

Next-Gen Hardware for Data Management

More a Blessing than a Curse?

Wolfgang Lehner – EDBT 2015, Brussels

Modern Hardware – All over the place...



Dresden Database
Systems Group

Transaction Logging Unleashed with NVRAM*

Tianzheng Wang Ryan Johnson
University of Toronto
{tzwang, ryan.johnson}@cs.toronto.edu

**Utilization Wall:
Dark Silicon's Effect on Multicore Scaling**

Spectrum of tradeoffs between # of cores and frequency

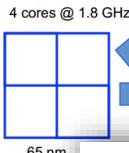
Example:
65 nm → 32 nm (S = 2)

2x4 cores @ 1.8 GHz
(8 cores dark, 8 dim)

PostgreSQL uses first buffered commit to ensure parallel hardware, becoming a bottleneck even with hyper-threading because the 45% of the CPU

SGI Scales Up HANA On UV NUMA System

June 3, 2014 by Timothy Prickett Morgan



Staring into the Abyss: An Evaluation of Concurrency Control with One Thousand Cores

Xiangyu Yu
MIT CSAIL
xyu@csail.mit.edu

George Bezerra
MIT CSAIL
gbezerra@csail.mit.edu

Andrew Pavlo
Carnegie Mellon University
pavlo@cs.cmu.edu

Srinivas Devadas
MIT CSAIL
devadas@csail.mit.edu

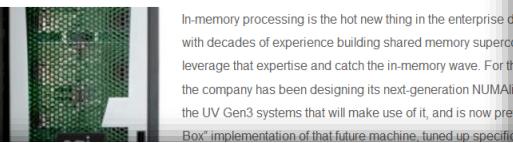
Michael Stonebraker
MIT CSAIL
stonebraker@csail.mit.edu

ABSTRACT

Computer architectures are moving towards an era dominated by many-core machines with dozens or even hundreds of cores on a single chip. This unprecedented level of on-chip parallelism introduces a new dimension of challenge for database management systems (DBMSs) that were not designed for it, particularly as the number of cores increases, the problem of concurrency control becomes extremely challenging. With hundreds of threads running in parallel, the complexity of coordinating competing accesses to data will likely diminish the gains from increased core counts.

In this paper, we take a peek at this dire future and examine what happens with transaction processing at one thousand cores. Rather than looking at all possible scalability challenges, we limit our scope to concurrency control. With hundreds of threads running in parallel, the complexity of coordinating competing accesses to data will become a major bottleneck to scalability, and will likely dwarf the complexity of other DBMS components.

To better understand just how unprepared current DBMSs are for future CPU architectures, we performed an evaluation of concurrency control for on-line transaction processing (OLTP) workloads on many-core chips. We implemented seven concurrency control algorithms on a main-memory DBMS and using computer simulation



Data-Oriented Transaction Execution

Ippokratis Pandis^{1,2} Ryan Johnson^{1,2} Nikos Hardavella¹
ippandis@ece.cmu.edu ryanjohn@ece.cmu.edu nikos@northwestern.edu

¹Carnegie Mellon University Pittsburgh, PA, USA ²École Polytechnique Fédérale de Lausanne Lausanne, VD, Switzerland

ABSTRACT

While hardware technology has undergone major advancements over the past decade, transaction processing systems have remained largely unchanged. The number of cores on a chip grows exponentially, following Moore's Law, allowing for an ever-increasing number of transactions to execute in parallel. As the number of concurrently-executing transactions increases, contended critical sections become scalability burdens. In typical transaction processing systems the central contention point is often the first contended component and set of locks.

In this paper, we identify the contention assignment policy as the primary bottleneck, we design DORA, a system that partitions transactions to smaller actions and assigns a lock to each action in which data each action is about to access. This allows each thread to mostly access threads that are not interacting with the contention manager. Built on top of a conventional transaction manager, DORA attains all the ACID properties. Evaluation of DORA on a multi-core system shows that DORA attains up to 4.8x higher throughput than Oracle's storage engine when running a variety of OLTP workloads.

high bandwidth, from Intel into a shared memory

chip equipped with specialized domains for both. With experts in both the number of cores and exponentially-growing new process generations, as the number of transactions increases exponentially, an upper

bound on the merge overhead is reached.

As the number of cores increases, the

partition. Inserting into the write-optimized structure can be

formed fast if the size of the structure is kept small enough. As an additional benefit, this also ensures that the read performance does not degrade significantly. However, keeping this size small implies merging frequently, which increases the overhead of updates.

In the first part of the paper, we report data analyses of 12 SAP Business Suite customer systems. In the second half, we present an optimized merge process reducing the merge overhead of current systems by a factor of 30. Our linear-time merge algorithm exploits the underlying high compute and bandwidth resources of modern multi-core CPUs with architecture-aware optimizations and efficient parallelization. This enables compressed in-memory column stores to handle the transactional update rate required by enterprise applications, while keeping properties of read-optimized databases for analytic-style queries.

LLAMA: A Cache/Storage Subsystem for Modern Hardware

Justin Levandoski
Microsoft Research
One Microsoft Way
Redmond, WA 98052
justinle@microsoft.com

David Lomet
Microsoft Research
One Microsoft Way
Redmond, WA 98052
lomet@microsoft.com

Sudipta Sengupta
Microsoft Research
One Microsoft Way
Redmond, WA 98052
sudipta@microsoft.com

ABSTRACT

LLAMA is a subsystem designed for new hardware environments that supports an API for page-oriented access methods, providing both cache and storage management. Caching (CL) and storage (SL) layers use a common mapping table that separates a page's logical and physical location. CL supports data updates and management updates (e.g., for index re-organization) via latch-free techniques. SL uses the same mapping table to cope with page location changes produced by log structuring on every page flush. To demonstrate LLAMA's suitability, we tailored our latch-free B-tree implementation to use LLAMA. The B-tree is a B-tree style index. Layered over LLAMA, it has higher performance and scalability using real workloads compared with BerkeleyDB's B-tree, which is known for good performance.

1 INTRODUCTION

1.1 Modern Architectures

Modern computer platforms have changed sufficiently that it is

We believe there are fundamental problems posed by current hardware that impact all access methods: B-trees, hashing, multithreaded, temporal, etc. Further, these problems can be solved with general mechanisms applicable to most access methods.

1. Good processor utilization and scaling with multi-core processors via latch-free techniques.
2. Good performance with multi-level cache based memory systems via delta updating that reduces cache invalidations.
3. Write limited storage in two senses: (1) limited performance of random writes; (2) flash write limits; via log structuring.

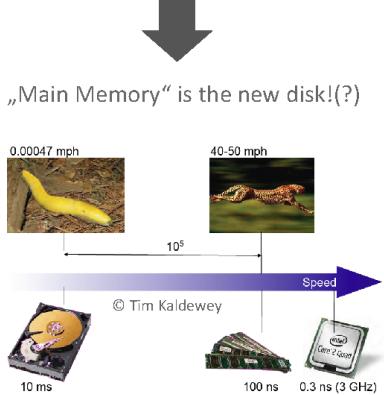
The Bw-tree [16], an index resembling B-trees [4, 7], is an example of a DC or key-value store that exploits these techniques. Indeed, it is an instance of a paradigm for how to achieve latch-free and log structuring more generally. In this paper, we describe a new architecture where the latch-free and log-structure techniques of the Bw-tree are implemented in a cache/storage subsystem capable of supporting multiple access methods, in the same way that a traditional cache/storage subsystem deals with latched access to fixed-size pages that are written back to disks as in-place updates.

> A Look at Hardware Trends

Database Technology Group

Increasing Main Memory Capacity*

„Main Memory“ is the new disk!(?)



0.00047 mph
40-50 mph

10 ms 100 ns 0.3 ns (3 GHz)

© Tim Kaldewey

Speed

10⁵

Xeon E7 20 HW Threads

I7-2600 6 HW cores

Intel Phi

Increasing Number of Cores

- CPU/GPU, hybrids
- FPGA (Field Programmable Gate Array)

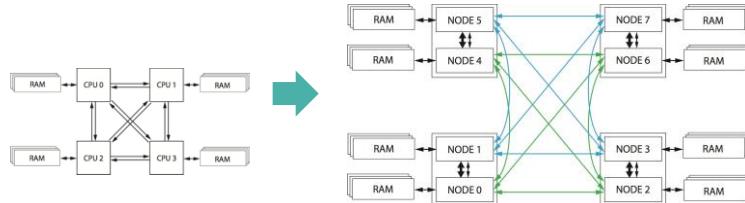
„Parallelism“ is the name of the game!

* stable RAM will be an additional game changer

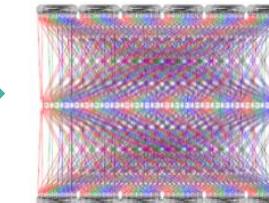
A Look at Hardware Trends - 2015

System Level ↑

Extreme NUMA Effects



1



„Main Memory“ is the new disk!(?)



„Parallelism“ is the name of the game!

Component Level ↓

Storage-Class Memory



3

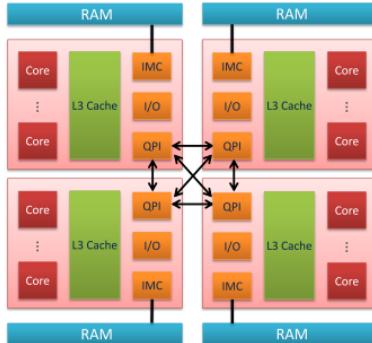
Application-Specific Instruction Sets



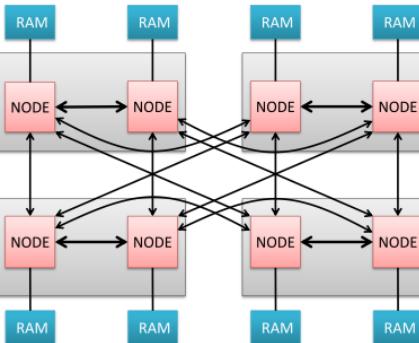
2

Extreme NUMA Awareness

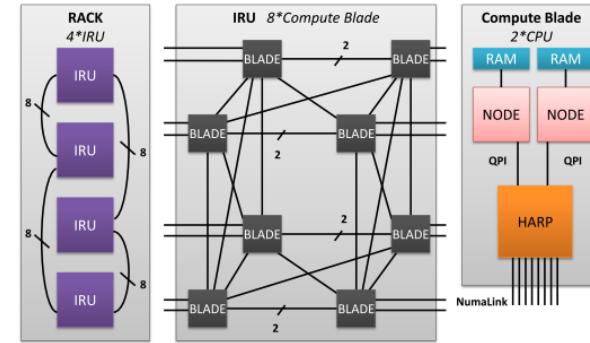
NUMA Awareness



(a) Intel Machine (Detailed).



(b) AMD Machine (Topology View).



(c) SGI Machine (Topology View).

Intel machine			AMD machine			SGI machine		
distance	bandwidth (GB/s)	latency (ns)	distance (link width)	bandwidth (GB/s)	latency (ns)	distance	bandwidth (GB/s)	latency (ns)
local	26.7	129	local	16.4	85	local	36.2	81
1 hop QPI	10.7	193	1 hop HT (full link)	5.8	136	2nd processor	9.5	400
			1 hop HT (split,single)	4.2	152	1 hop NUMALink	7.5	505 - 515
			1 hop HT (split,dual)	2.9	152	2 hop NUMALink	7.5	625 - 635
			2 hop HT (split,single)	3.7	196	3 hop NUMALink	7.1	745 - 755
			2 hop HT (split,dual)	1.8	196	4 hop NUMALink	6.5	870

Which Architecture ?

Pros & Cons

Challenges

Data-Oriented Transaction Execution

Ippokratis Pandis^{1,2}

ipandis@ece.cmu.edu

Ryan Johnson^{1,2}

ryanjohn@ece.cmu.edu

Nikos Hardavellas³

nikos@northwestern.edu

Anastasia Ailamaki^{2,1}

natassa@epfl.ch

¹Carnegie Mellon University
Pittsburgh, PA, USA

²École Polytechnique Fédérale de Lausanne
Lausanne, VD, Switzerland

³Northwestern University
Evanston, IL, USA

ABSTRACT

While hardware technology has undergone major advancements over the past decade, transaction processing systems have remained largely unchanged. The number of cores on a chip grows exponentially, following Moore's Law, allowing for an ever-increasing number of transactions to execute in parallel. As the number of concurrently-executing transactions increases, contended critical sections become scalability burdens. In typical transaction processing systems the centralized lock manager is often the first contended component and scalability bottleneck.

