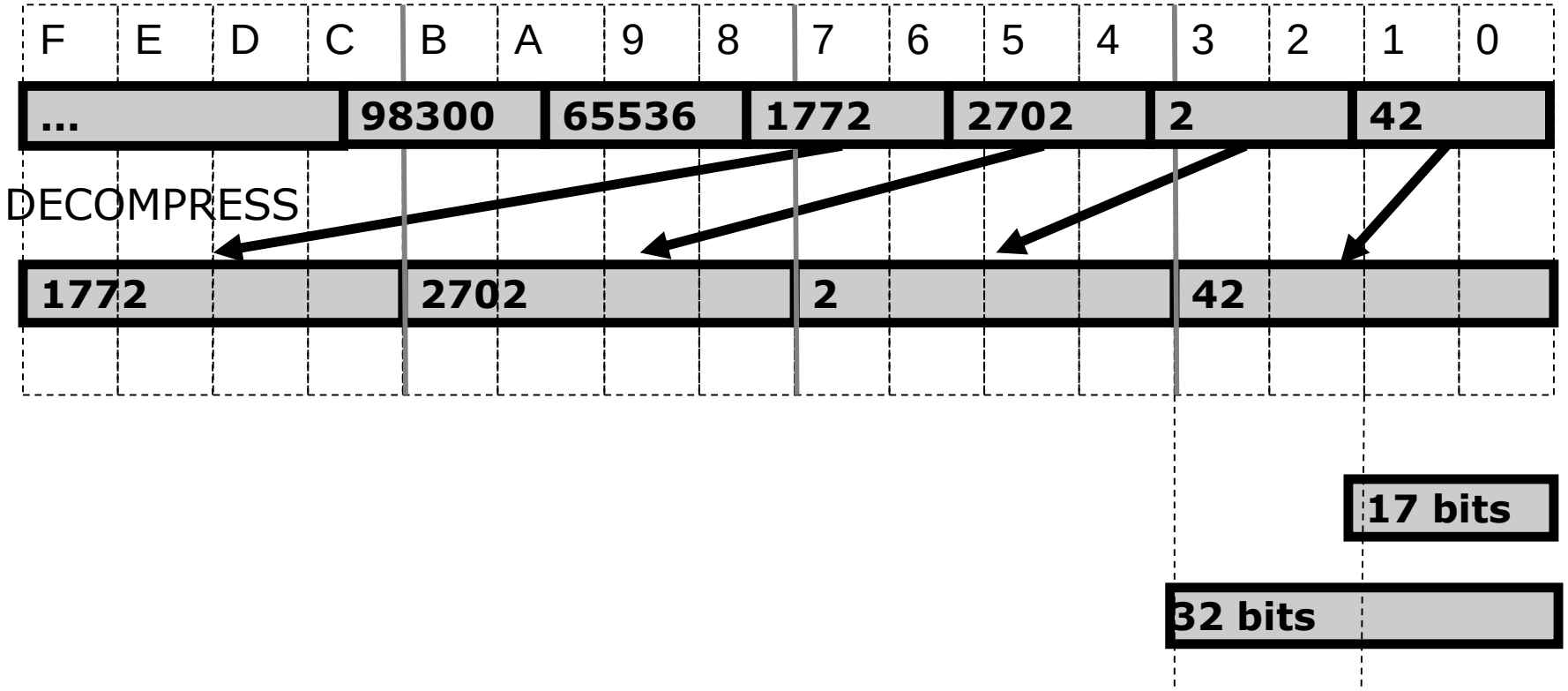


Use-Case: Pack Integers in Bit-Fields

Example: Packed 17-bit fields

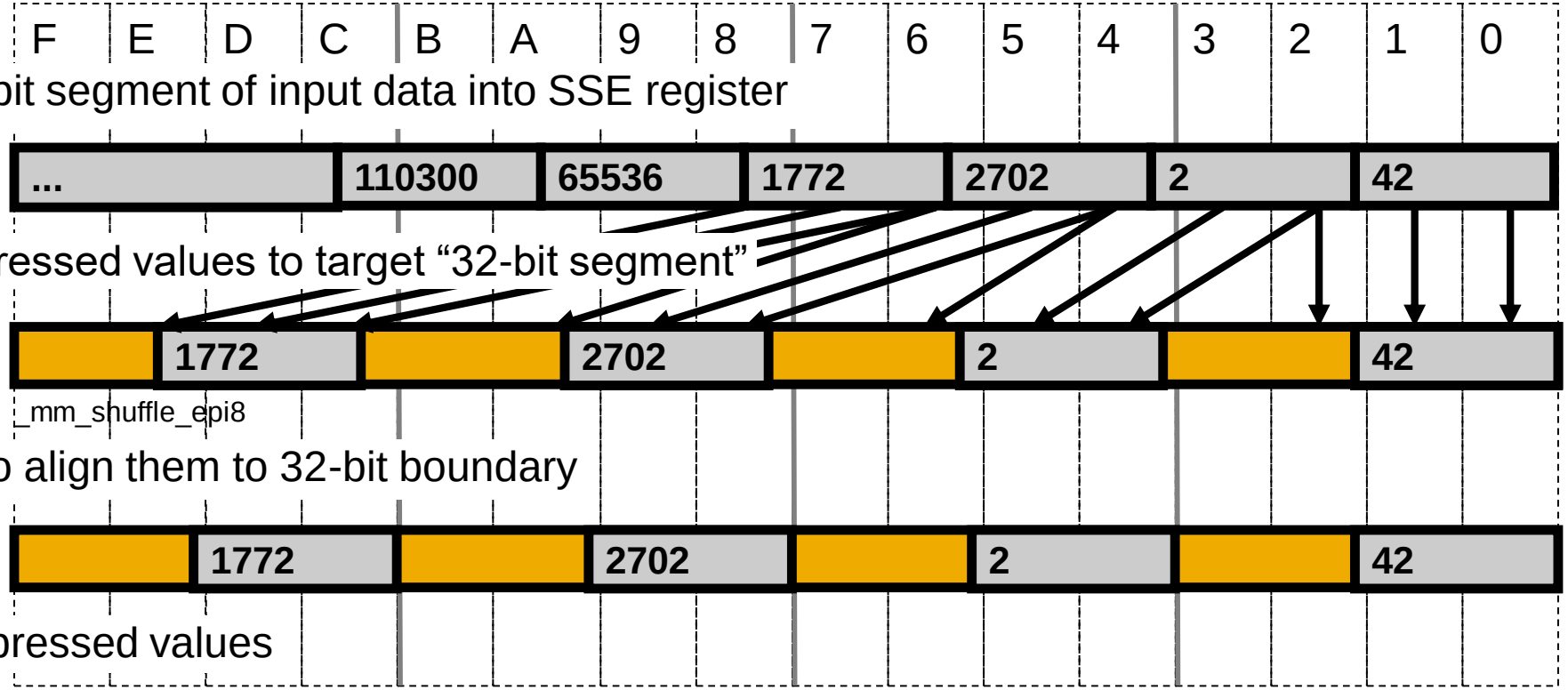
Integers in the range **[0, 100000]** need only **17 bits**