In this paper, we identify the conventional thread-to-transaction assignment policy as the primary cause of contention. Then, we design DORA, a system that decomposes each transaction to smaller actions and assigns actions to threads based on which data each action is about to access. DORA's design allows each thread to mostly access thread-local data structures, minimizing interaction with the contention-prone centralized lock manager. Built on top of a conventional storage engine, DORA maintains all the ACID properties. Evaluation of a prototype implementation of DORA on a multicore system demonstrates that DORA attains up to 4.8x higher throughput than a state-of-the-art storage engine when running a variety of synthetic and real-world OLTP workloads.

chip equipped with 8 cores¹, while multicores targeting specialized domains find market viability at even larger scales. With experts in both industry and academia forecasting that the number of cores on a chip will follow Moore's Law, an exponentially-growing number of cores will be available with each new process generation.

As the number of hardware contexts on a chip increases exponentially, an unprecedented number of threads execute concurrently, contending for access to shared resources. Thread-parallel applications, such as online transaction processing (OLTP), running on multicores suffer of increasing delays in heavily-contended critical sections, with detrimental performance effects [14]. To tap the increasing computational power of multicores, the software systems must alleviate such contention bottlenecks and allow performance to scale commensurately with the number of cores.

OLTP is an indispensable operation in most enterprises. In the past decades, transaction processing systems have evolved into sophisticated software systems with codebases measuring in the millions of lines. Several fundamental design principles, however, have remained largely unchanged since their inception. The execution of transaction processing is full of critical sections [14]. Consequently, these systems face significant performance and scalability problems on highly-

hitecture
shared-nothing

OS

DB

...

...

parallel systems

indirection required

final by design

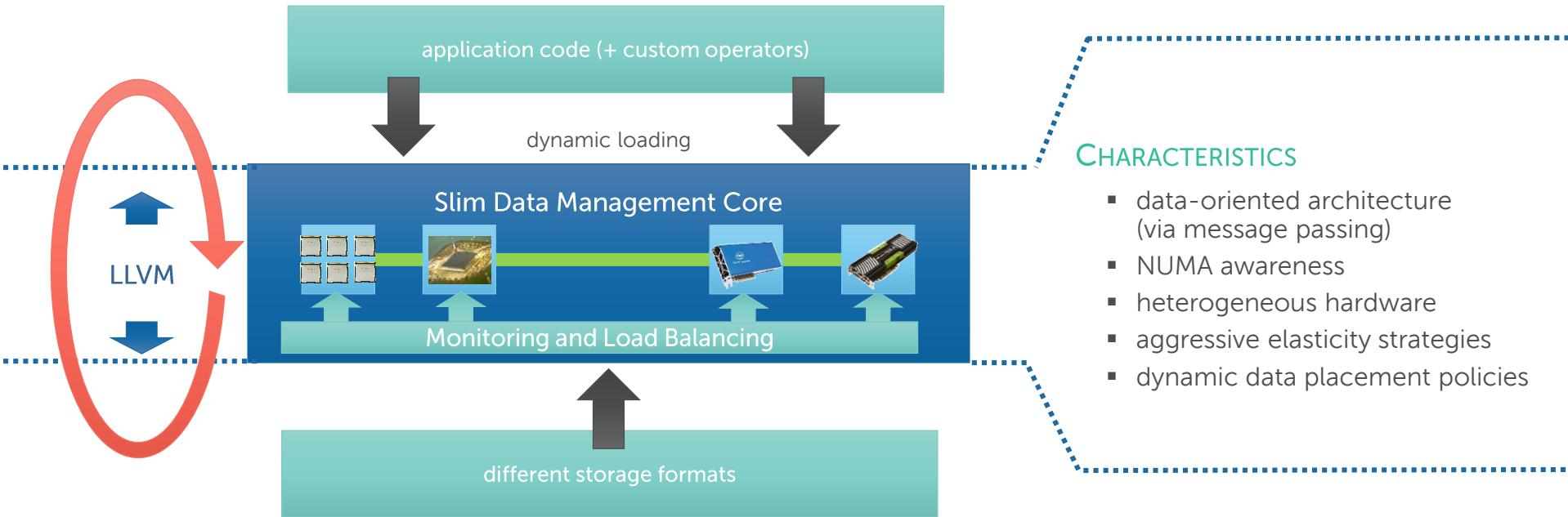
indirection to work
key systems
data-oriented architecture

ERIS Data Management Core



Dresden Database
Systems Group

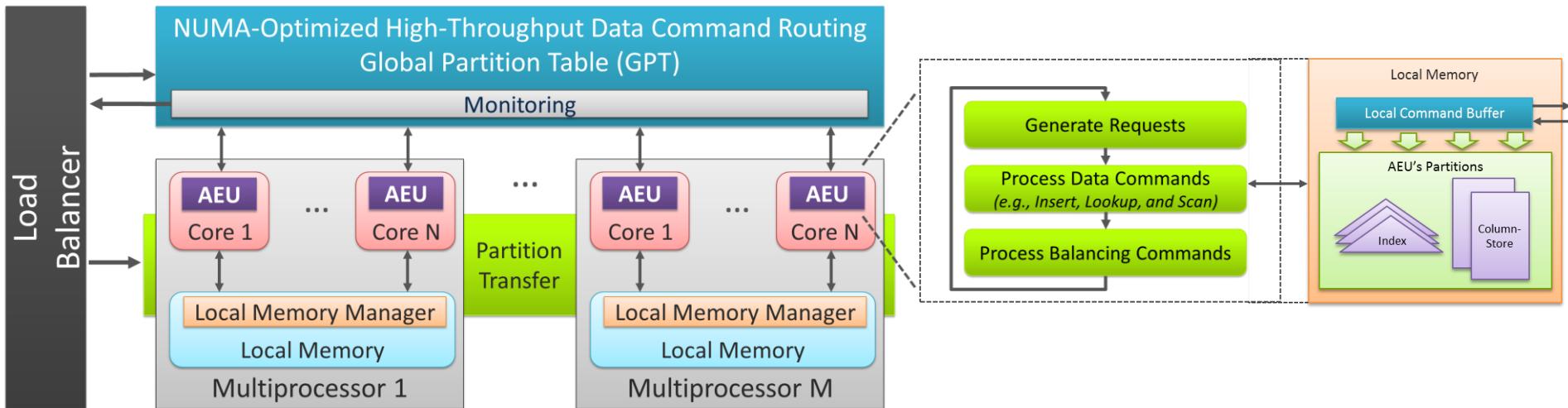
... an academic playground for modern DB techniques



ERIS Overall Architecture

DATA-ORIENTED ARCHITECTURE

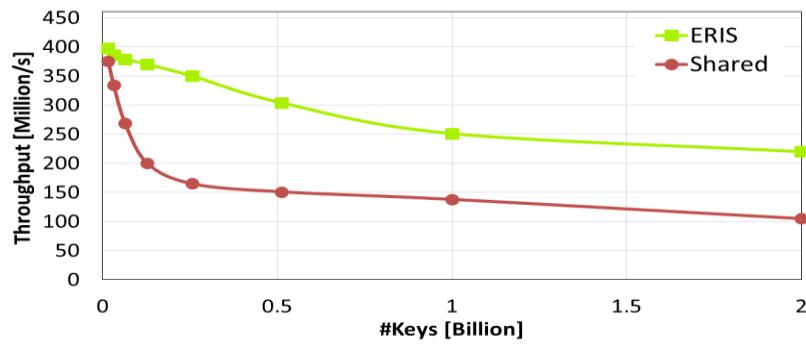
- Follows MVCC principle
- Distribution based on logical partitioning
- Aggressive re-partitioning using copy as well as link strategies



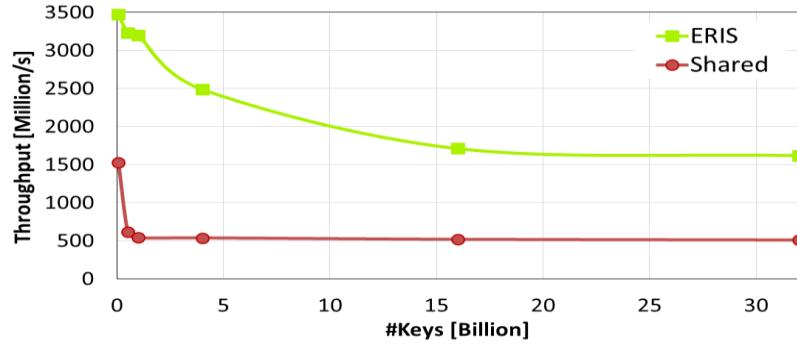
Evaluation: Some MicroBenchmarking

LOOKUP/UPSERT THROUGHPUT DEPENDING ON INDEX SIZE

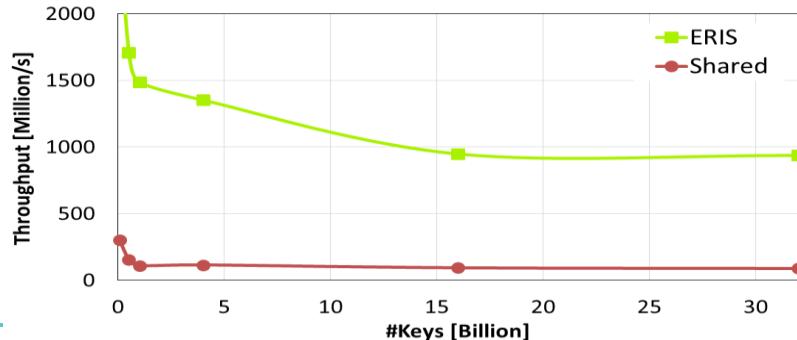
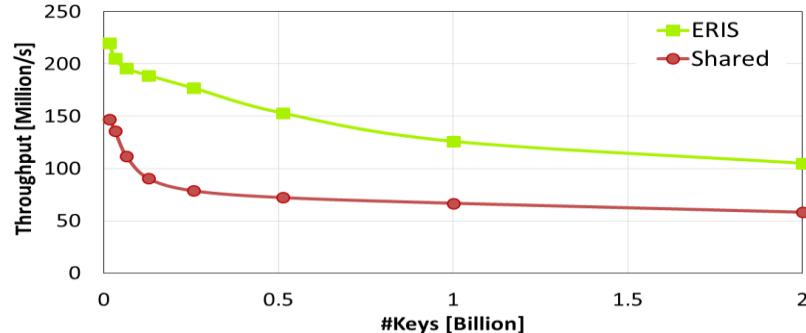
AMD Machine



SGI Machine



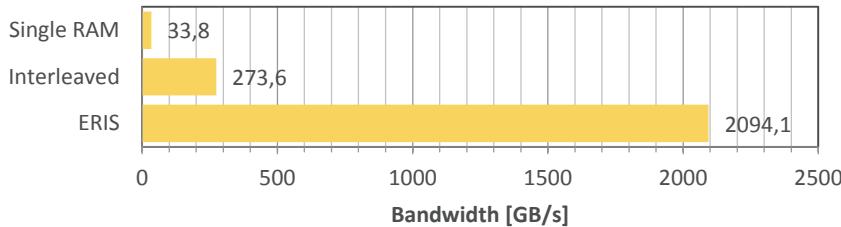
Upsert



Evaluation: Scan Throughput

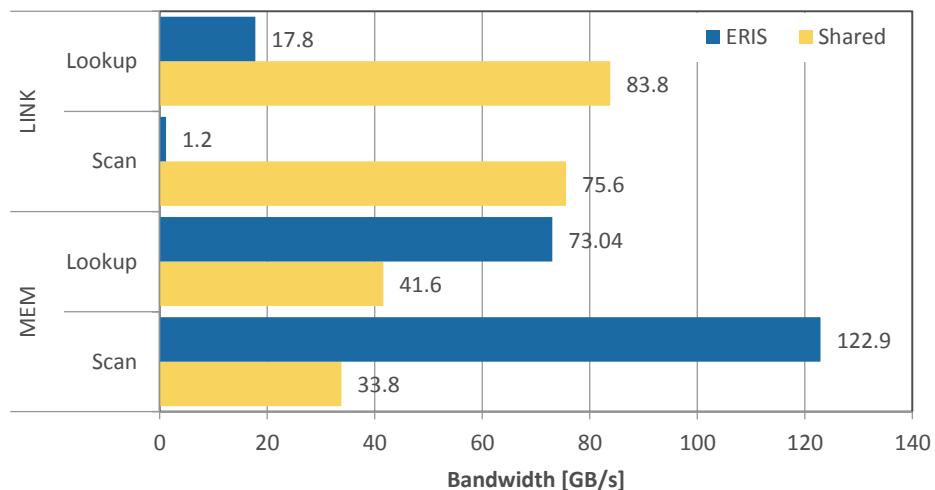
SCAN PERFORMANCE

- SGI Machine
- 488 cores – parallel scan
- 8 billion entries in the column store