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



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

1. Load a 128-bit segment of input data into SSE register



2. Shuffle compressed values to target “32-bit segment”

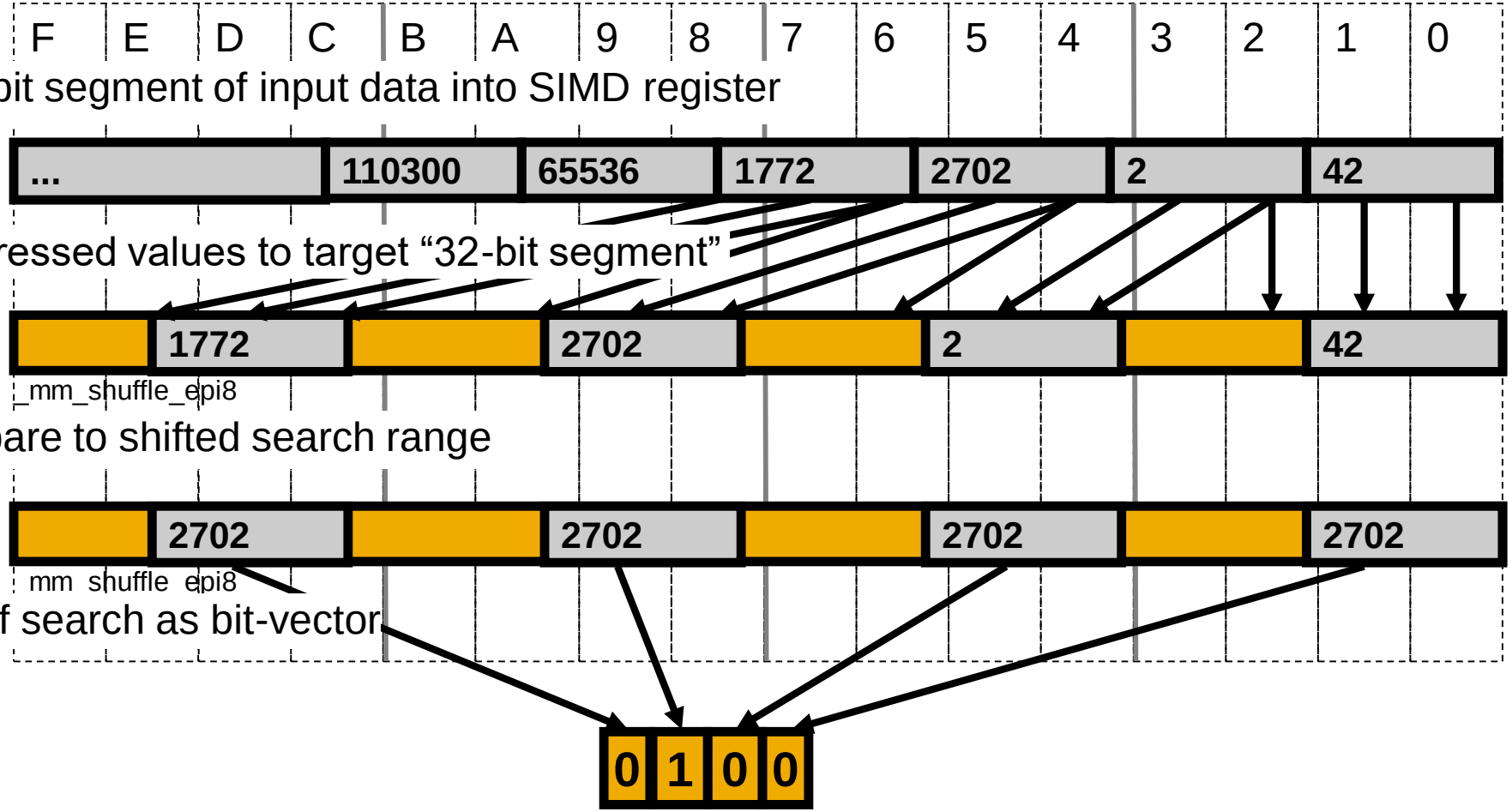
_mm_shuffle_epi8

3. Shift values to align them to 32-bit boundary

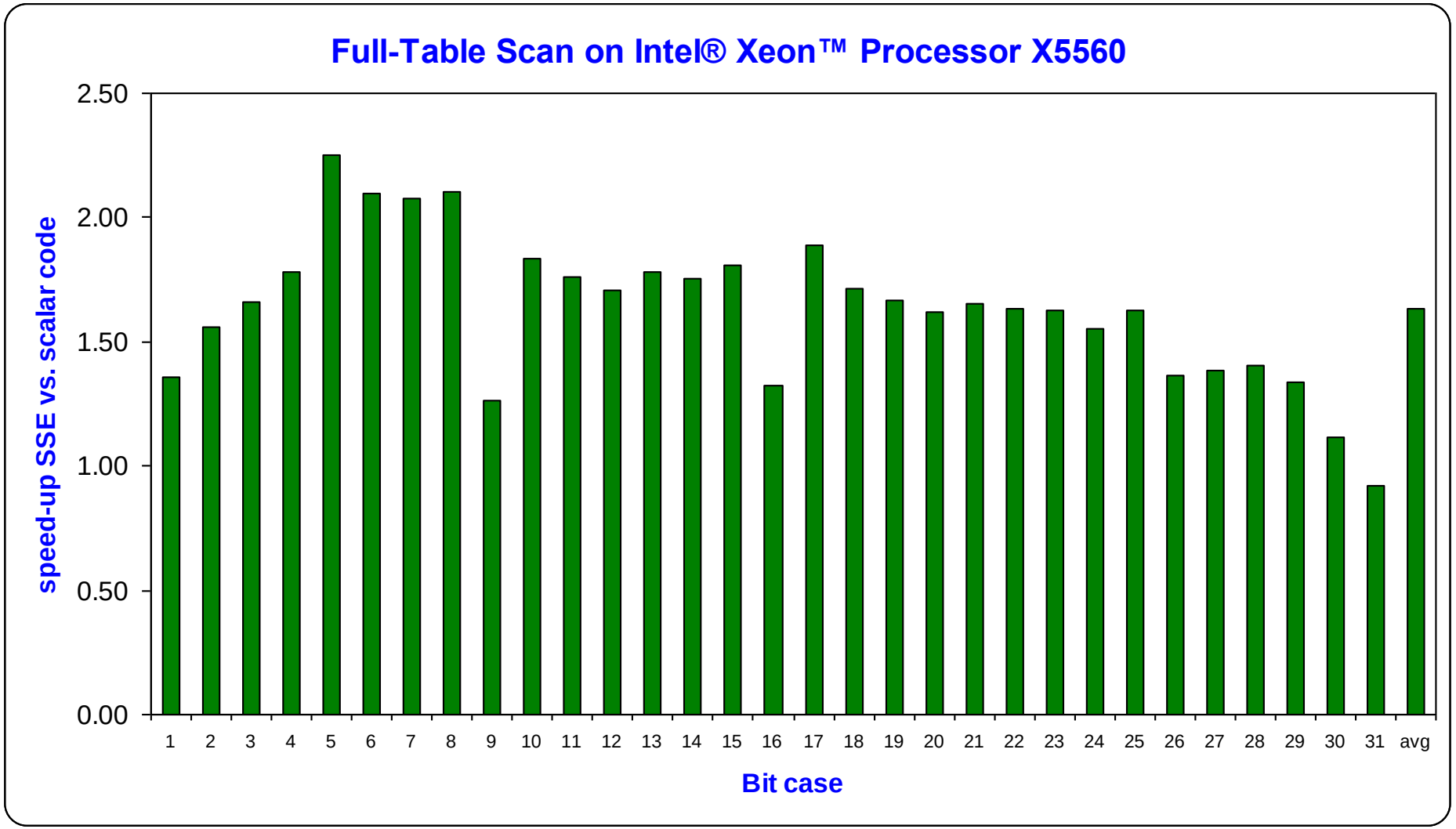
4. Store uncompressed values

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

1. Load a 128-bit segment of input data into SIMD 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

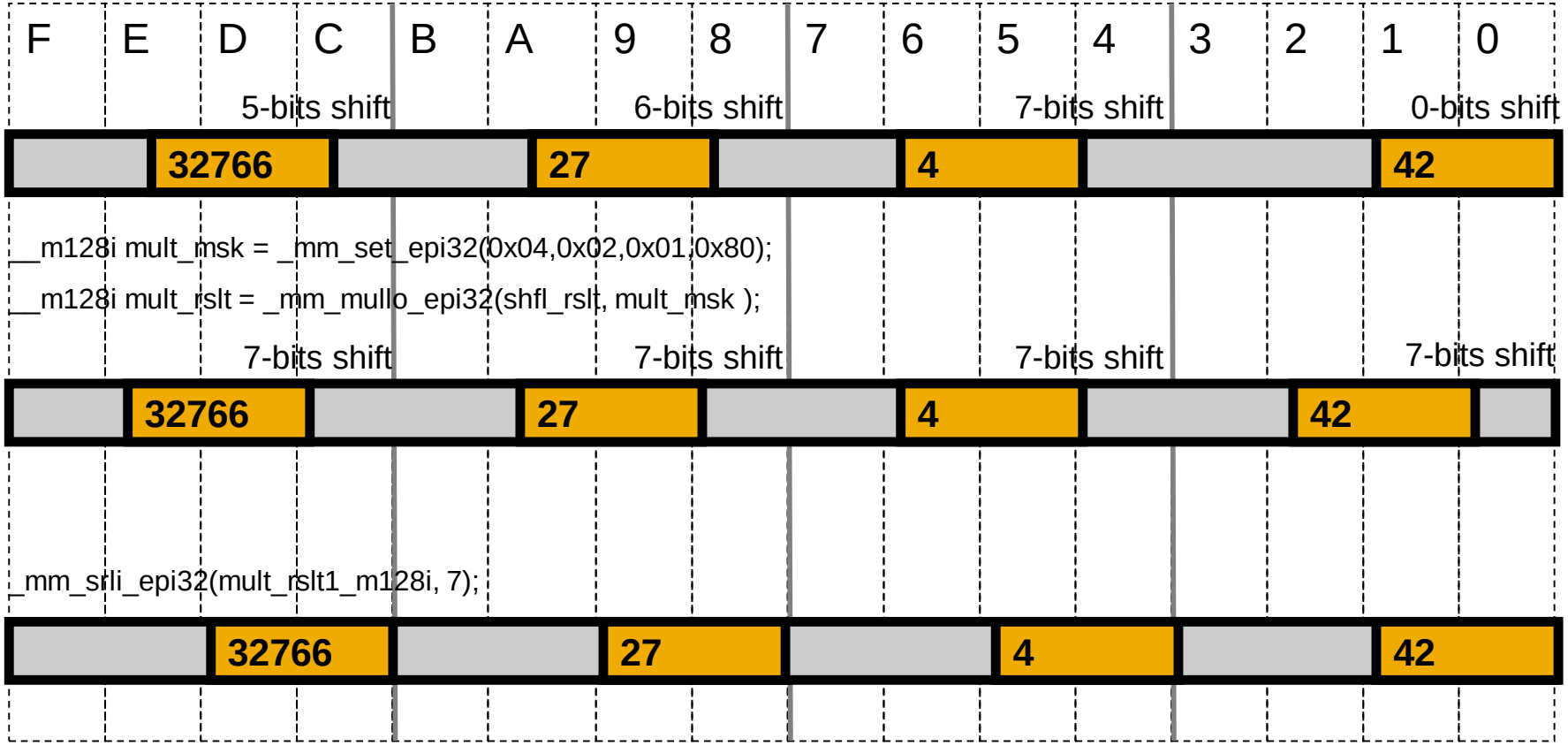


Source: Willhalm et al. SIMD-scan: ultra fast in-memory table scan using on-chip vector processing units. *PVLDB 2009*

Problem: No “vector-vector shift” in SSE

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

Solution: Use multiplication for shifting



Unpacking of Bit-Fields with Intel® AVX2

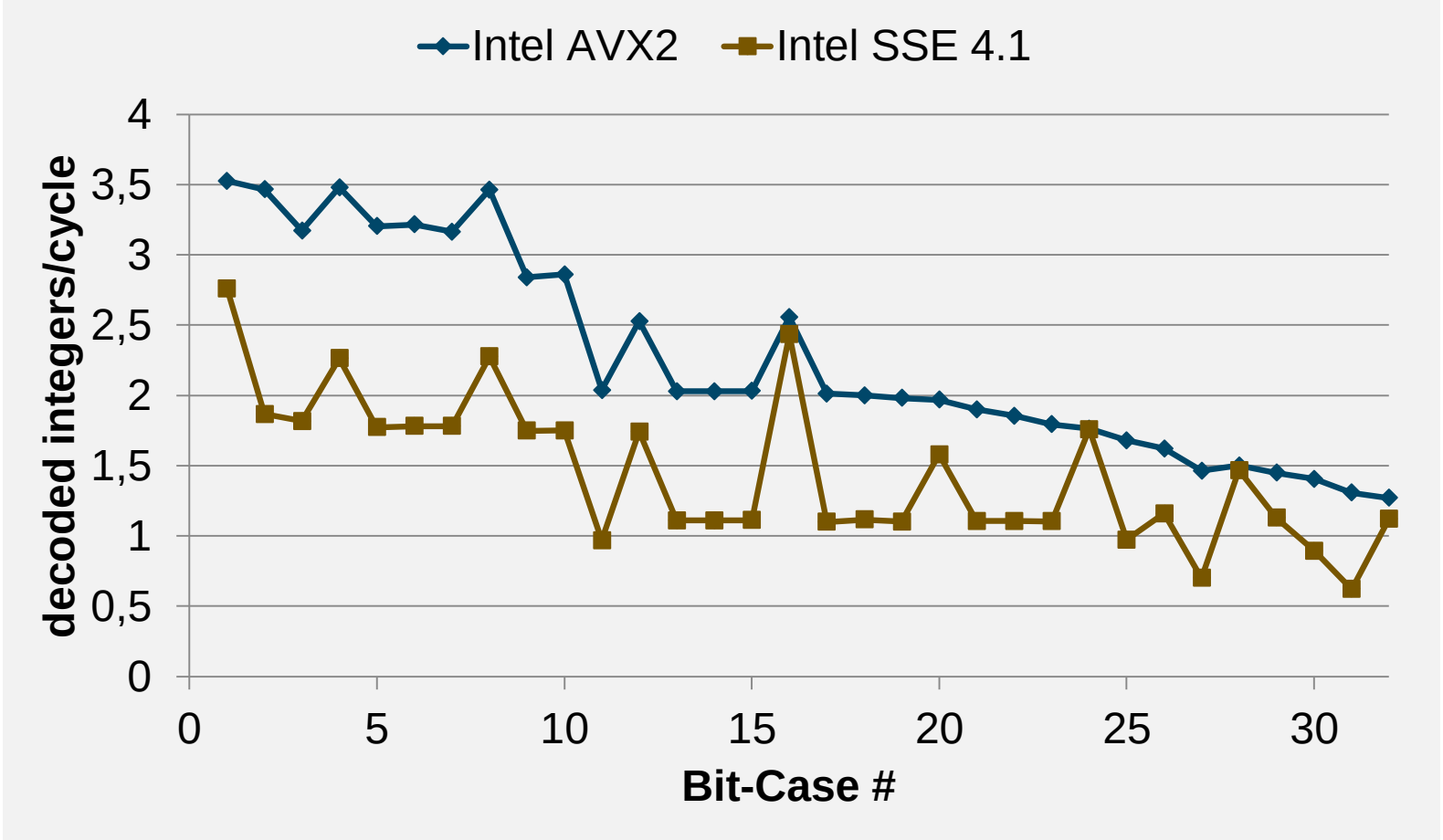
Pseudo Code	SSE 4.1	AVX2
vector load v from input array	`movdqu x8(%r10,%rcx,1),%xmm6`	`vmovdqu xmm8, xmmword ptr[rax+rcx*1+0x11]` `vinserti128 ymm9, ymm8, xmmword ptr [rax+rcx*1+0x19], 0x01`
byte shuffle v	`pshufb %xmm1,%xmm6`	`vpshufb ymm10, ymm9, ymm1`
vector shift v	`pmulld %xmm2,%xmm6` `psrld $0xe,%xmm6`	`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:

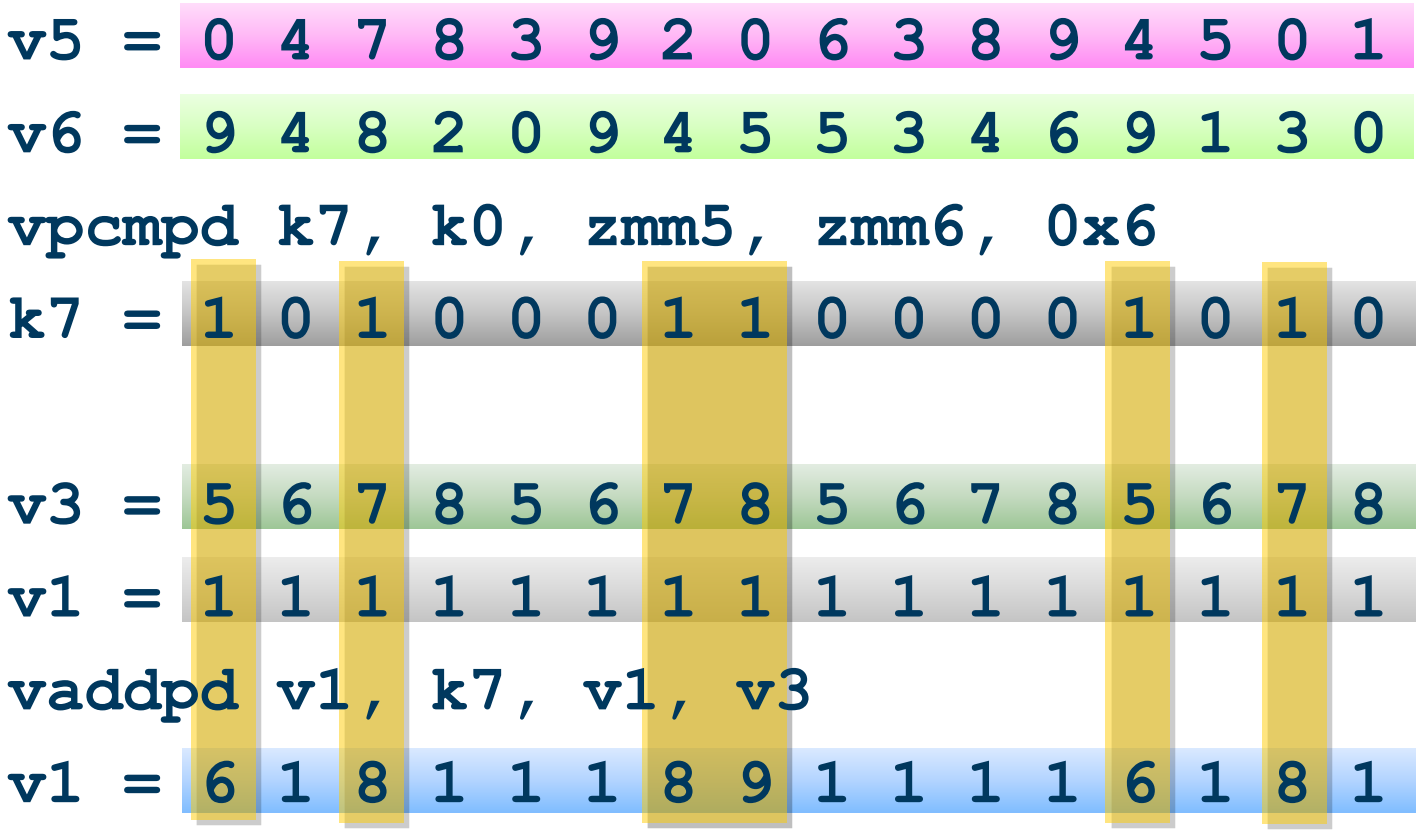
- Results of comparisons
- Masking operations

New instructions:

- vpcompressd – extract selected DWORDs
- Scatter – distribute values

Using Mask Registers for Predication

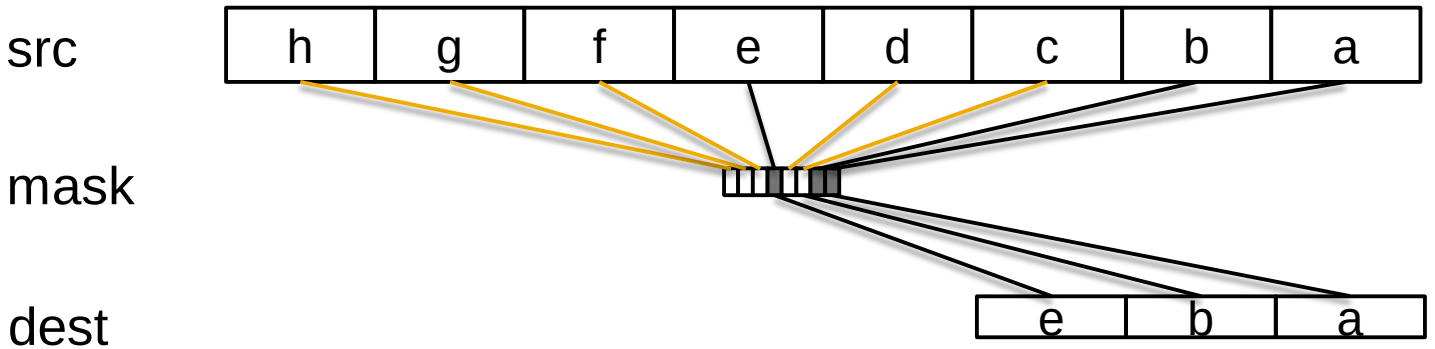
```
if (v5<v6) {v1 += v3;}
```



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

VCOMPRESS

Compress values that are marked in mask:



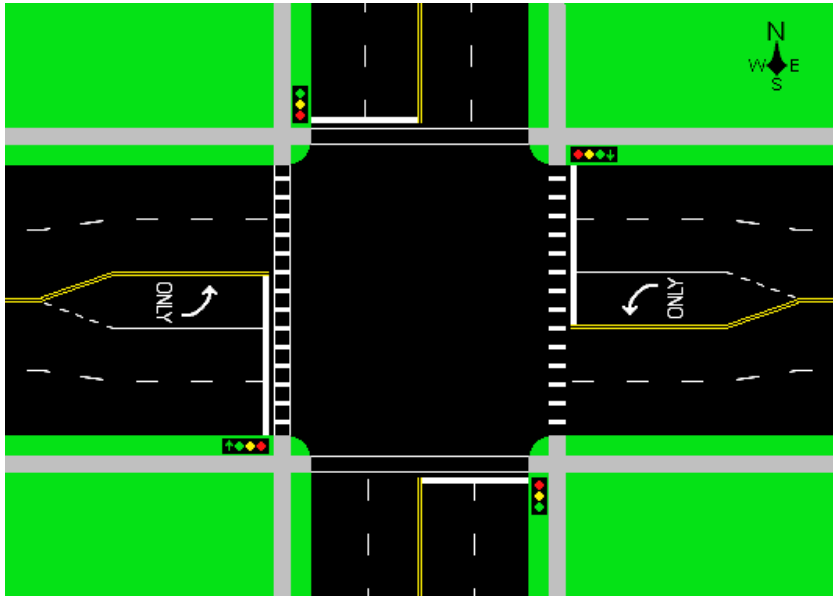
VCOMPRESS generates list of values that are selected by bits in mask (“bit-vector”)