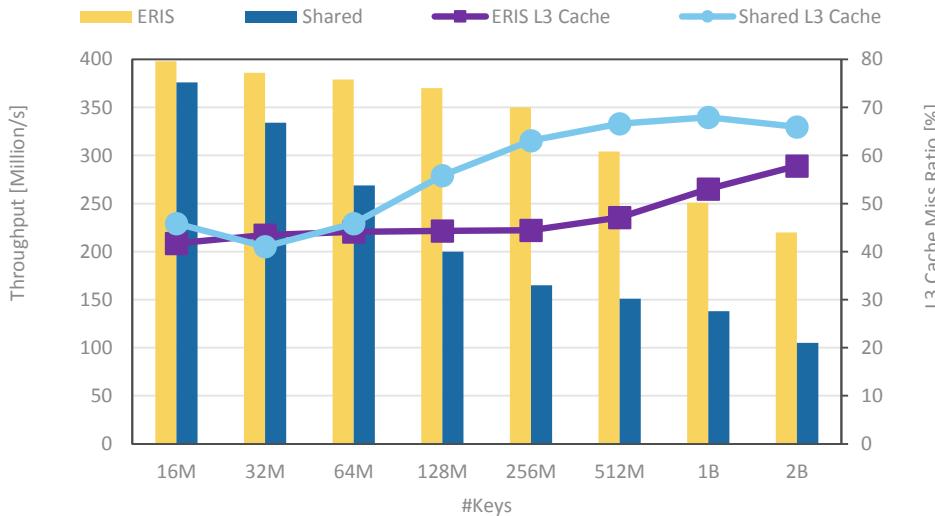
LINK AND MEMORY CONTROLLER ACTIVITY

- AMD Machine
- Scan: 8B Keys
- Lookup: 1B Keys



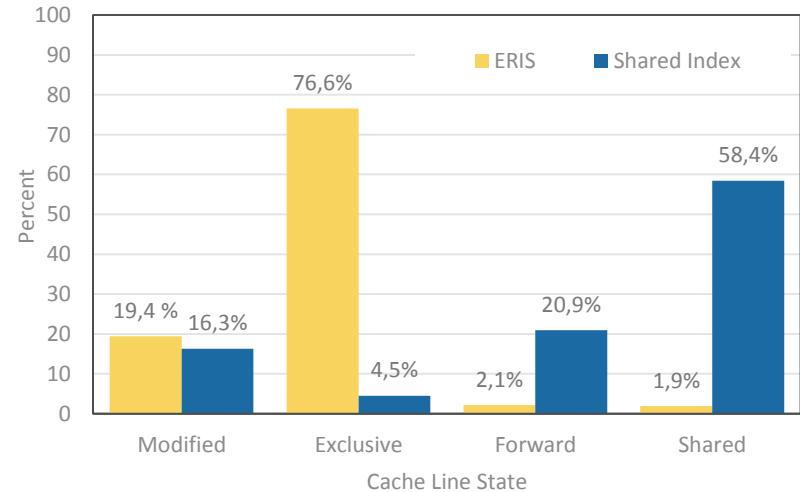
Evaluation: L3 Cache Usage

L3 CACHE USAGE – INDEX LOOKUP



L3 CACHE LINE STATE – INDEX LOOKUP

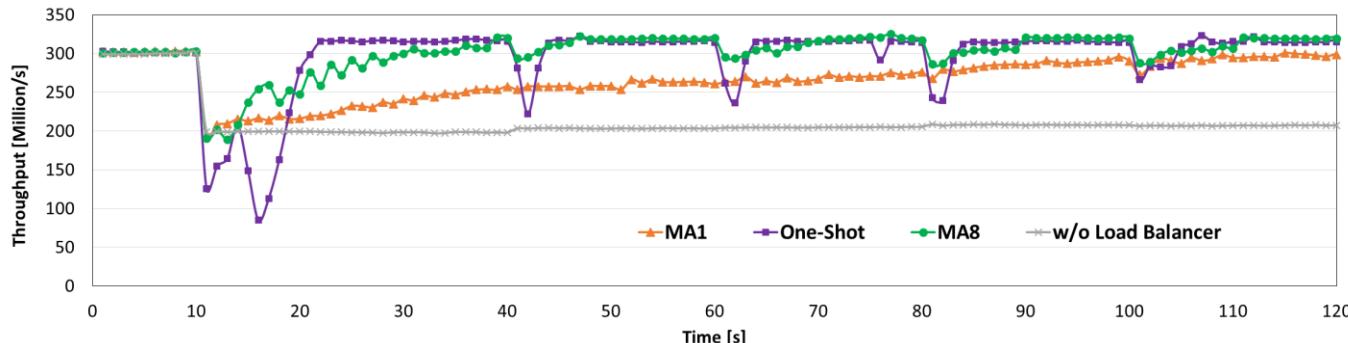
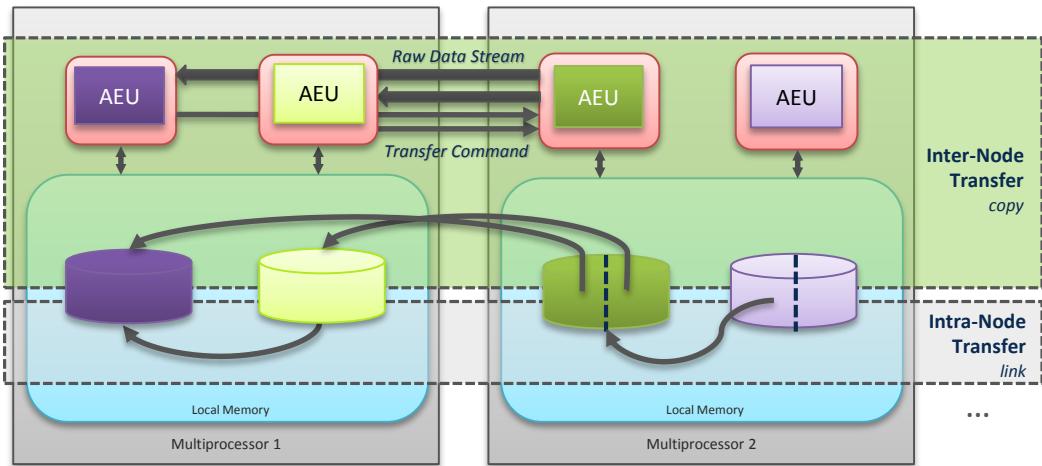
- Percentage of all hits
- 1B keys



Key Component: Load Balancer

BALANCING STRATEGIES

- Copy Strategy:
copy data within NUMA systems
between different sockets
- Link Strategy:
delegate pointer, delay data movement



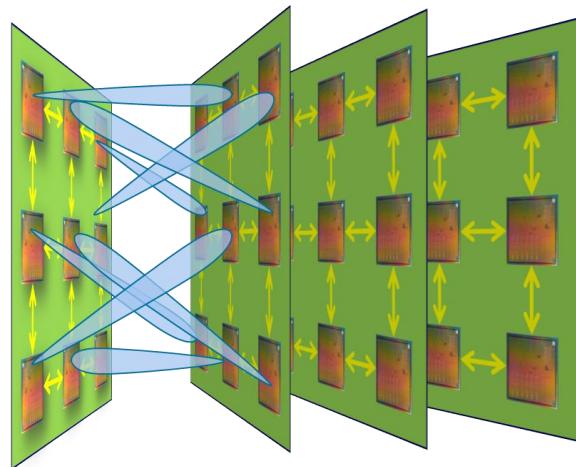
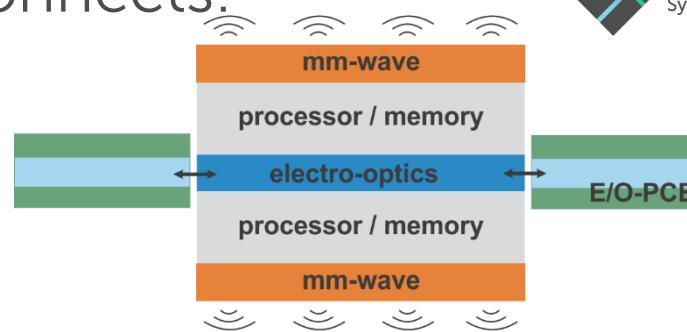
What's Next? Wireless Interconnects!

ADVANCES IN WIRED INTERCONNECTS

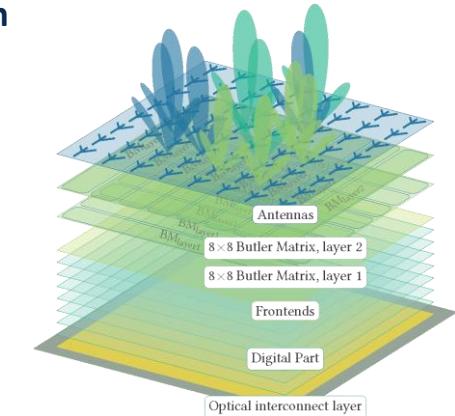
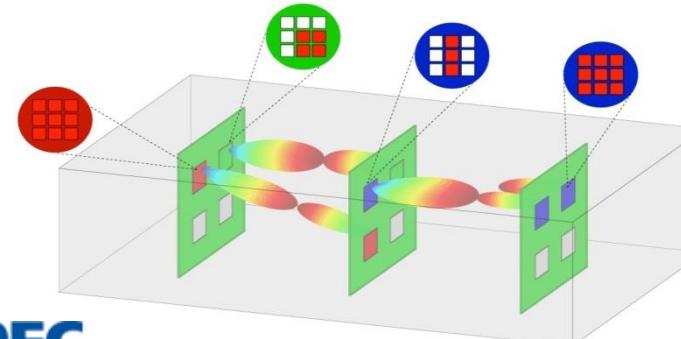
- Optical interconnects, ...

...BUT: THE BREAKTHROUGH MAY COME WITH

- High-speed, short-range wireless links

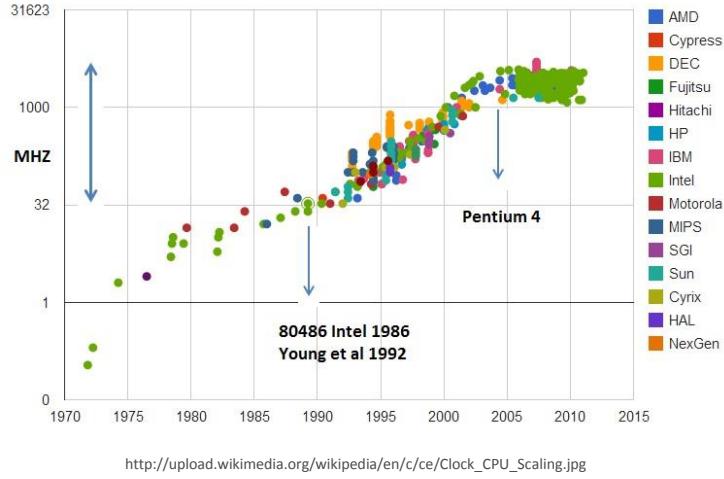


Antennas and Wave Propagation
for Adaptive Wireless Backplane Communication



Database-specific Instructions

xPU Developments and Consequences



Hadi Esmaeilzadeh¹ Emily Blem² Renée St. Amant³ Karthikeyan Sankaralingam⁴ Doug Burger⁵

¹University of Washington ²University of Wisconsin-Madison

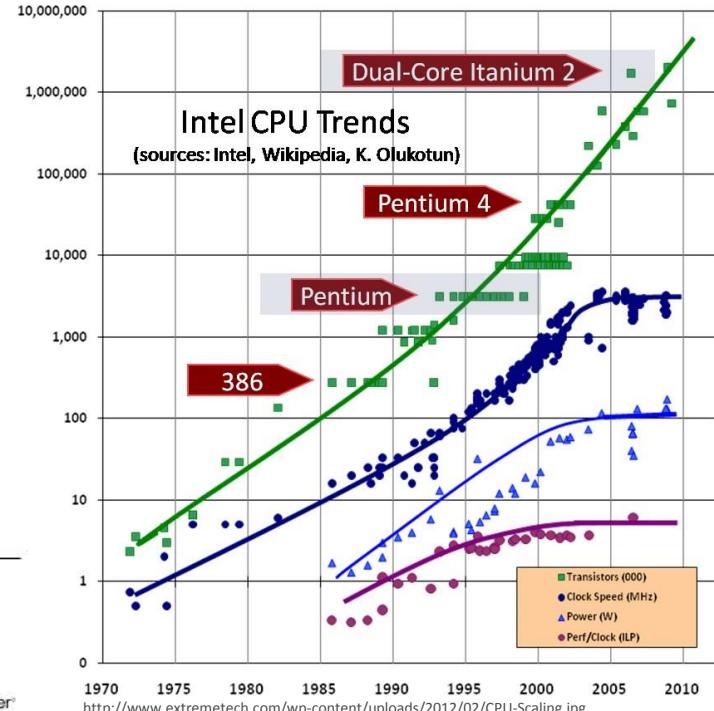
³The University of Texas at Austin ⁴Microsoft Research

hadiane@cs.washington.edu blem@cs.wisc.edu stamant@cs.utexas.edu karu@wisc.edu dburger@microsoft.com

ABSTRACT

Since 2005, processor designers have increased core counts to exploit Moore's Law scaling, rather than focusing on single-core per-

ture, and compiler advances. Moore's Law, coupled with Dennard scaling [1], has resulted in commensurate exponential performance increases. The recent shift to multicore designs has aimed to in-