Very useful to generate list of values in a column

- Filter-predicates

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

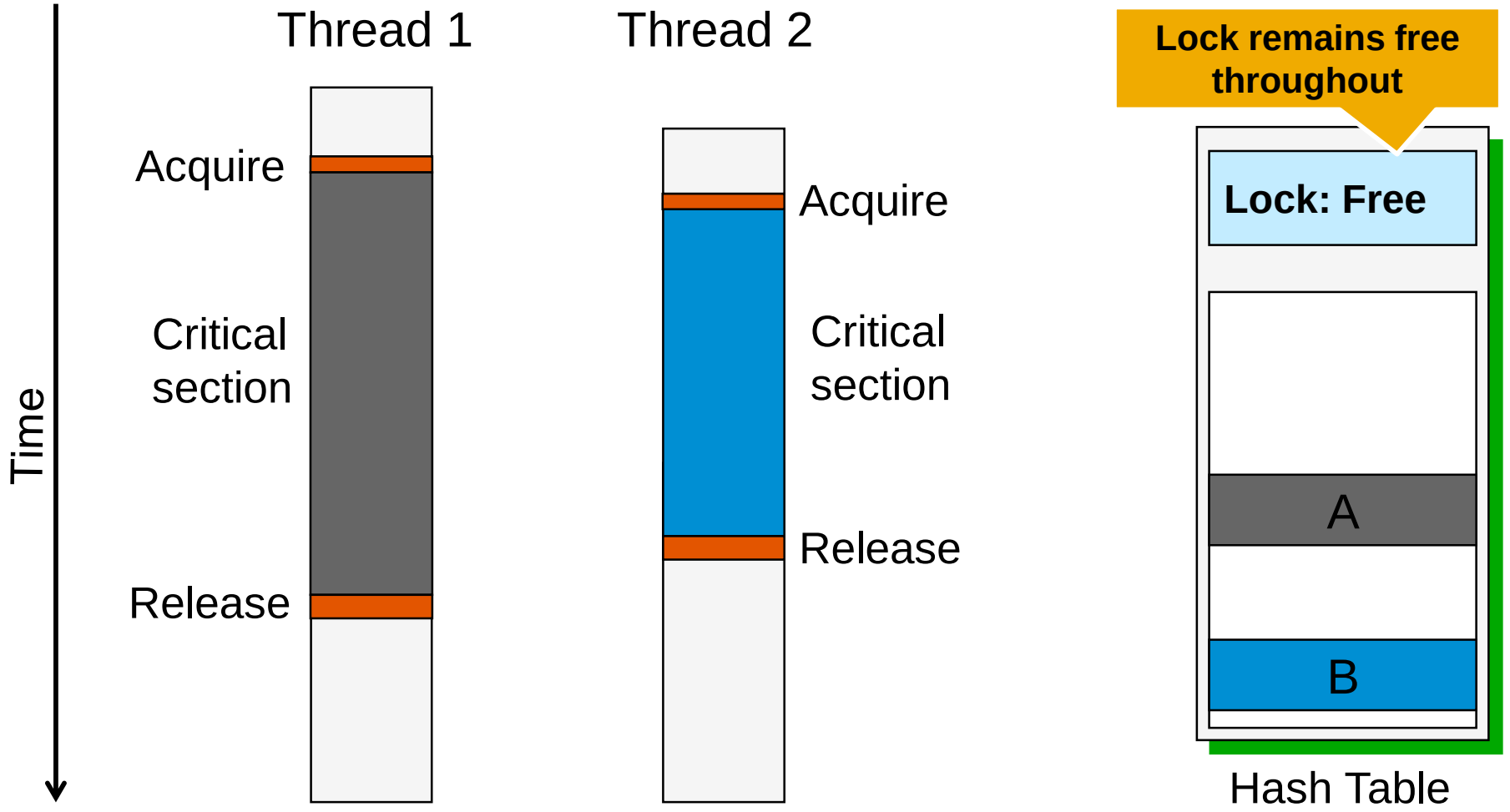
Execute without acquiring lock

- 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

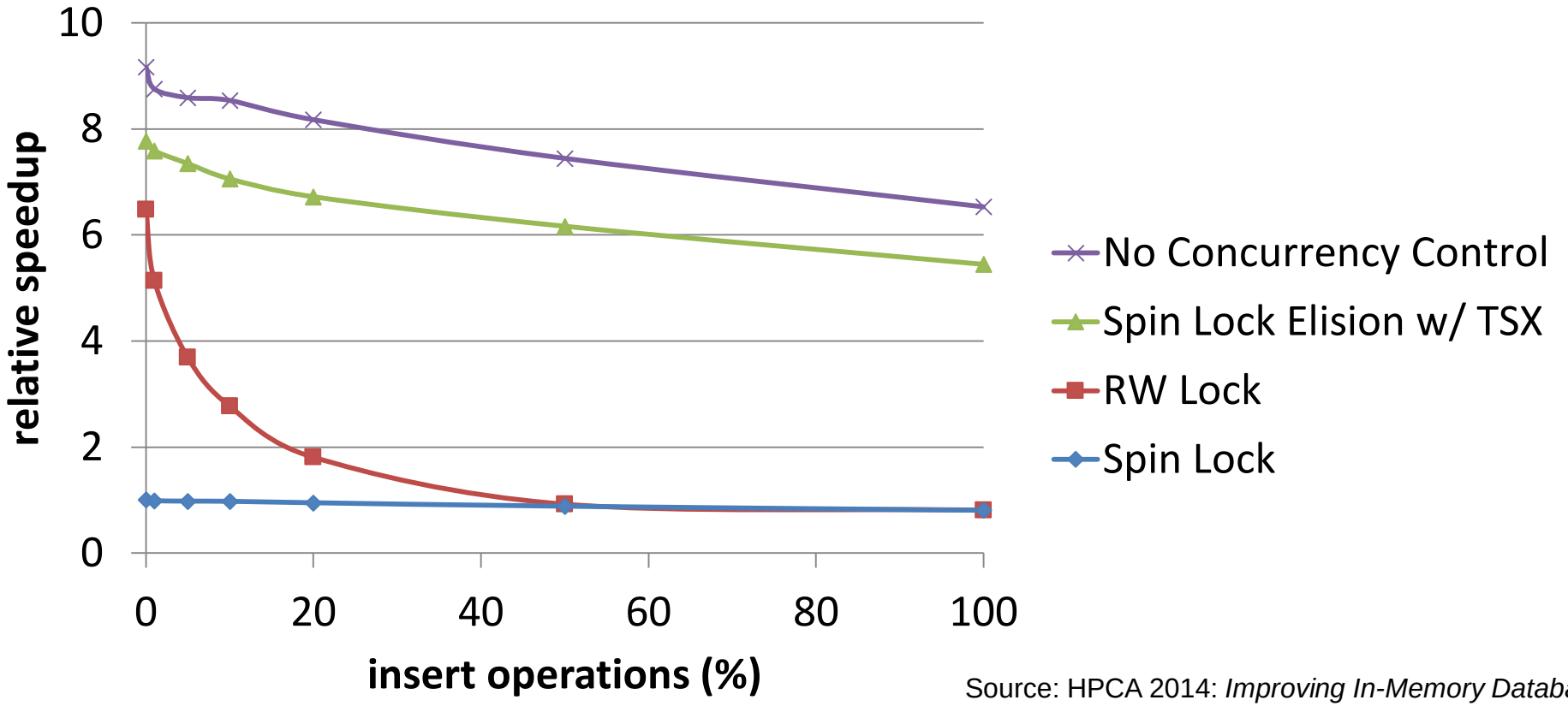


No Serialization and no communication if no data conflicts

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

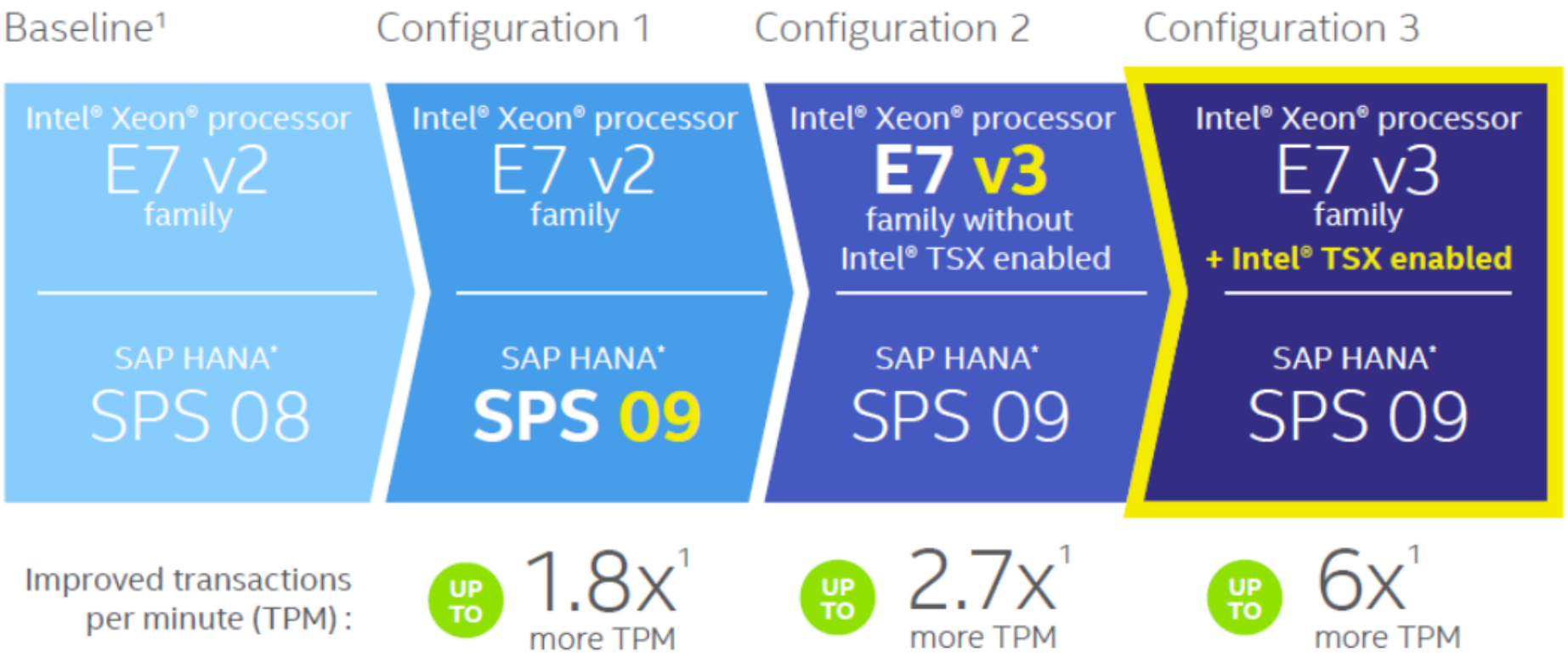


Figure 1. Upgrading to the Intel® Xeon® processor E7 v3 family and SAP HANA* SPS 09 (S-OLTP stress test lab results) provides incremental performance gains.[1]

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.

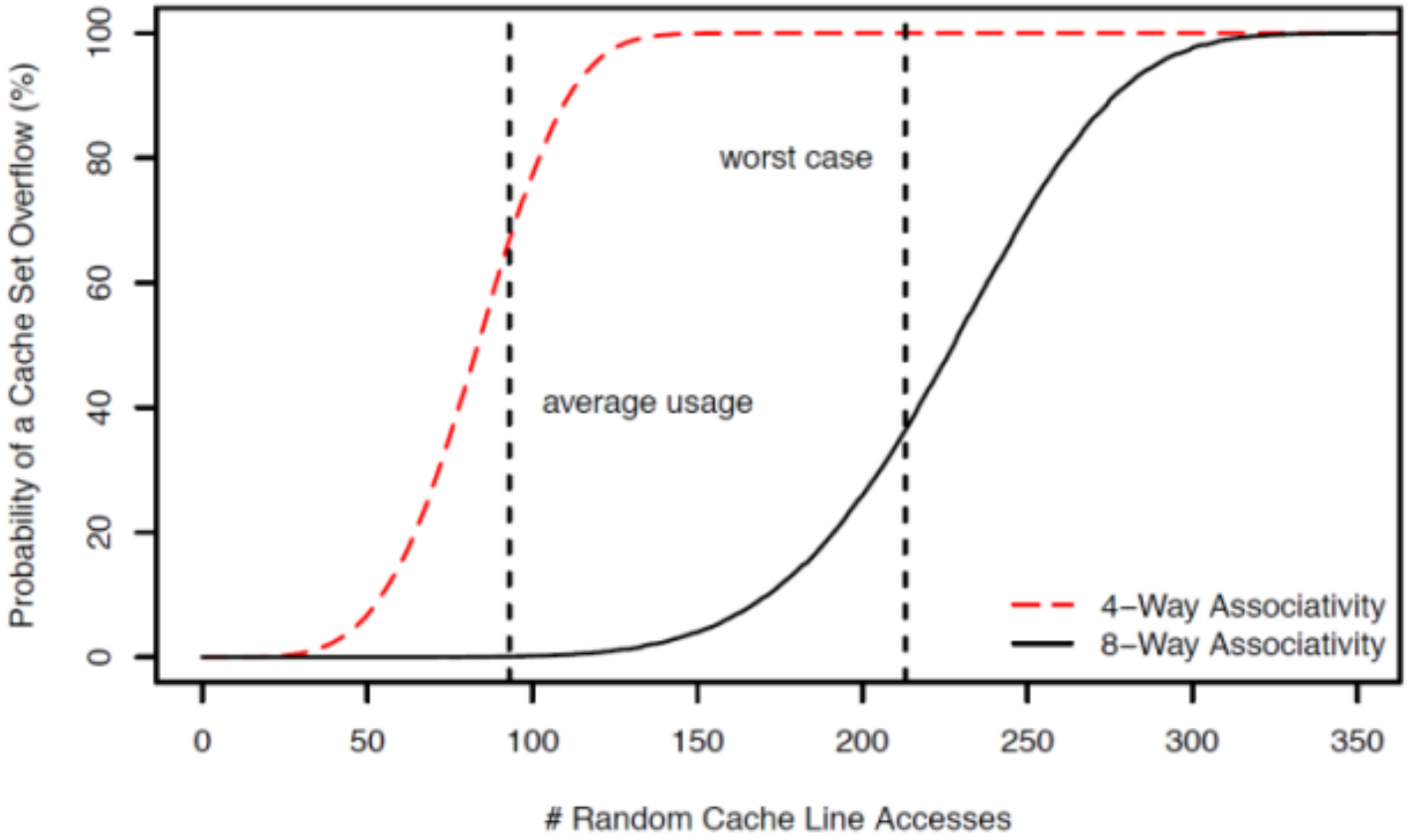
Analysis: TSX Aborts in Delta Storage Index

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



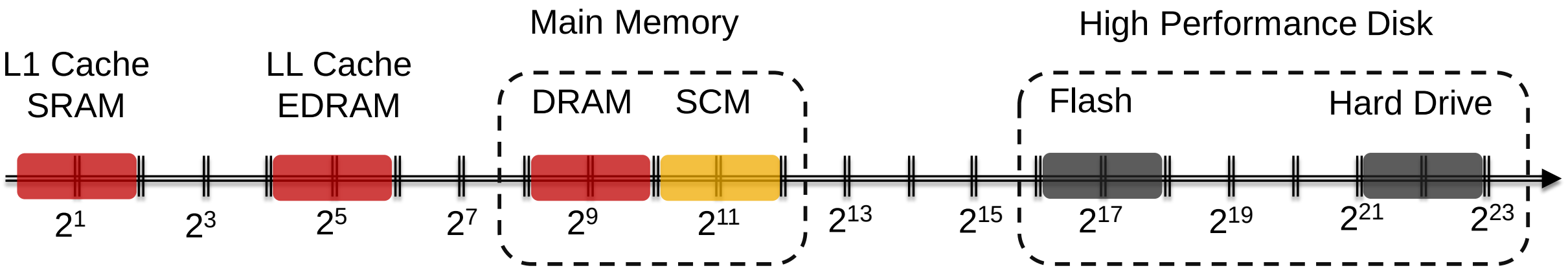
Source: Karnagel et al. *Improving In-Memory Database Index Performance with Intel® Transactional Synchronization Extensions. In* HPCA 2014.

Analysis leading to improvements in future HW generations

NVM Non-Volatile Memory

Storage-Class Memory

SCM is byte-addressable Non-Volatile Memory



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

SCM writes slower than reads

Limited write endurance

SCM 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.

Storage-Class Memory

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

→ SCM as potential remedy

Opportunities:

- Increased scalability
  - Larger memory modules means more memory available per server
- Significant cost savings
  - SCM is 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

Sample Use Case: Reducing In-Memory Database Down-Time

Logging, warehousing, processing information: lifeline of companies

Information availability depends on database availability (9's)

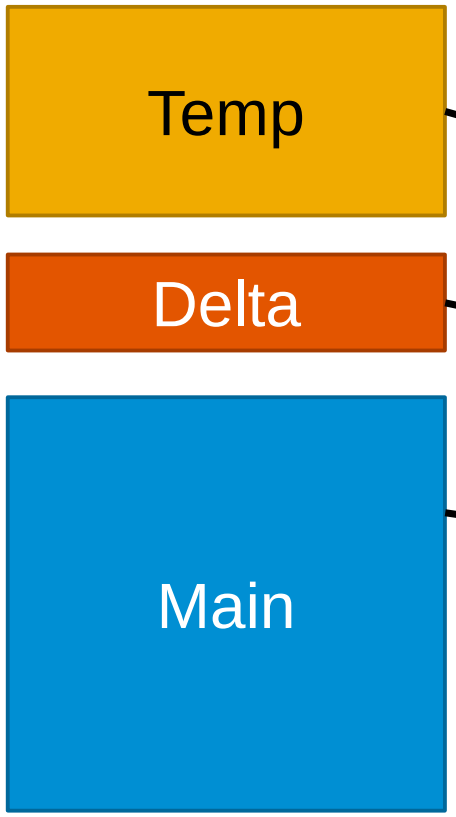
Minimize restart time to improve database availability

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

SAP HANA - Memory Architecture



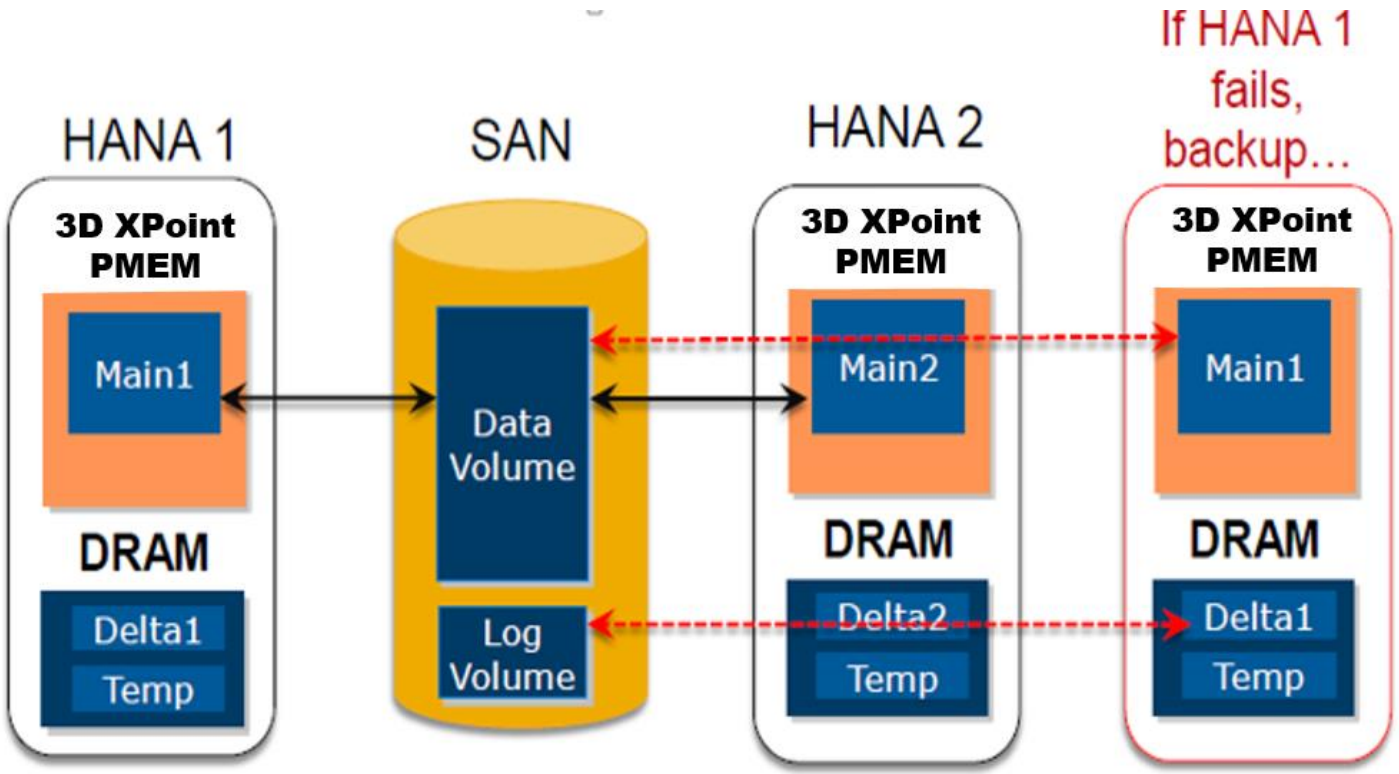
Function	Organization	Access type
Scratch pad for intermediate results	Scratch pad space	Reads and writes
Portions of data that has been modified	~5% of data, less compressed	Reads and writes
Data in memory format	~95% of data, 10-20X compressed	Reads (scans)

Main store contains ~95% of the data in highly compressed format

Main Store is the perfect fit for SCM!

	Technology differentiators	Why Main is well suited
+	Large capacity	Since 95% of data contained in main, HANA can scale up to larger datasets due to increased memory capacities
+	Persistence	Avoid loading data from storage and reduce downtime
–	Higher latencies	References are in form of scans. Hardware and software prefetchers can hide latencies for such reference patterns

Leveraging 3D XPoint™ PMEM technology for SAP HANA Main Store

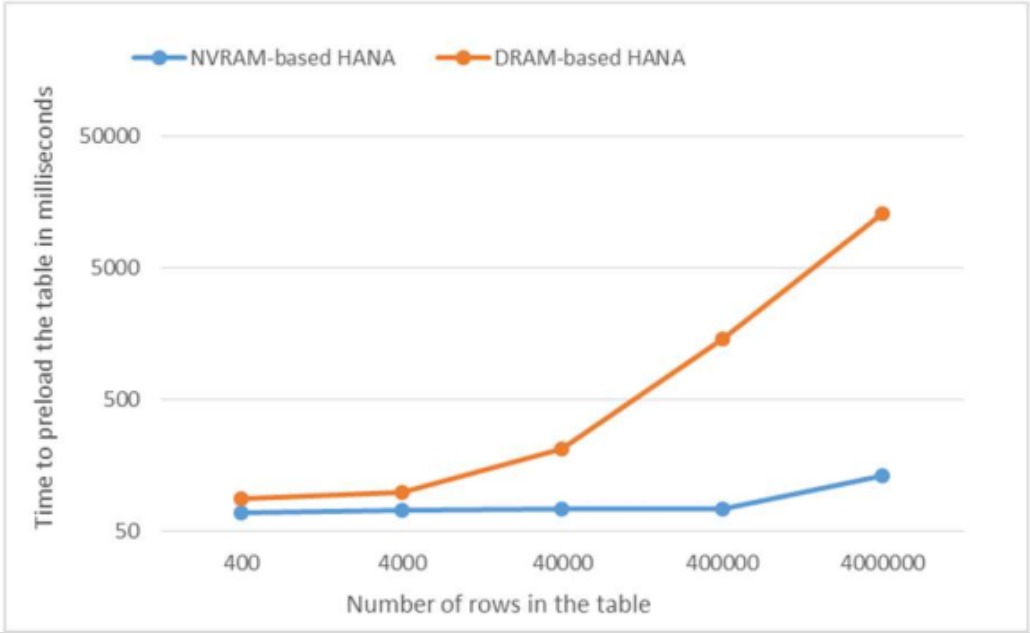


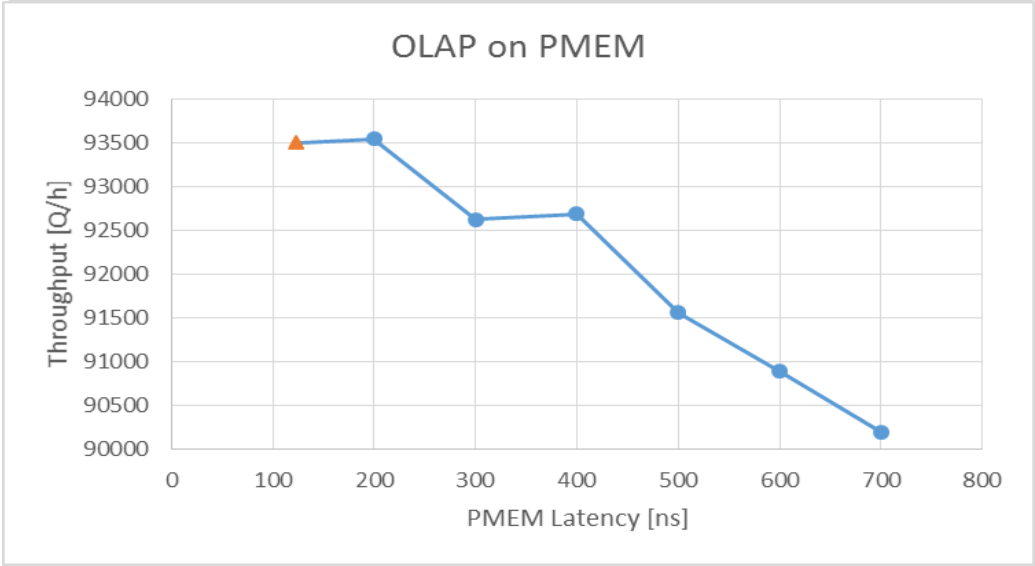
- Primary data store is data volume (in SAN or local storage)
- Main is in 3D XPoint™ PM instead of DRAM and is now persistent
- On Restart: Main already in 3D XPoint™ PM, no need to load data from SAN
- On HW failure: Backup server loads data from data volume

Leveraging 3D XPoint™ PM for SAP HANA: Performance analysis

Promising initial results from a prototype (using HW emulation):

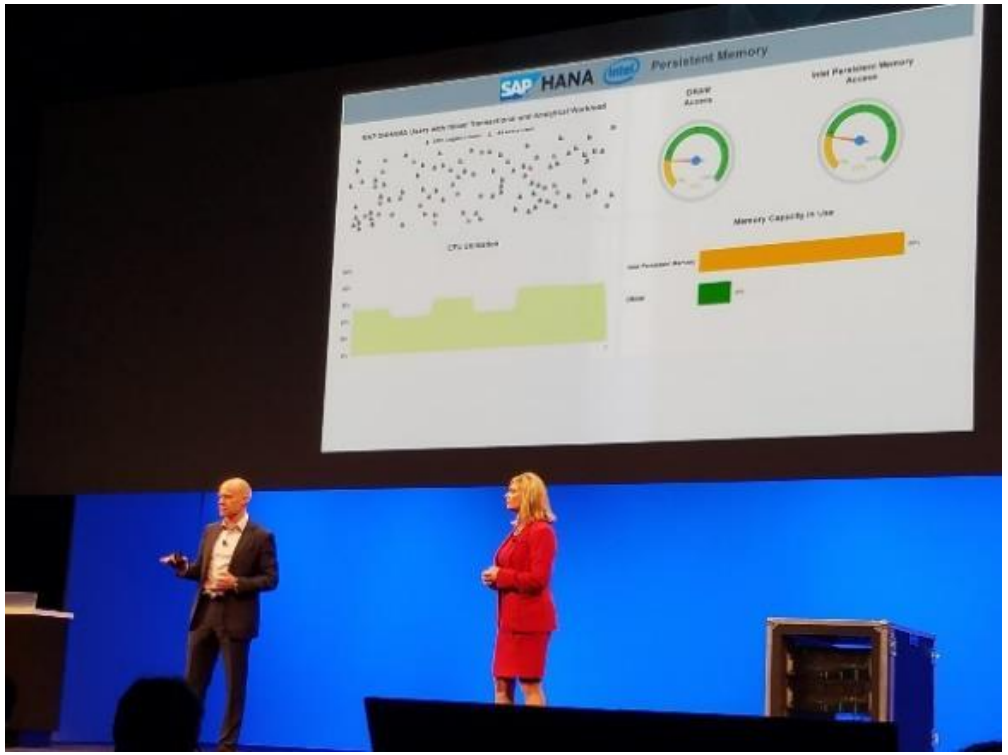
- Significant improvements in the restart time
  - >100x improvement measured
- Acceptable performance impact of higher (than DRAM) latencies, resulting in slightly lower performance
  - Measured (simulated) performance degradation was within the expected range for most workloads





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

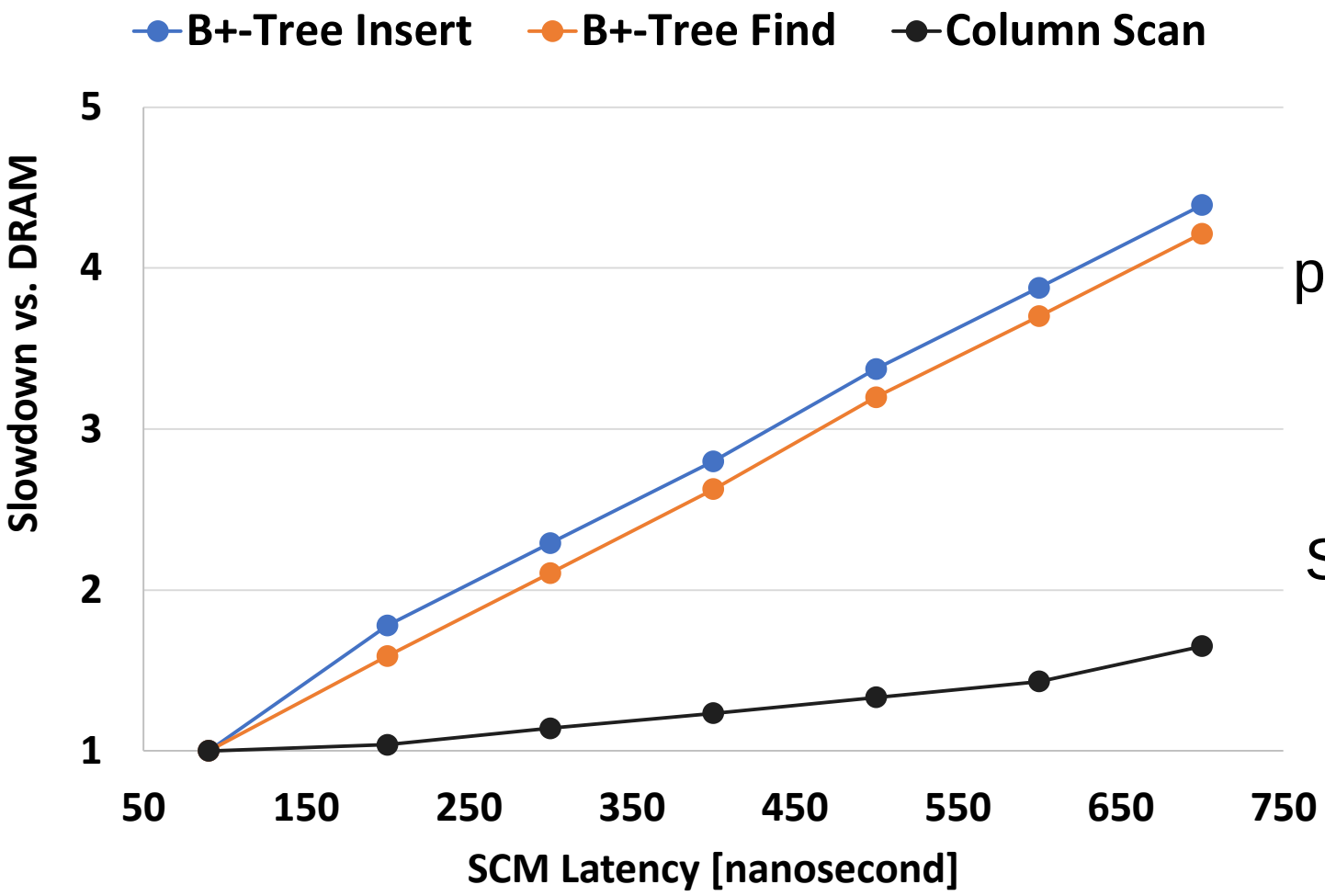
World‘s first Intel Persistent Memory Demo at Sapphire 2017



- High capacity for scalability
- Data persistence without disk I/O
- Lower cost
- Less downtime

Next step: Mutable data structures on SCM

SCM Performance Implications



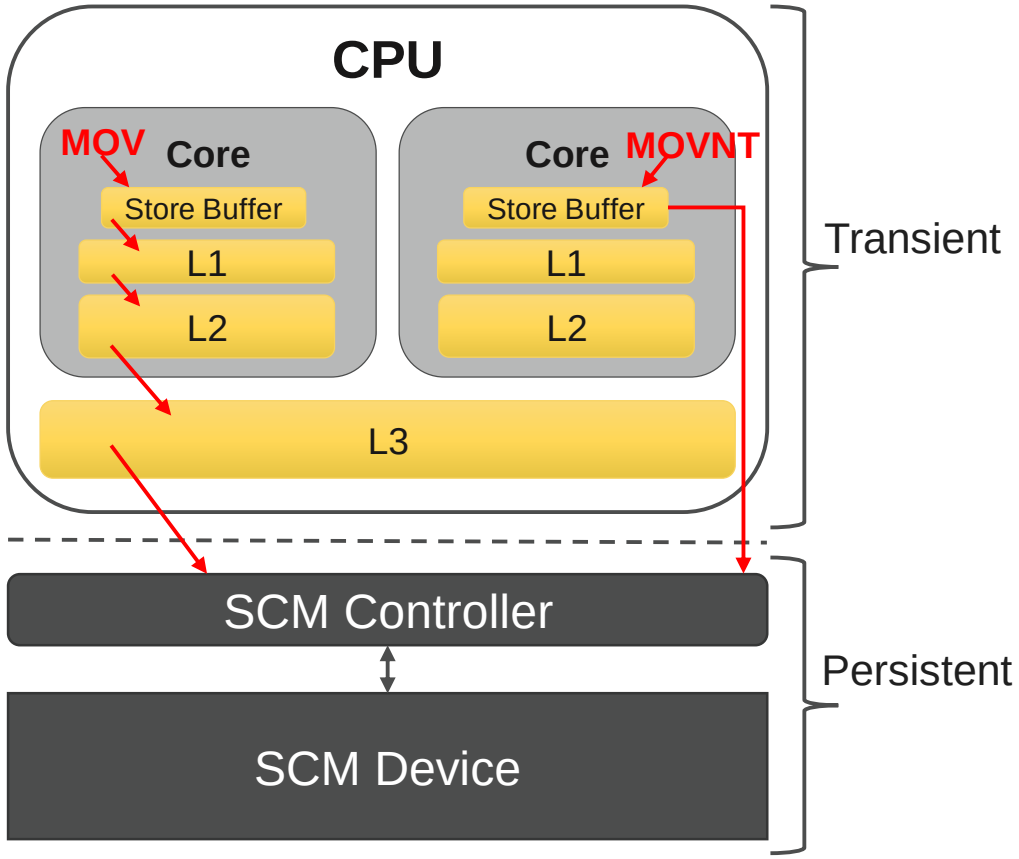
Random memory access patterns significantly affected

Sequential memory access patterns hardly affected

Need to keep DRAM next to SCM

SCM 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

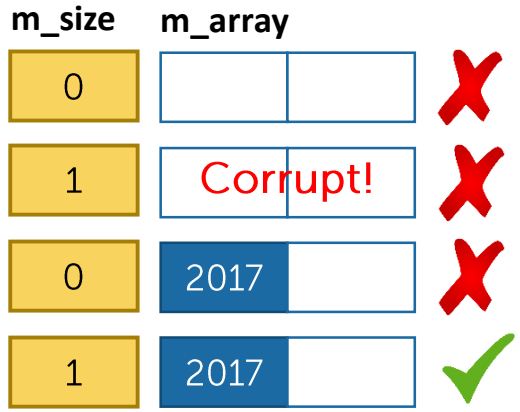


How to persist data in SCM?