Motivation of „DB Processor“

TODAY'S DATABASE SERVERS

- Fat cores (area & power)
- Few HW adaptions
- CMOS scaling



DATABASE MACHINES (ANCIENT)

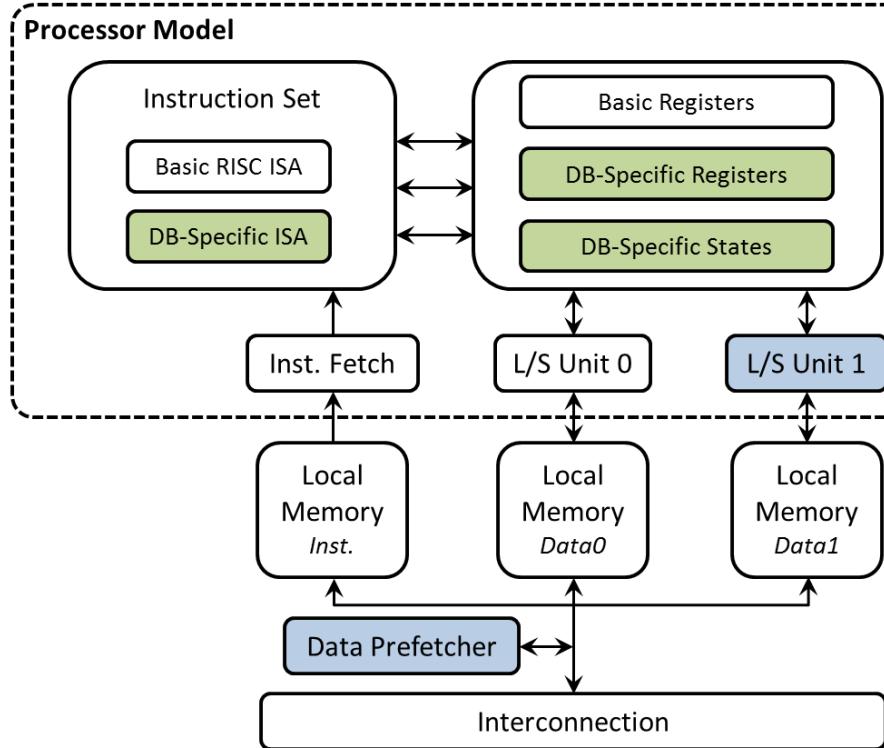
- Processors build from scratch
- Long development cycles
- High development costs



OUR APPROACH

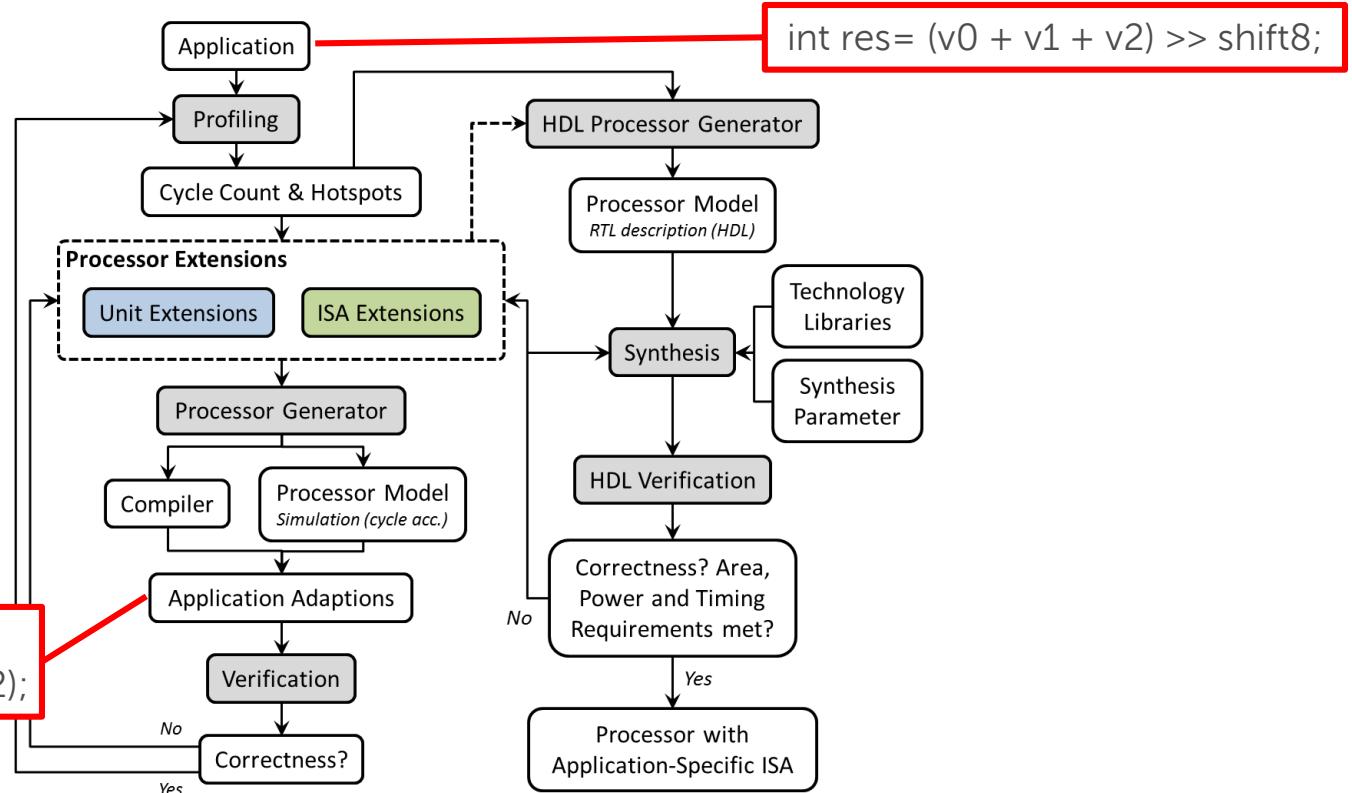
- HW/SW Co-design
- Customizable processor
- Application-specific ISA extensions
- Tool flow & short HW development cycles

Customizable Processor Model



Basic Core:
Tensilica LX4

Tool Flow



Application Scenario

SET OPERATIONS

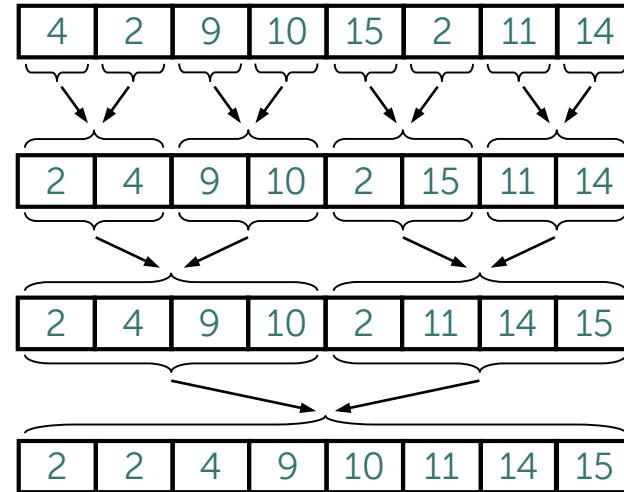
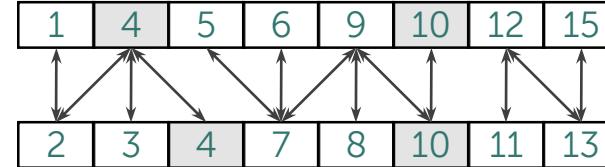
- Intersection
- Difference
- Union

SORTING

- Merge Sort

HASH OPERATIONS

- Integer Hashing
- String Hashing
- Hash Table Management



Sorted-Set Intersection

```

int intersect (int* A, int* B, int l_a, int l_b, int* C) {
    int pos_a = 0, pos_b = 0, pos_c = 0;
    while( pos_a < l_a && pos_b < l_b ) {
        if( A[ pos_a ] == B[ pos_b ] ) {
            C[ pos_c++ ] = A[ pos_a ];
            pos_a++;
            pos_b++;
        }
        else if( A[ pos_a ] < B[ pos_b ] )
            pos_a++;
        else
            pos_b++;
    }
    return pos_c;
}

```

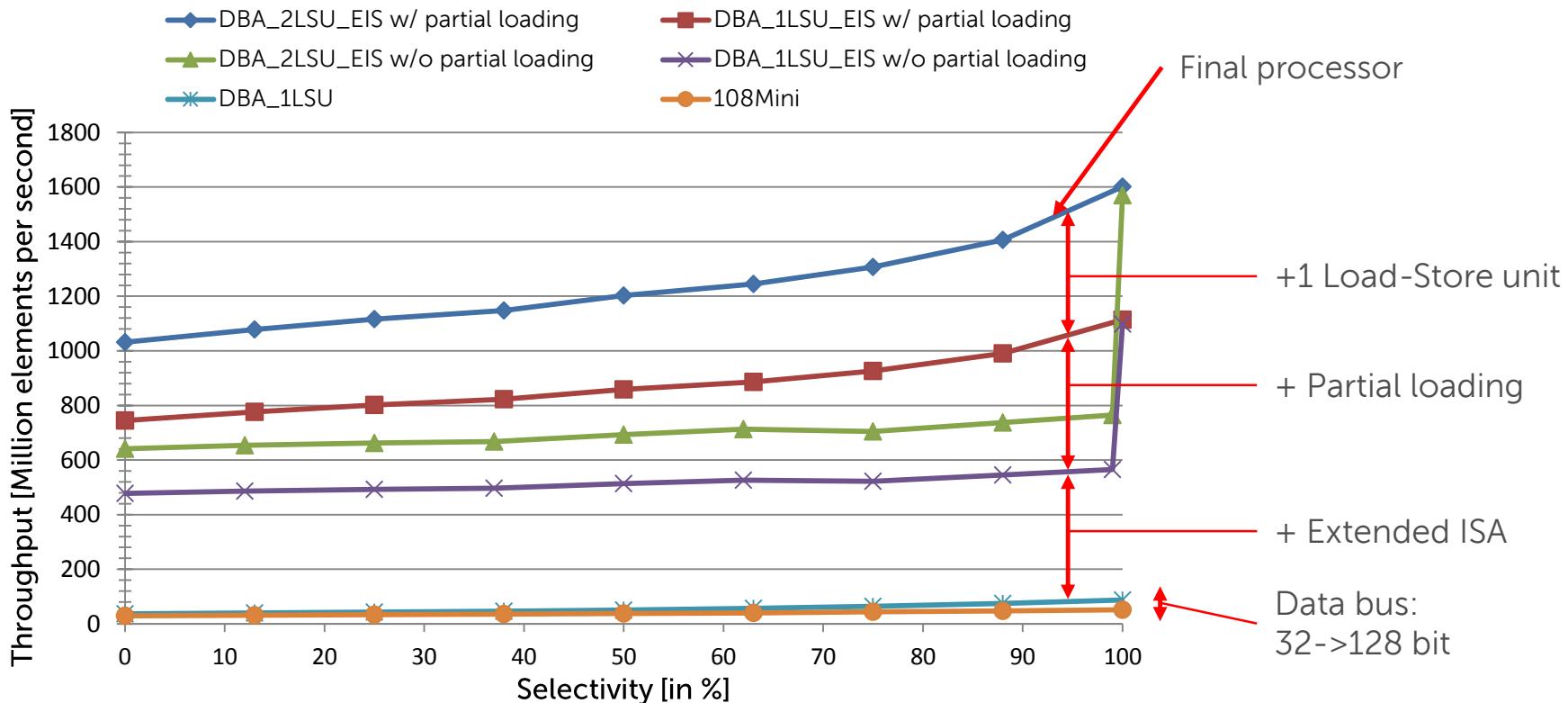


internal states

merged in one inst.

→ + 2x 128 bit data busses + explicit load-store + SIMD + ...

Selectivity: Intersection



Throughput [Mib/s]

Selectivity: 50%

+1 Load-Store unit

Processor	Partial Load–Store	f[MHz]	Intersection	Union	Difference	Merge–Sort
108MINI ¹	-	442	31.3	26.4	35.7	1.7
DBA_1LSU	-	435	50.7	47.7	50.4	3.2
DBA_1LSU_EIS	no	424	513.4	665.0	658.8	29.3
DBA_2LSU_EIS	no	410	693.0	643.0	637.0	28.3
DBA_1LSU_EIS	yes	424	859.0	574.2	859.0	29.3
DBA_2LSU_EIS	yes	410	1203.0	780.4	1192.6	28.3

→ Data bus: 32->128 bit

Final processor

→ DBA_2LSU_EIS vs. 108MINI: 38x 30x 33x 17x

Comparison

SORTED-SET INTERSECTION

	INTEL i7-920	DBA_2LSU_EIS	
Throughput (elements/s)	1,100 mio	1,203 mio	
Clock frequency	2.67 GHz	0.41 GHz	→ 7x improvement
Max. TDP	130 W	0.135 W	→ 963x improvement
Cores/Threads	4/8	1/1	
Feature size	45 nm	65 nm	
Area (logic & memory)	263 mm ²	1.5 mm ²	→ 175x improvement

Timing and Area

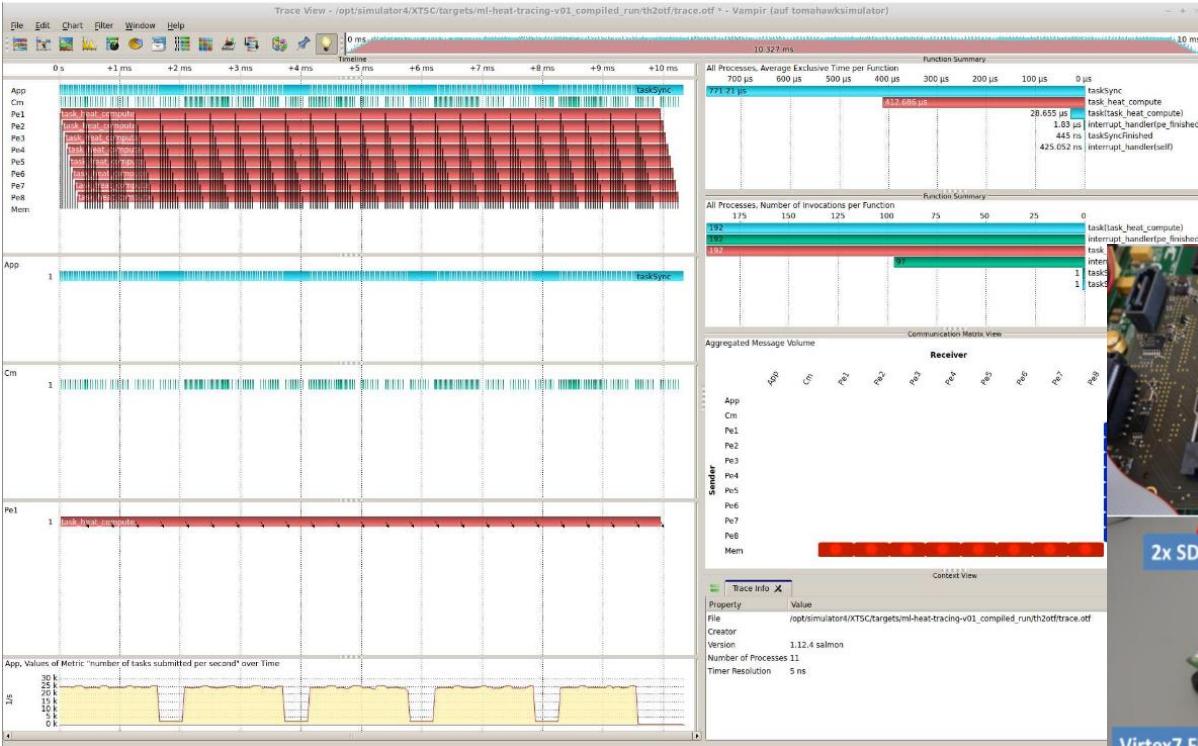
Technology	Processor	$A_{LOGIC} [mm^2]$	$A_{MEM} [mm^2]$	$f_{MAX} [\text{MHz}]$	$P[\text{mW}] @ f_{MAX}$	Final processor
65 nm	108MINI	0.220 ¹	-	442 ¹	27.4 ¹	
	DBA_1LSU	0.177	0.874	435	56.6	
	DBA_2LSU	0.177	0.870	429	57.1	
	DBA_1LSU_EIS	0.523	0.874	424	123.5	
	DBA_2LSU_EIS	0.645	0.870	410	135.1	
28 nm	DBA_2LSU_EIS	0.169	0.232	500	47.0	

¹<http://www.tensilica.com/uploads/pdf/108Mini.pdf>

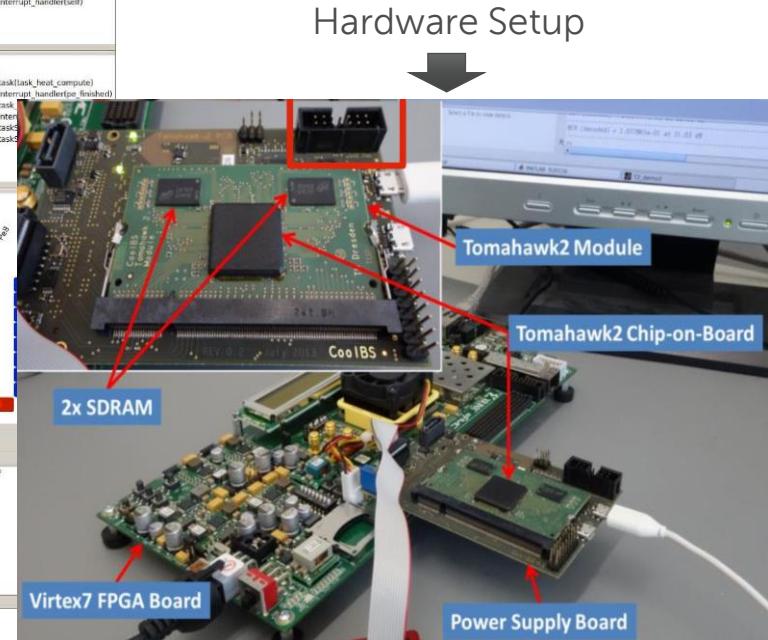
Relative Area Consumption(DBA_2LSU_EIS)

Part	Area[%]
Basic Core	20.5
Decoding/Muxing	14.4
States	14.7
Op: All	11.3
Op: Intersection	6.8
Op: Difference	9.0
Op: Union	17.6
Op: Merge-Sort	5.7
SUM	100

Tomahawk2 Programming

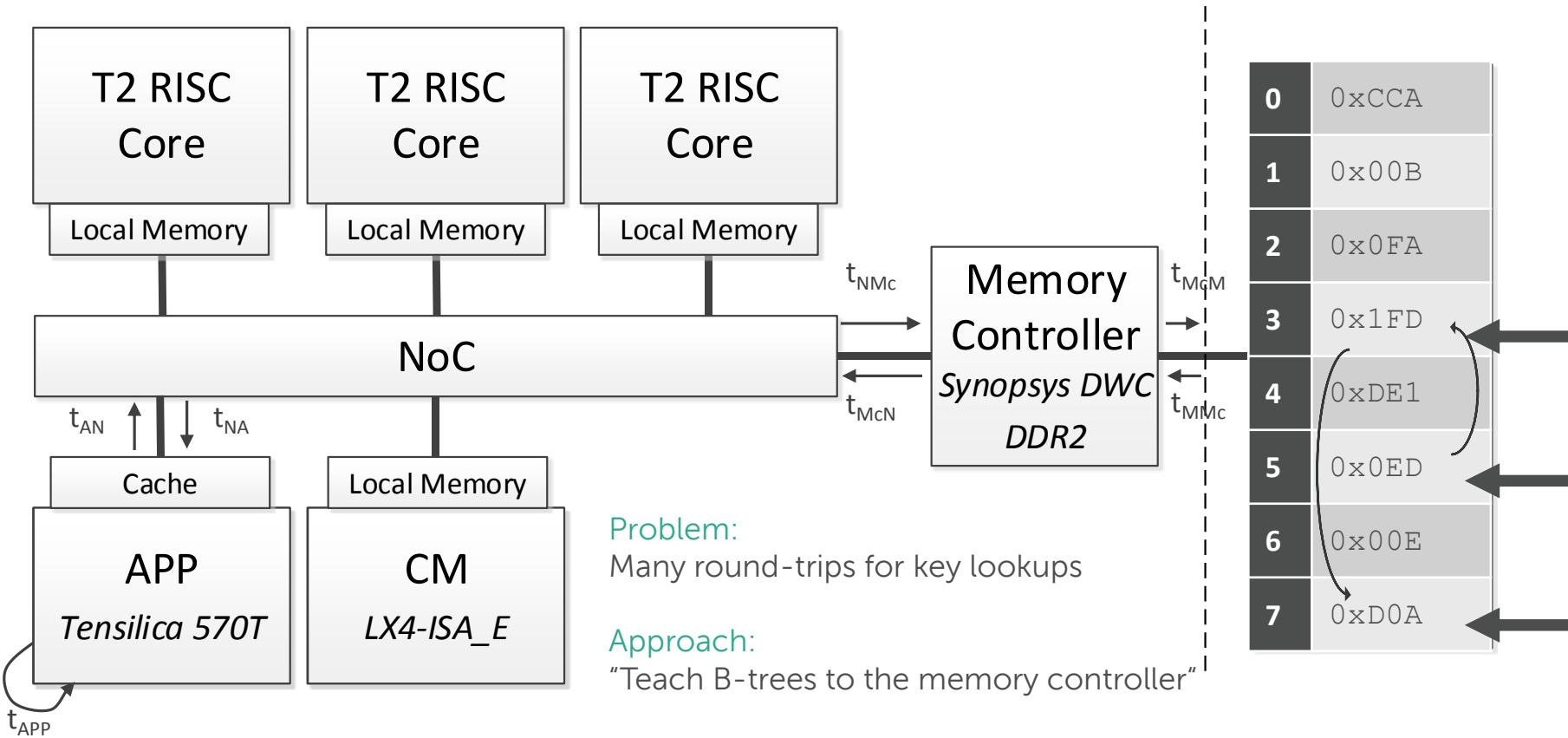


Profiling Software

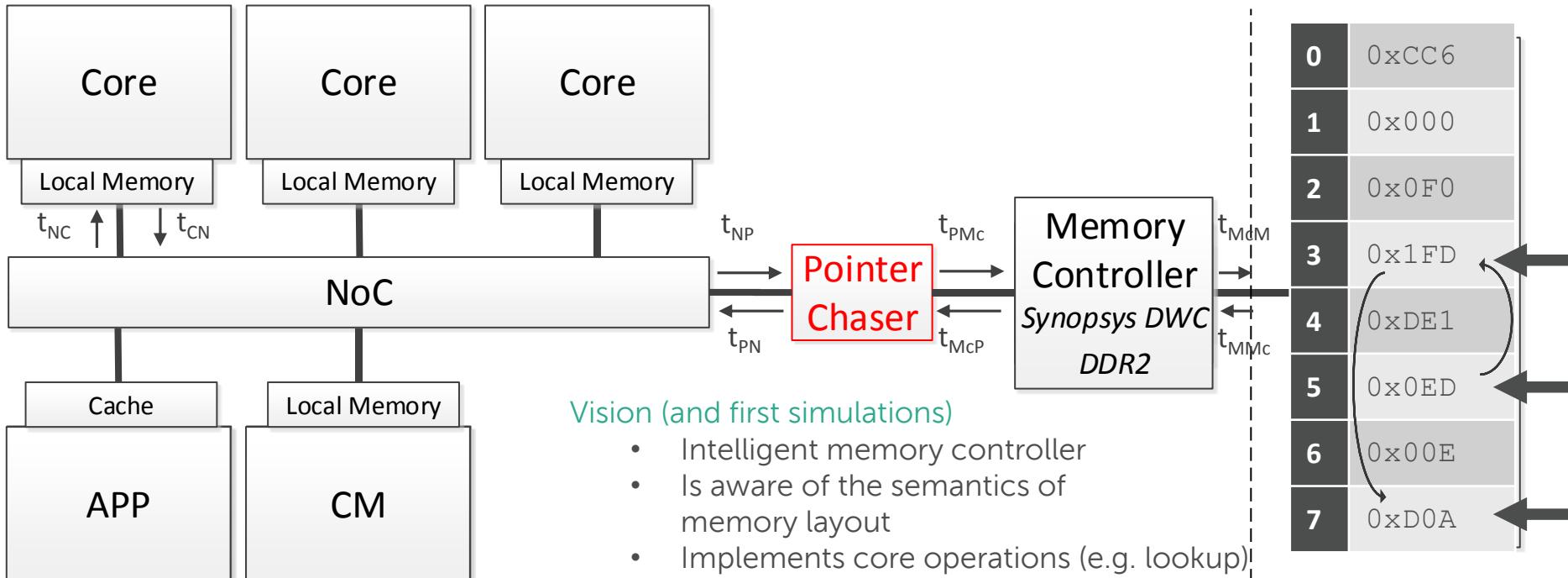


Hardware Setup

What's next? → Pointer Chasing



Basic Principle



Vision (and first simulations)

- Intelligent memory controller
- Is aware of the semantics of memory layout
- Implements core operations (e.g. lookup)

Implementation (no yet in silicon)