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 SCM?



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

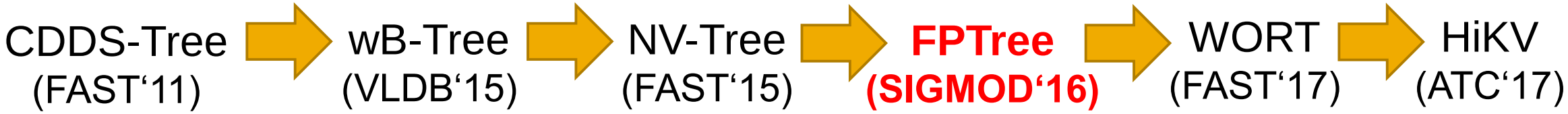
Pros:
- Easy to use and to reason about

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

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

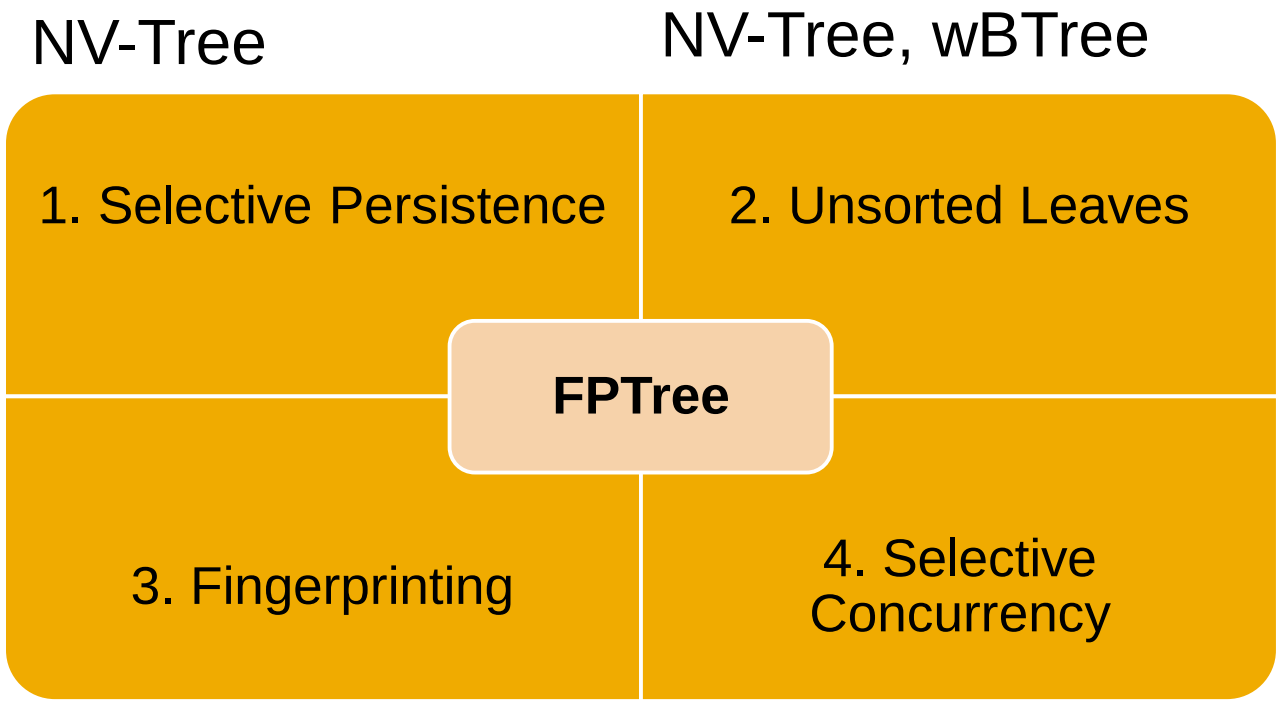
Literature focuses mostly on tree-based data structures
→ Failure-atomic updates
→ Reduce SCM writes

Literature timeline



FPTree Design Goals

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

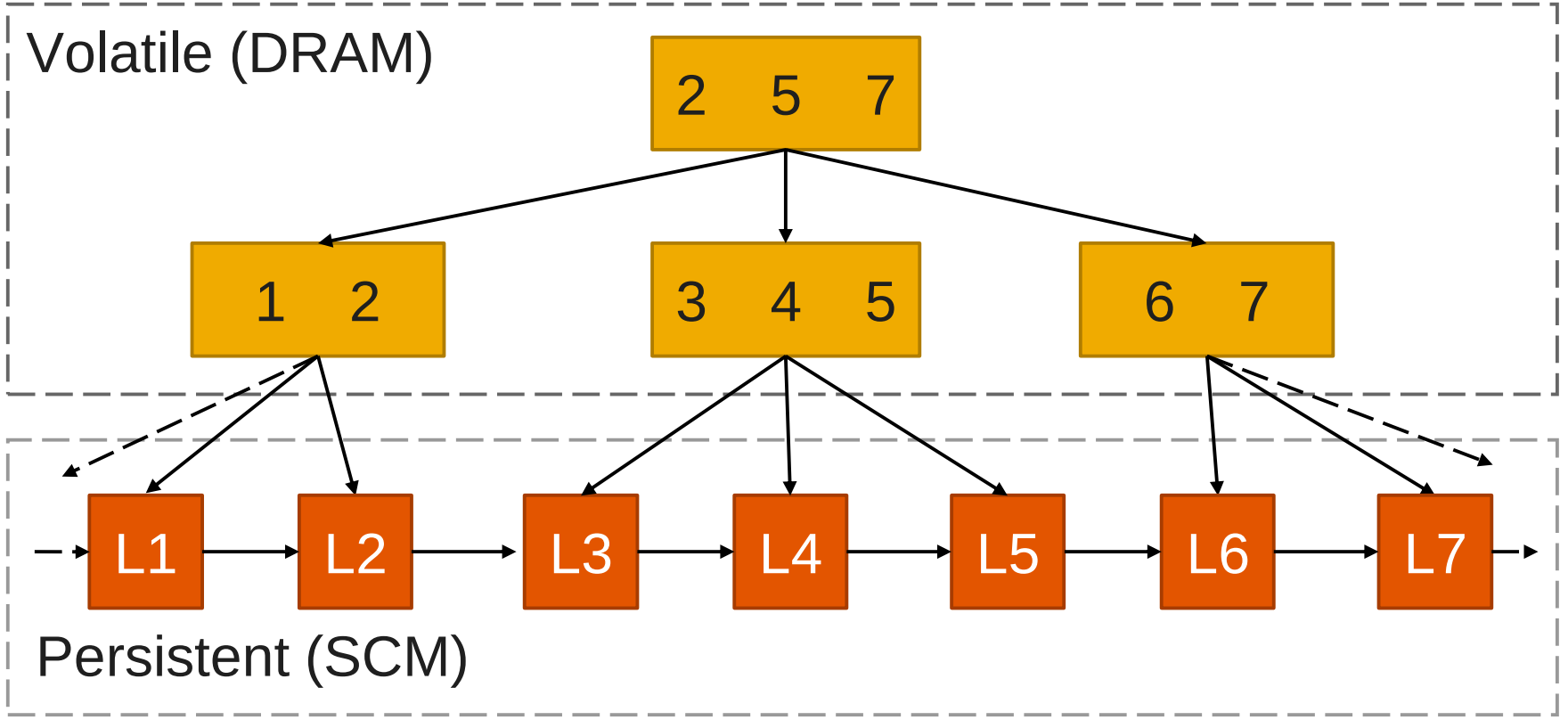


Source: Oukid et al. *FPTree: A Hybrid SCM-DRAM Persistent and Concurrent B-Tree for Storage-Class Memory*. In SIGMOD 2016.