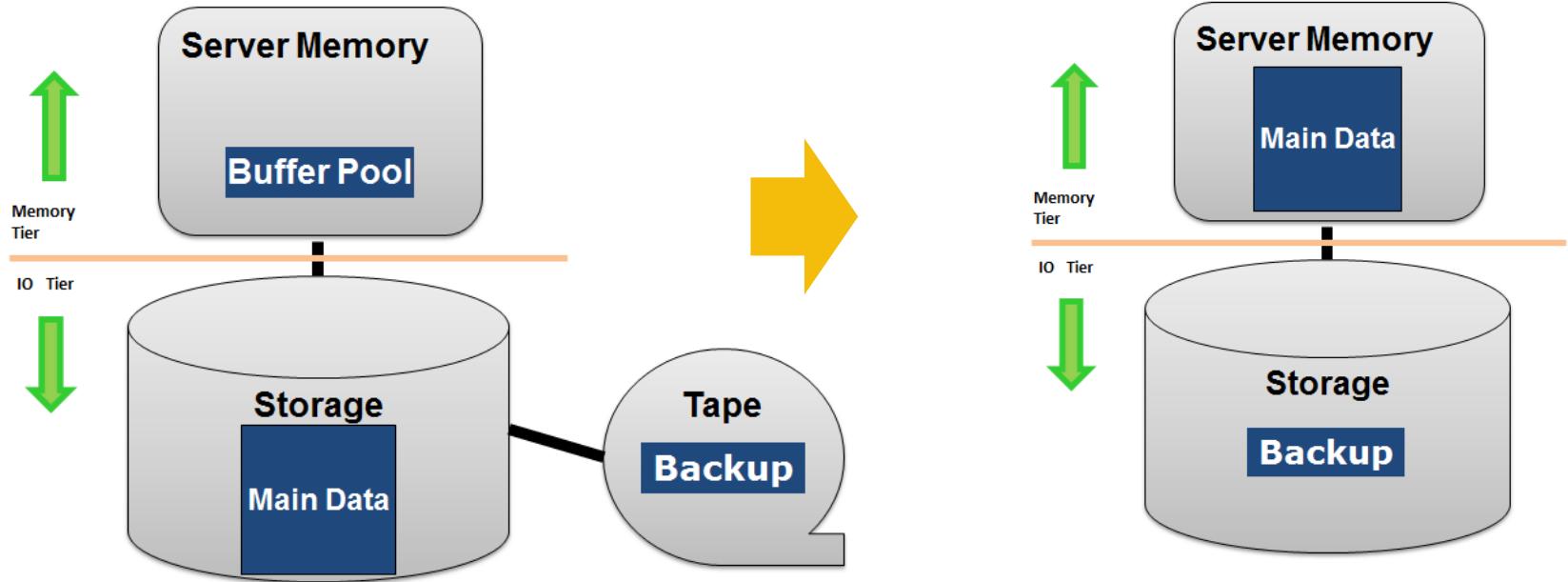
- 0,183mm² PE with 200Mhz

Storage Class Memory

Project „SOFORT“

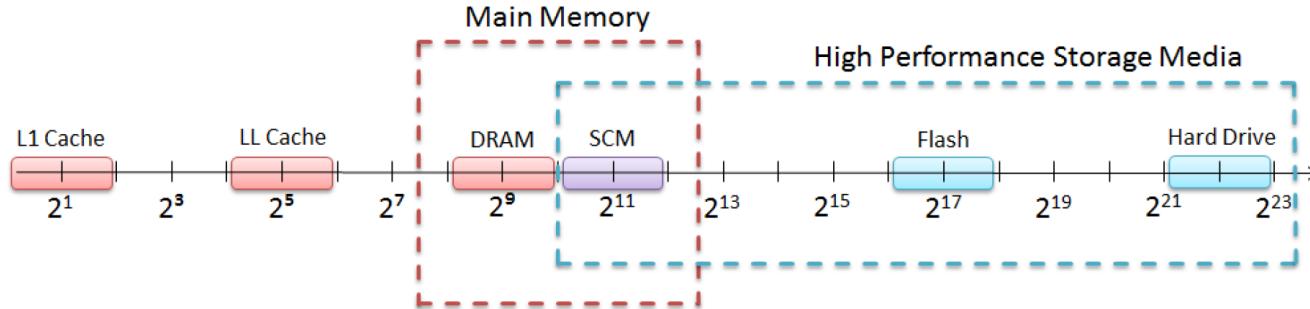


Database Evolution (10 years ago...)



DRAM is hitting costs and power consumption limits!

Storage Class Memory / Non-Volatile RAM



Adapted from: M. K. Qureshi, V. Srinivasan, and J. A. Rivers. Scalable high performance main memory system using phase-change memory technology. In ISCA 2009

EMERGING TECHNOLOGY

- Examples: MRAM(IBM), FRAM (Ramtron), PCM(Samsung)
- Merging Point between storage and memory

ADVANTAGES

- ~4x denser than DRAM
- SCM does not consume energy if not used
- SCM is cheaper, persistent, byte-addressable

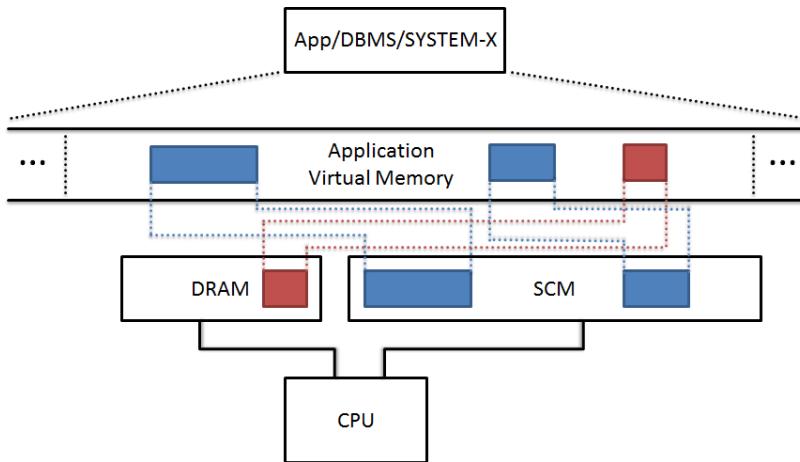
DRAWBACKS

- Number of writes is limited (life expectancy 2-10 years)
- SCM has higher latency than DRAM
 - Read latency ~2x slower than DRAM
 - Write latency ~10x slower than DRAM

SCM Access // File Level

HYBRID STORAGE MODEL

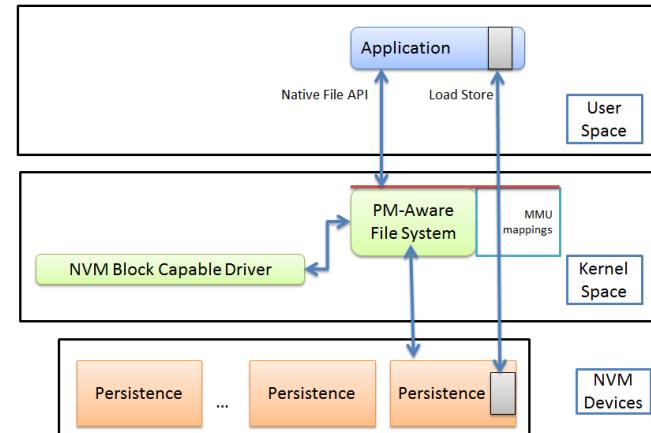
- Application may distinguish between
 - Traditional memory
 - Persistent memory blocks
- Files names are used as identification handle

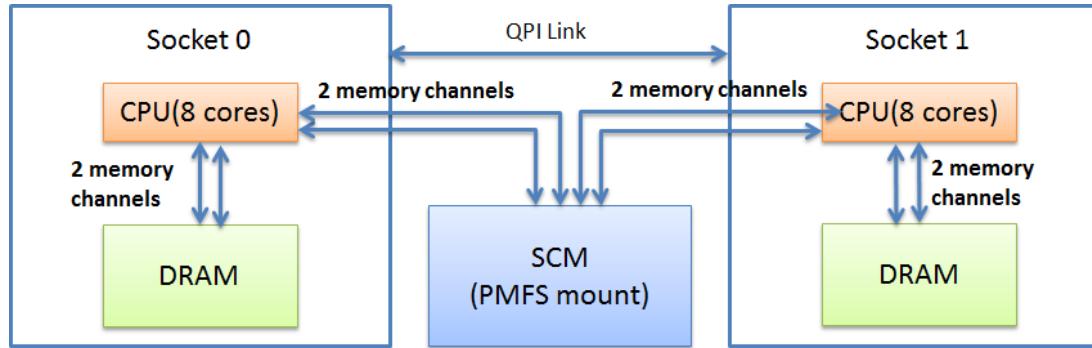


NVM PM MODE – FILE LEVEL (PMFS)

- Shows how to get persistent memory to the application level
- Requires a persistent memory-aware file system
- Direct access to regions of persistent memory inside the application

NVM PM Mode – File Level





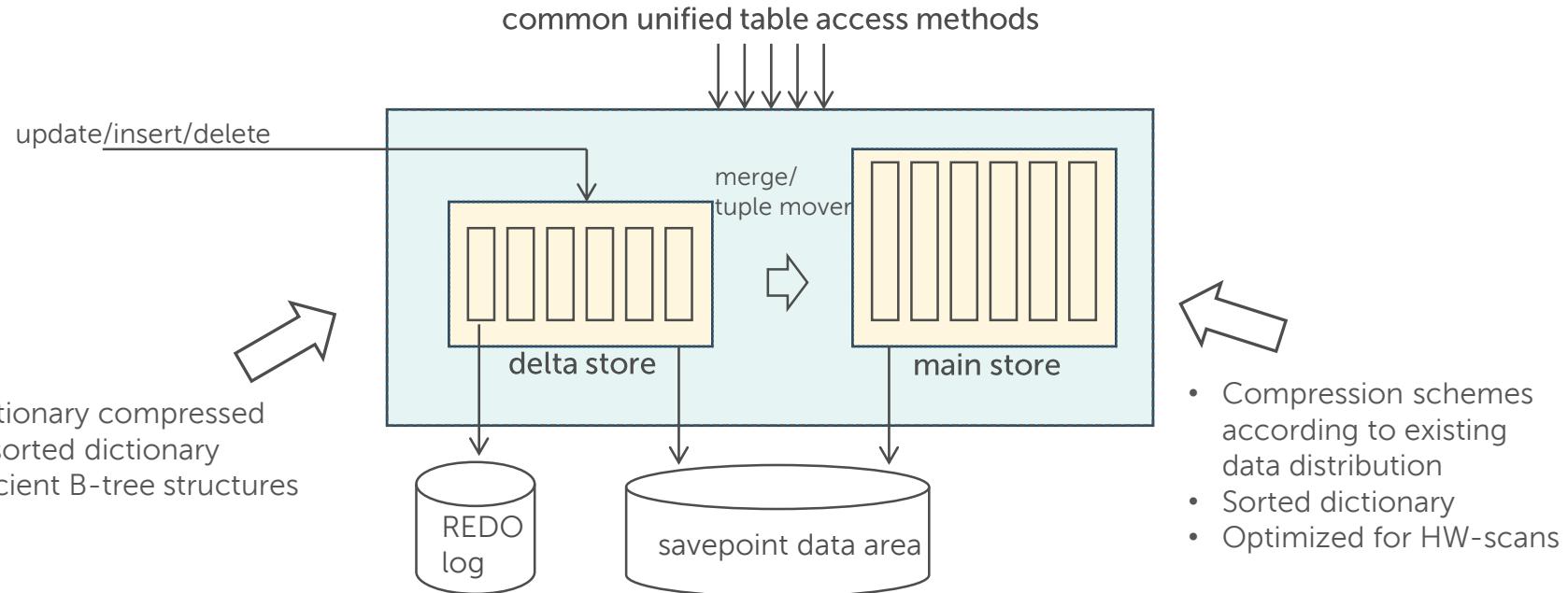
EXPERIMENTAL SETUP

- hybrid architecture with both DRAM and SCM, software can allocate in SCM through a separate memory interface
- Special BIOS allows simulation of different latency by the means of a microcode patch
 - Conceptually, local DRAM is attached to each socket, whereas the SCM is attached to both sockets emulating a uniform SCM region by interleaving memory on a cache-line level.
- SCM latency is fixed to 200ns by default. DRAM latency is 90ns.

Hybrid Storage Architecture for Column Stores

USE OF COMPRESSION IMPLIES TWO STORES

- Write optimized store (WOS)
- Read optimized (compressed) store (ROS)

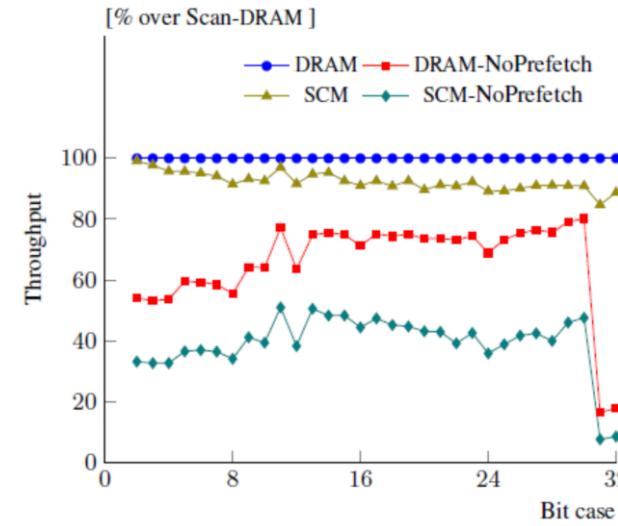
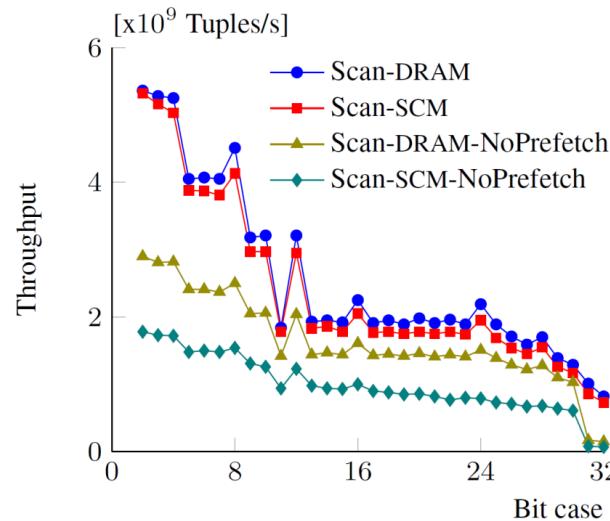


NVRAM for ROS-Structure

MICRO EXPERIMENTS: SIMD SCAN PERFORMANCE ON DRAM AND SCM

- With prefetching: average penalty for using SCM instead of DRAM is only 8%.
- Without prefetching: average penalty for using SCM instead of DRAM is 41%.

NOTE FOR OPERATORS WITH SEQUENTIAL MEMORY ACCESS PATTERNS, SCM PERFORMS ALMOST AS GOOD AS DRAM



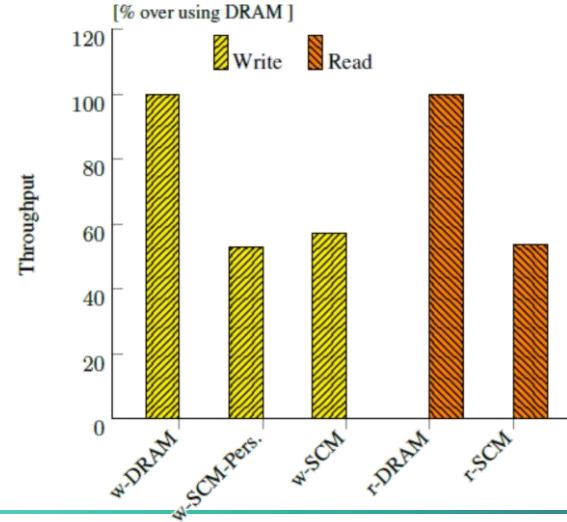
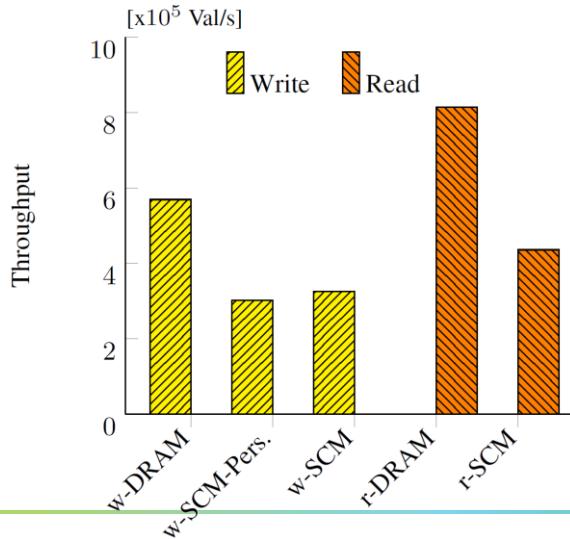
NVRAM for WOS-Structure

MICRO EXPERIMENT

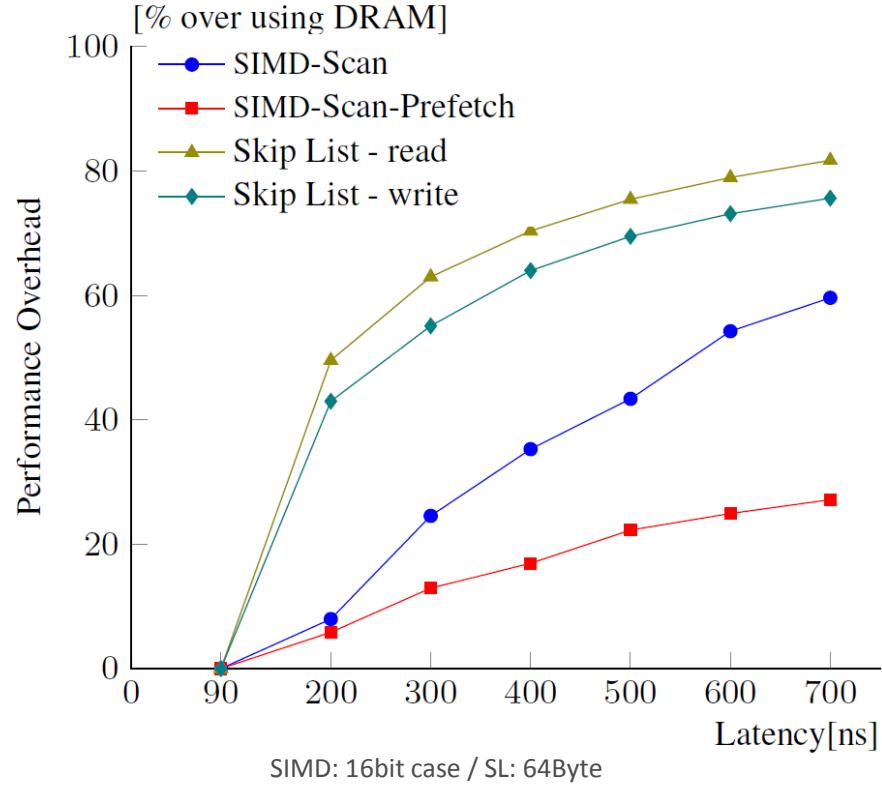
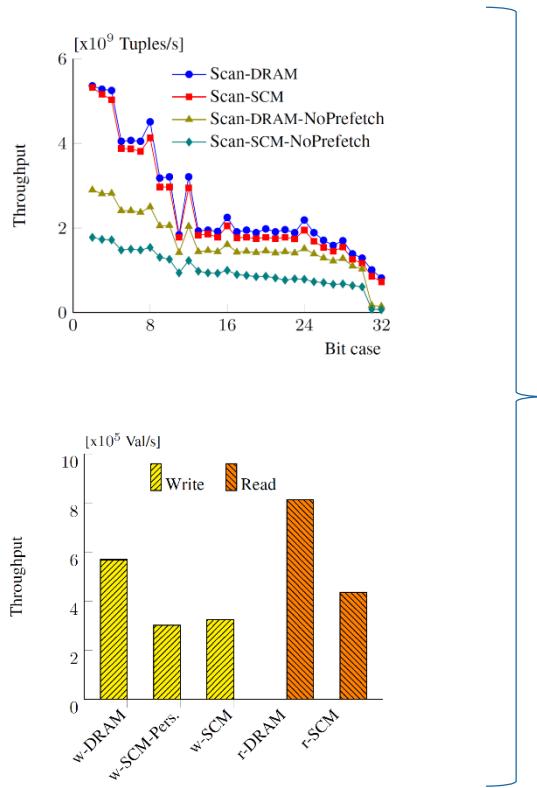
- Skip List read/write performance on DRAM and SCM
- 47% penalty for reads, and 43-47% penalty for writes for using SCM instead of DRAM.

OPERATIONS WITH RANDOM MEMORY ACCESS PATTERNS ARE EXPENSIVE IN SCM

- Writing persistent and concurrent data structures is NOT trivial



Impact of NVRAM latency



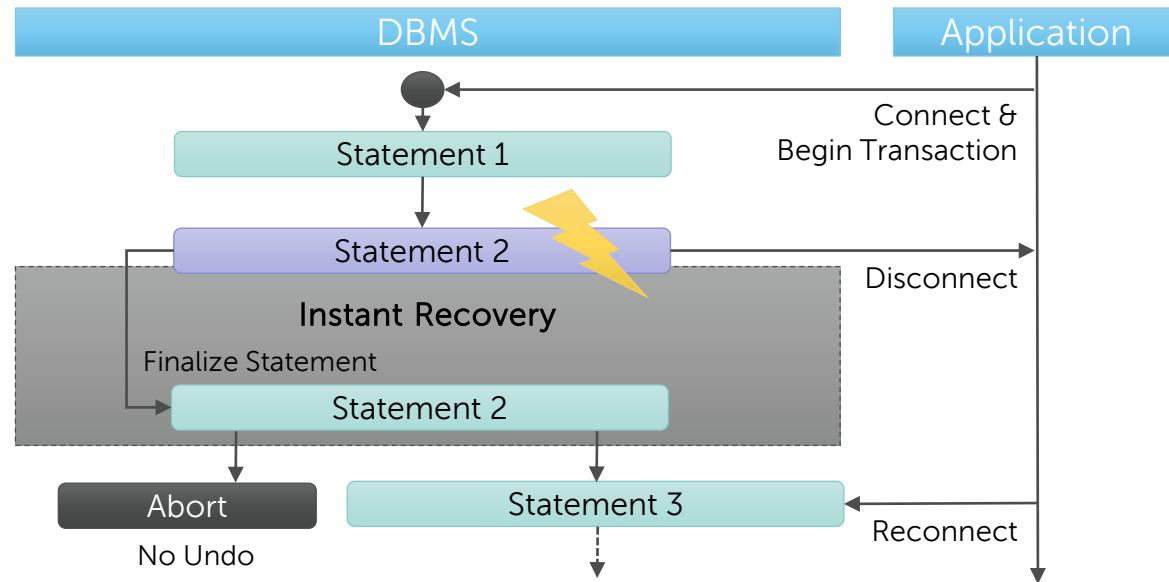
SCM Recovery

DRAWBACK OF TRADITIONAL APPROACH WITH LARGE MAIN-MEMORIES

- System is operational after recovery of all database objects is performed
- Recovery may take a significant amount of time

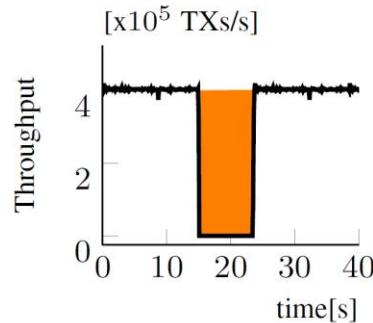
GOAL: INSTANT RECOVERY

- Each executed statement is guaranteed to have persisted its changes in SCM
- The transaction array is persistent, allowing unfinished transactions at crash time to continue
- No need for system-triggered rollback

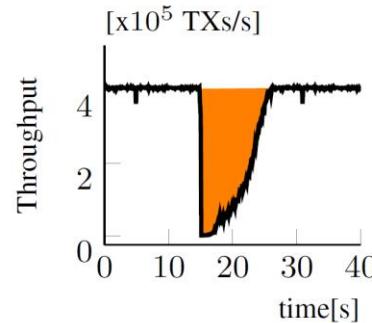


Experiment: Recovery Performance

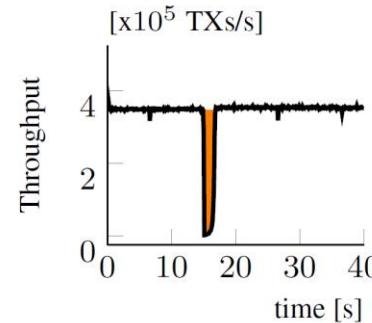
Different recovery schemes. TATP scale factor 500, 4 users. The database is crashed at second 15.



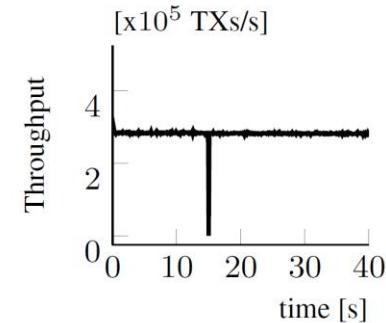
(a) Scenario 1



(b) Scenario 2
no indexes in SCM



(c) Scenario 2
40% indexes in SCM



(d) Scenario 3

- Scenario 1: rebuild all secondary data structures before starting answering queries.
- Scenario 2: rebuild secondary data structures in the background and start immediately answering queries using primary, persistent data. Recovery area decreased by 16%.
- Scenario 3: similar to scenario 2, with 40% persistent secondary data structures. Recovery area decreased by 82%. Throughput decreased by 14%.
- Scenario 4: all secondary data structures are persistent. Recovery area decreased by 99.8%. Throughput decreased by 30%.