1. Selective Persistence

Inner nodes in DRAM for better performance

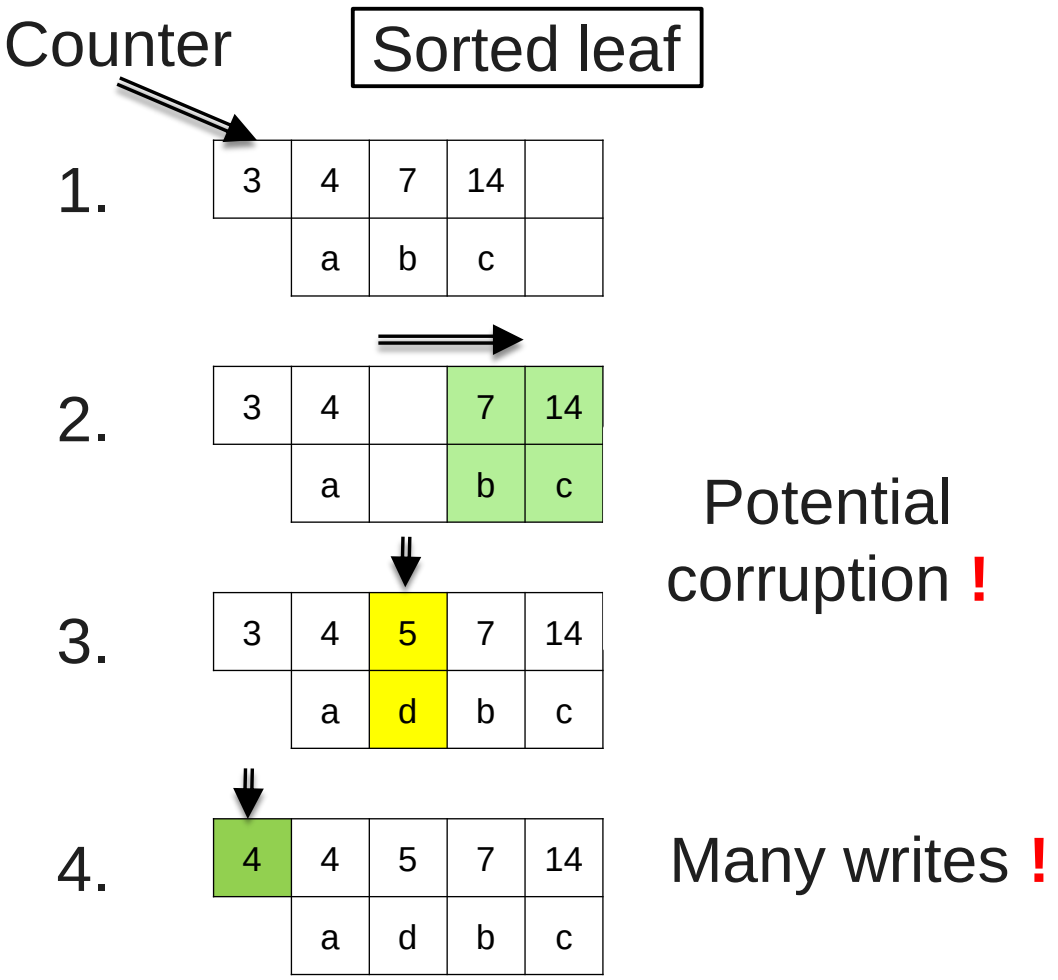
Leaves in SCM to ensure durability



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

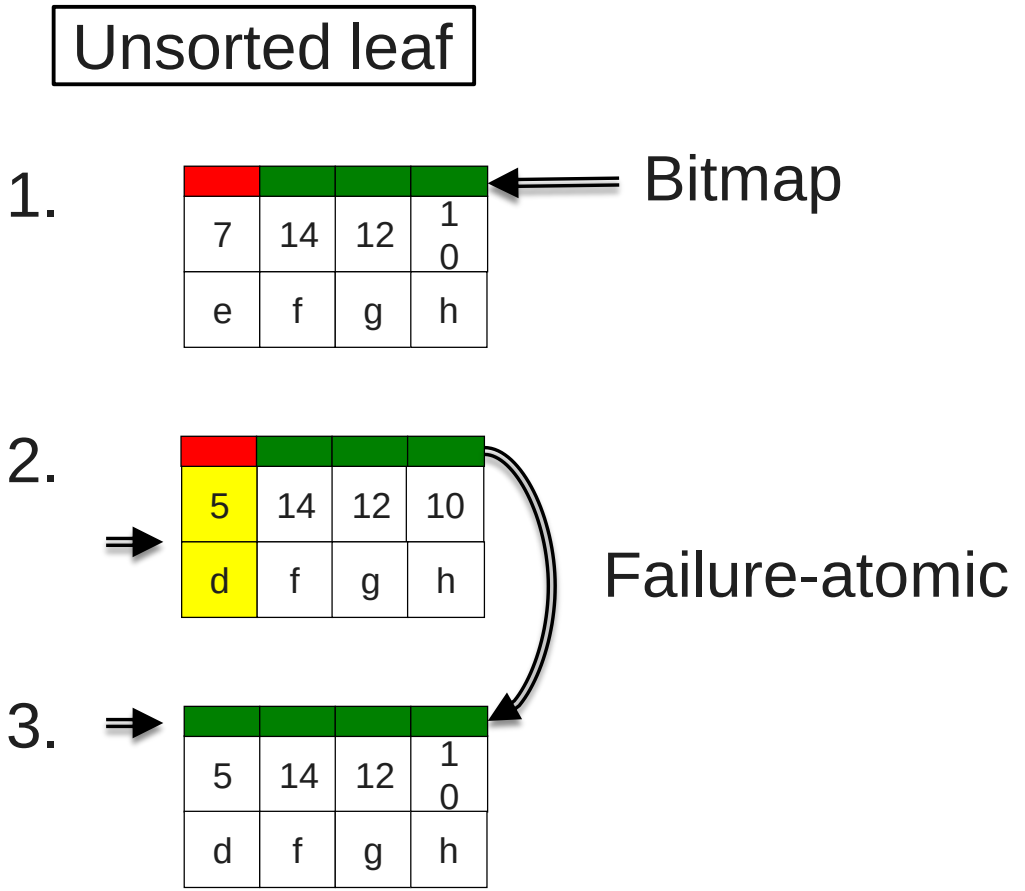
Recovery is up to 100x faster than a full rebuild

2. Unsorted Leaves



Potential corruption !

Many writes !

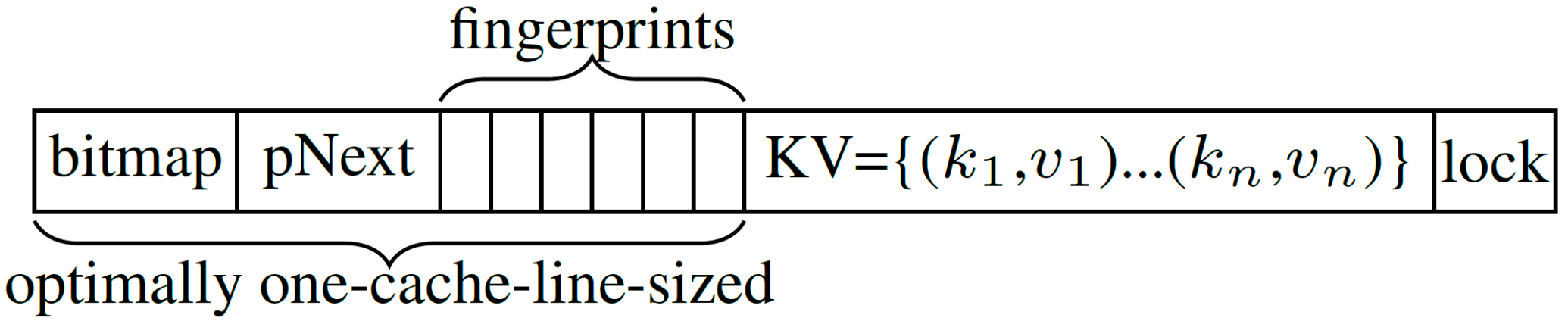


Failure-atomicity + fewer writes

But…linear search instead of binary search

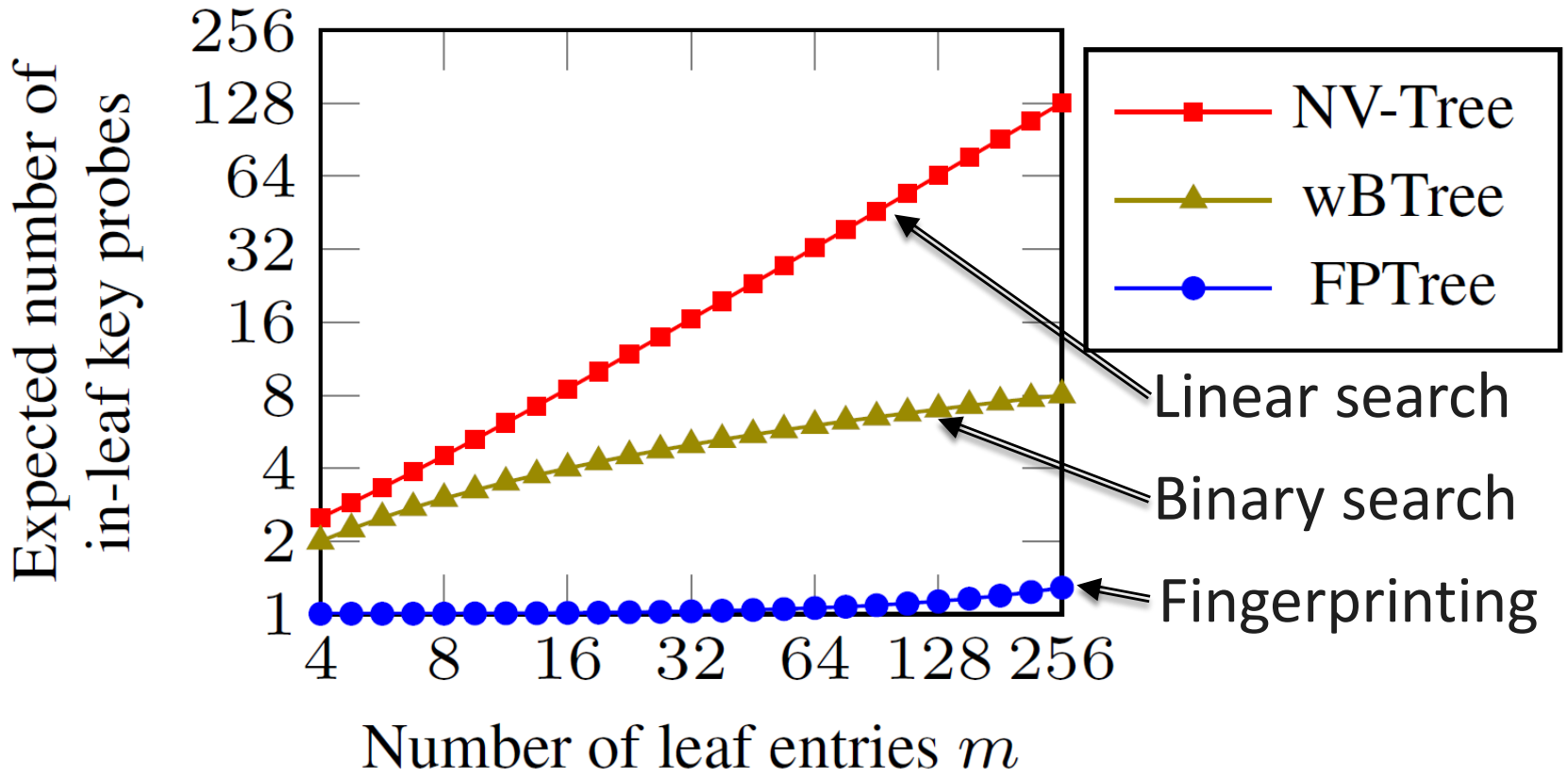
3. Fingerprinting

A fingerprint is a 1-byte hash of a key



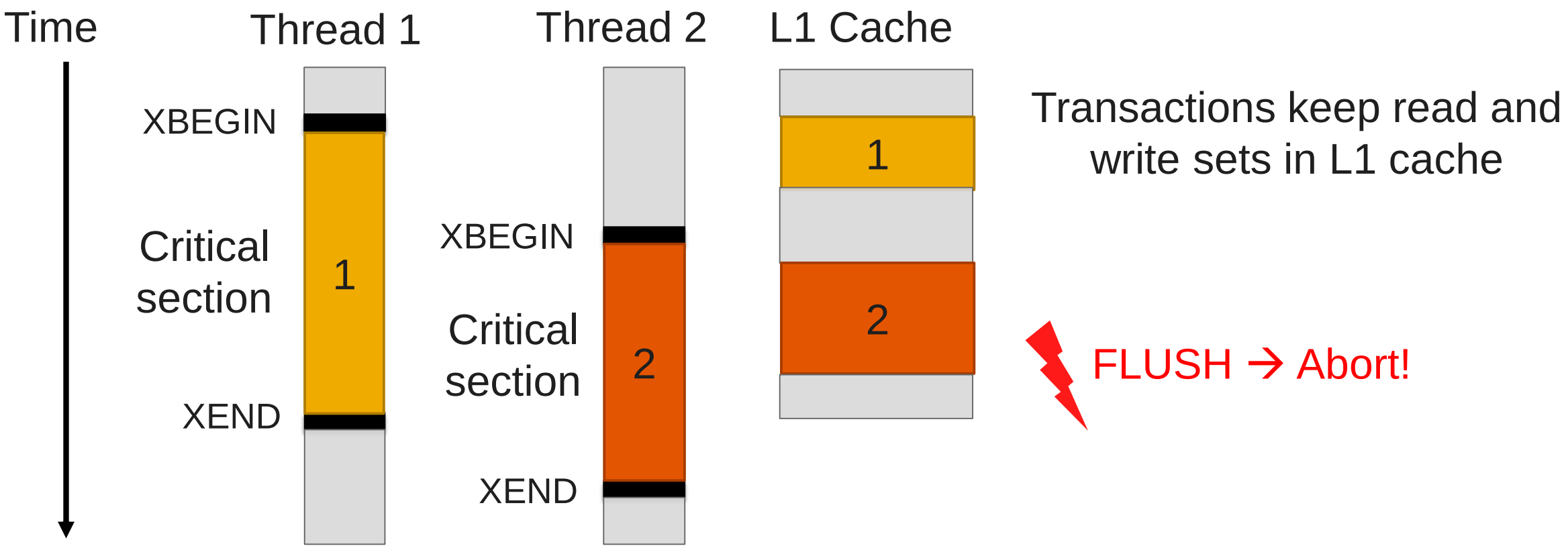
Fingerprints act as a filter to limit the number of key probes

3. Fingerprinting



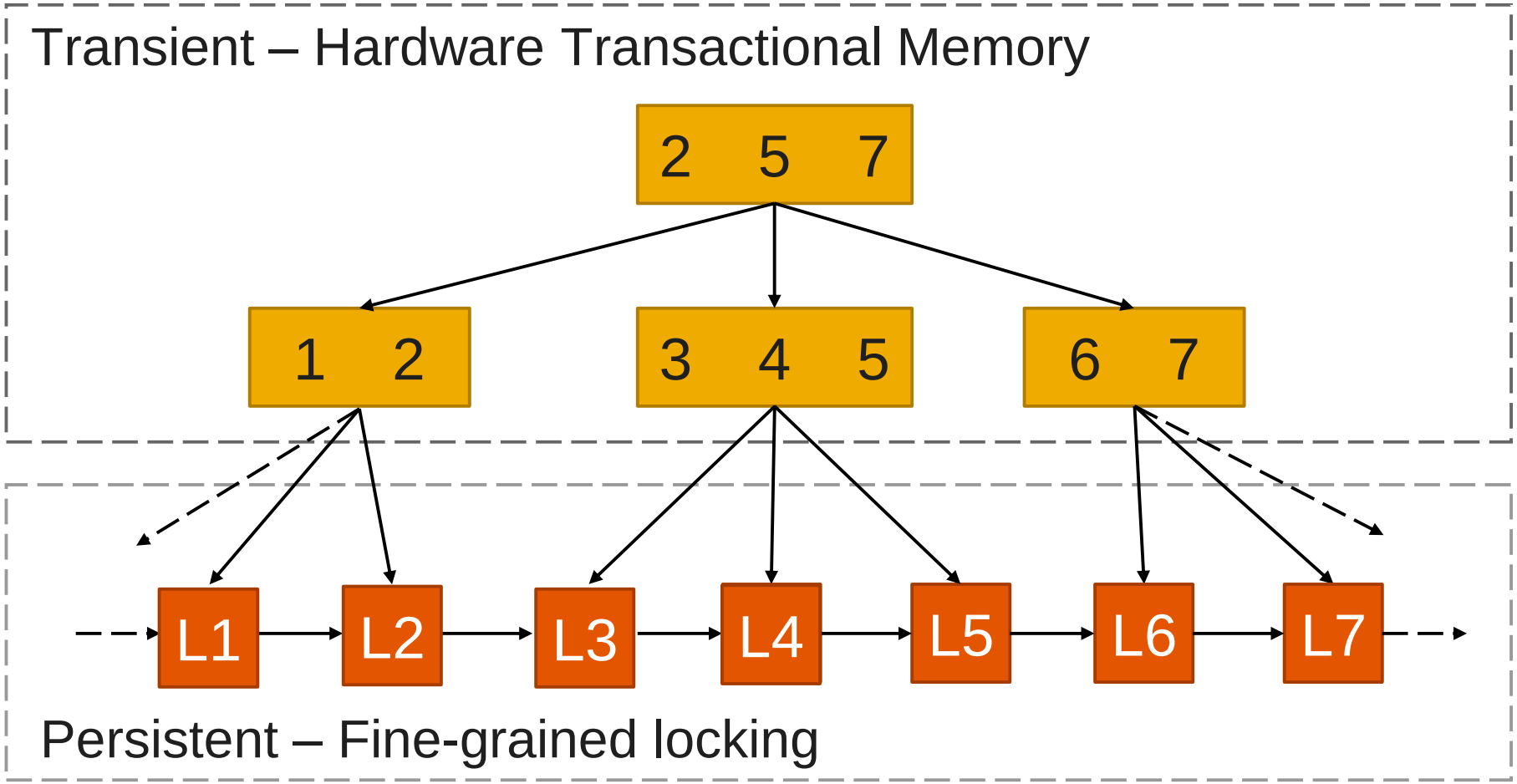
Expected number of probed keys is one for leaf sizes up to 64