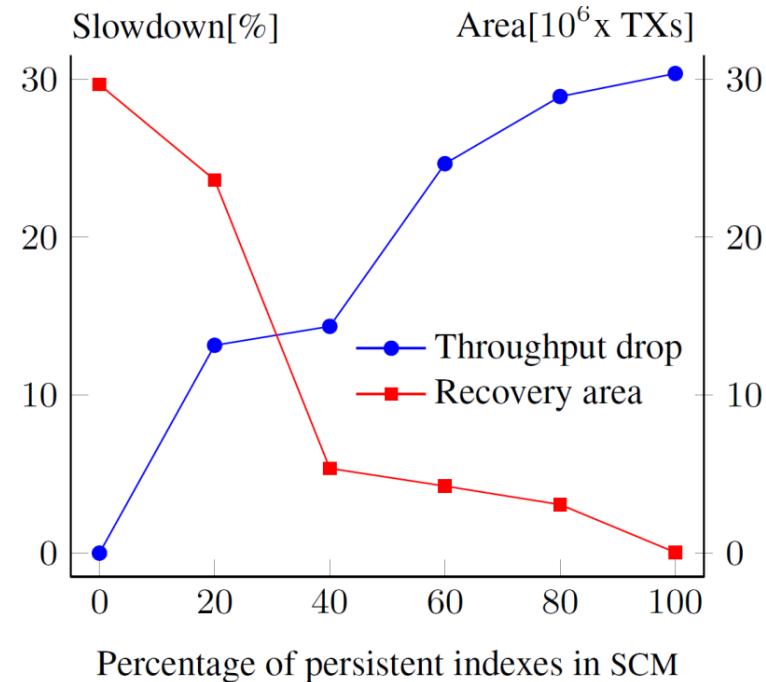
Recovery vs. Query Performance

EXPERIMENT

- Impact of persistent indexes in SCM on throughput during usual processing and recovery area. TATP scale factor 500, 4 users.

OBSERVATION

- The more secondary data structures we persist in SCM, the smaller the recovery area and the lower the throughput during usual processing.
- The throughput drop is limited to 30%.
- Both curves are not linear: not all secondary data structures are equally important.



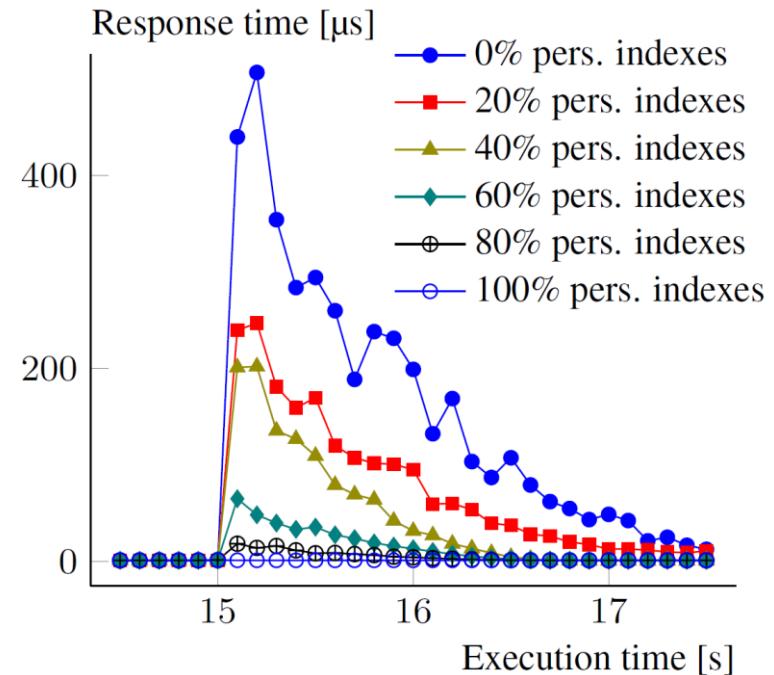
Response Time during Recovery

EXPERIMENT

- average recovery response time. TATP scale factor 500, 4 users.

OBSERVATION

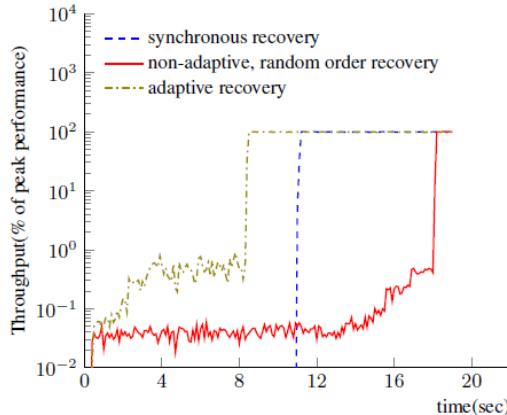
- The more secondary data structures we put in SCM, the lower the average response time during recovery
- Maximum response time for the two extreme cases:
 - All in DRAM: 506 μ s
 - All in SCM: 2 ms



Recovery Optimization

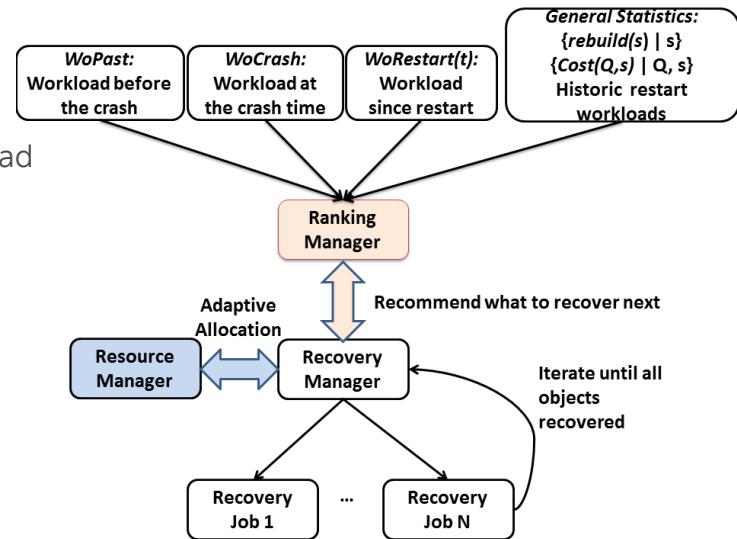
OPTIMIZE THE SEQUENCE OF RECOVERY OBJECTS

- Not all index structures show the same relevance for a given workload
- Workload may change due to recovery procedures



RANKING MANAGER <> RECOVERY MANAGER

RESSOURCE ALLOCATION FOR RECOVERY AND QUERY PROCESSING



$$\text{Benefit}(\text{WoPast}, \text{WoCrash}, s) = \beta * \text{Benefit}(\text{WoPast}, s) + (1 - \beta) * \text{Benefit}(\text{WoCrash}, s)$$

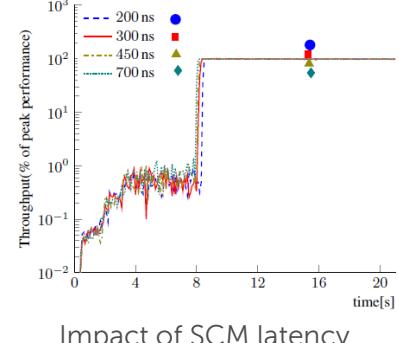
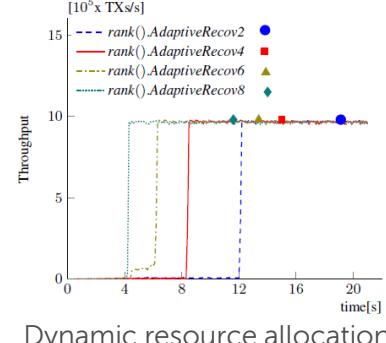
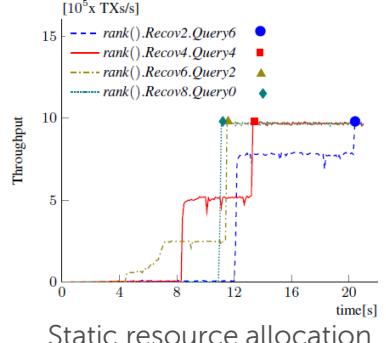
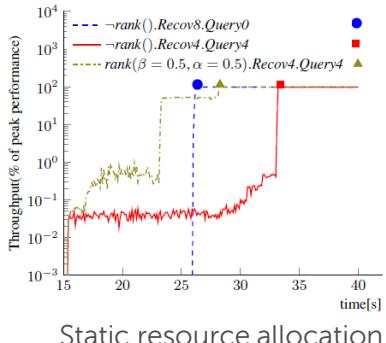
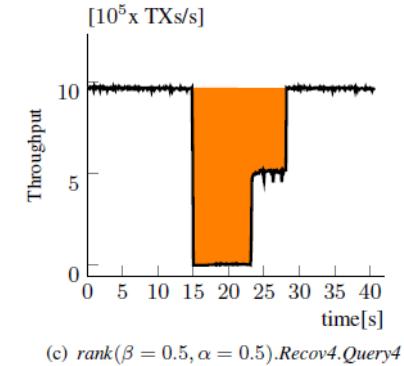
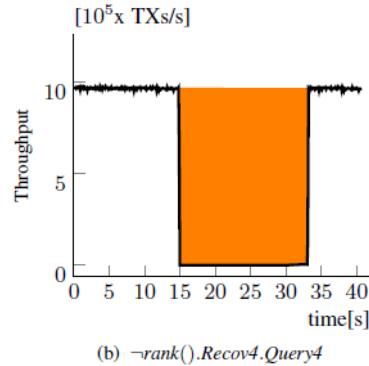
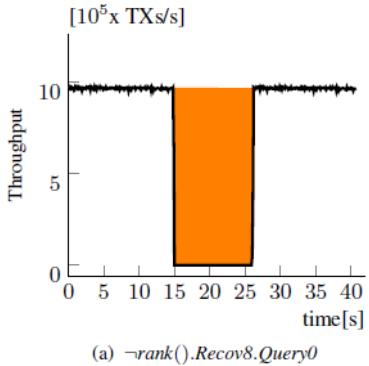
$$\text{rank}(0, s) = \text{Benefit}(\text{WoPast}, \text{WoCrash}, s) - \text{rebuild}(s)$$

$$n = \text{sizeof}(\text{WoRestart}(t))$$

$$\begin{aligned} \text{rank}(t, s) = & \alpha(n) * \text{Benefit}(\text{WoPast}, \text{WoCrash}, s) + \\ & + (1 - \alpha(n)) * \text{Benefit}(\text{WoRestart}(t), s) \\ & - \text{rebuild}(s) \end{aligned}$$

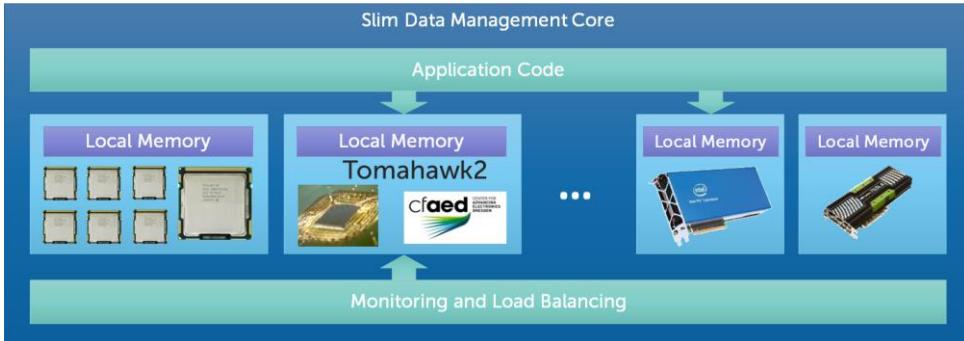
Recovery Optimization (2)

EXPERIMENTAL RESULTS



Summary and Conclusion

Conclusion



- **MODERN HARDWARE DEMANDS SIGNIFICANT RE-THINKING OF DB ARCHITECTURES**

- extreme NUMA Systems: „distributed system“ with common address space
- customizable processors („dark silicon“): revival of the database machine?
- communication: optical / wireless & RMDA:
blurring the boundaries between scale out and scale-up
- Non-Volatile RAM: will be a game-changer!

- **SLIM AND EXTENSIBLE DATA MANAGEMENT CORE TO**

- efficiently support today's applications requirements (mixed OLTP/OLAP workloads etc.)
- embrace and exploit capabilities of modern hardware



Next-Gen Hardware for Data Management

More a Blessing than a Curse?

Wolfgang Lehner – EDBT 2015, Brussels