Hardware Transactional Memory (HTM)

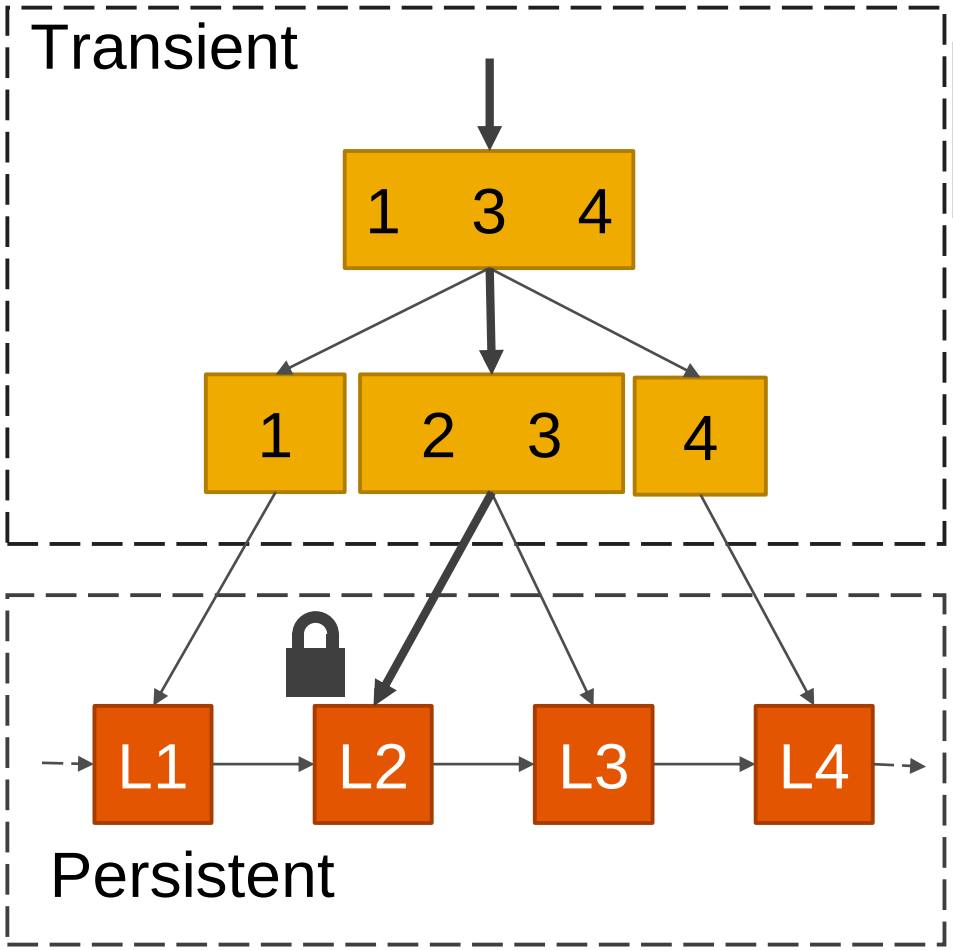


HTM and SCM are apparently incompatible

4. Selective Concurrency



4. Selective Concurrency: Insertion



1. Find and lock leaf	2. Modify leaf	3. Update parents	4. Unlock leaf

XBEGIN

FLUSH

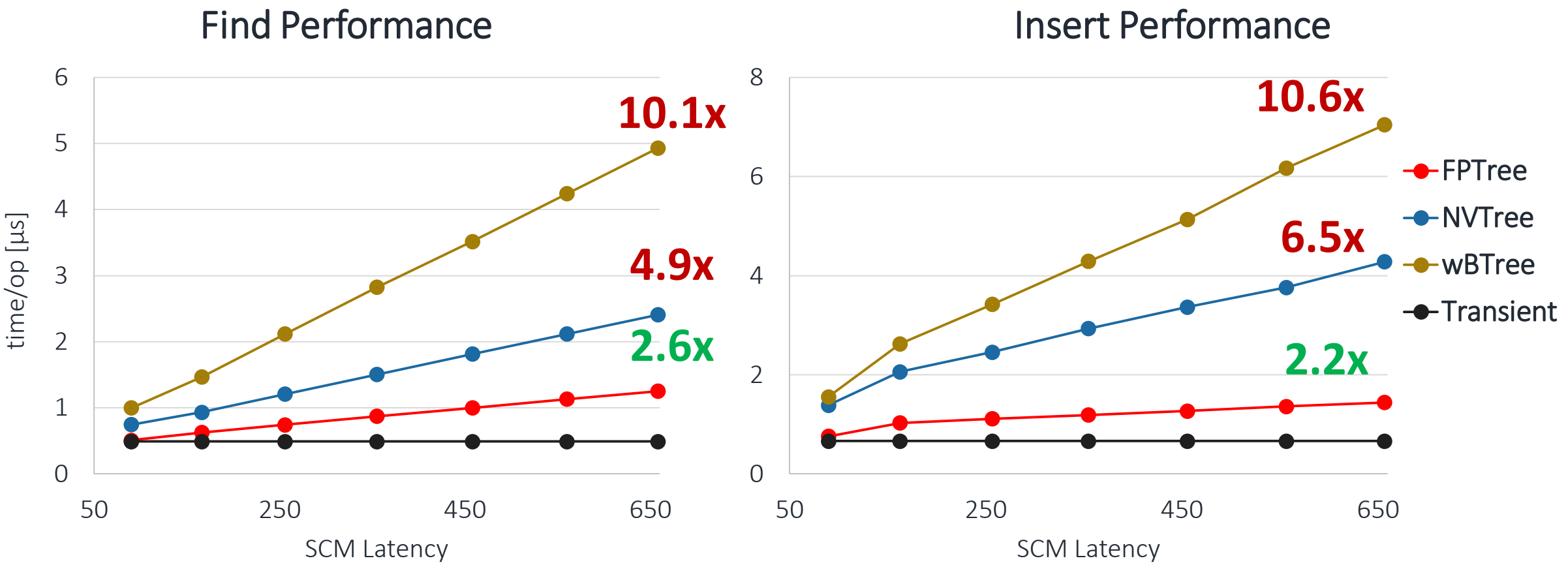
XEND

XEND

XBEGIN

Selective Concurrency solves the incompatibility of HTM and SCM

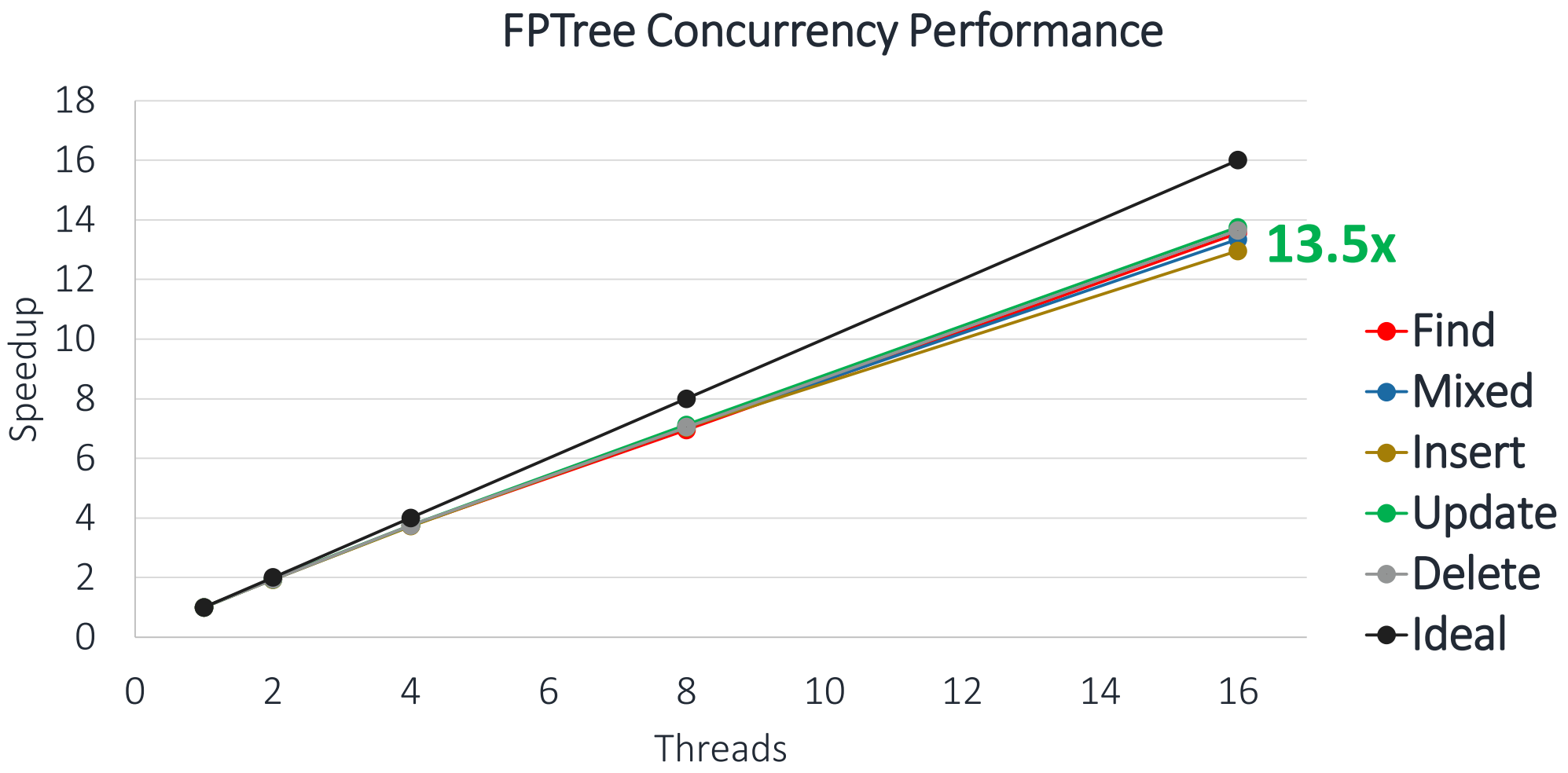
FPTree Performance Evaluation: Single-Threaded



The FPTree is competitive with a Transient B+-Tree

Only ~3% of data in DRAM

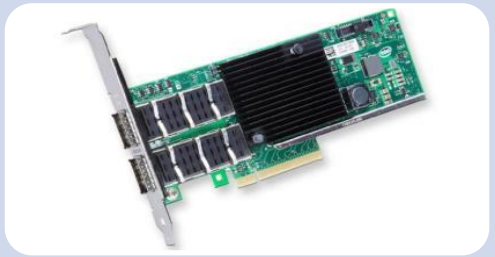
FPTree Performance Evaluation: Multi-Threaded

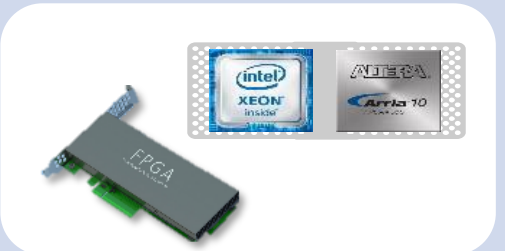


The FPTree scales nearly linearly

Outlook

Extended Hardware ingredients for Optimizations





Intel QuickAssist Technology:

Hardware acceleration for high-compute workloads (crypto, compression)

NIC HW acceleration:

Intelligent or Programmable NICs

FPGA acceleration:

Processor companion chip or as a PCIe add-on board to accelerate platform or workload functions

Summary

- SIMD - Novel use-cases for databases
- TSX – fine-grained locking with coarse-grained implementation
- Storage-Class Memory – Merging storage and memory
- More exciting topics are ahead

Innovative approaches are possible when hardware and software are changed at the same time

Thank you.

Contact information:

Ismail Oukid
Ismail.Oukid@sap.com

We are hiring!

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

Contact: students-hana@sap.com

* 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.