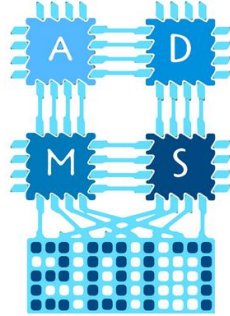


# How Disruptive is Modern Hardware?

Wolfgang Lehner

# Interest in Modern HW?

...IN RESEARCH



Thirteenth International Workshop on  
**Data Management on New Hardware**  
(**DaMoN 2017**)

## HardBD 2017

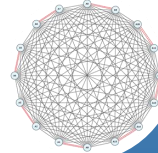
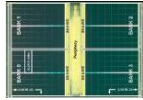
International Workshop on Big Data Management on Emerging Hardware, Sponsored  
and Held in conjunction with ICDE 2017 April 22, 2017, San Diego, USA

## Active'17

First International Workshop on Data Management on  
Virtualized Active Systems.



# Interest in Modern HW?



Compute Units



Memory



Networking

# Interest in Modern HW?

...in commercial DB settings

- some developments (acceleration models, ...)
- last disruptive development: >10 years back!



**Why was In-Memory Computing development disruptive?**

# Disruptiveness

disruptive 

[dis-ruhp-tiv]

Spell Syllables

Examples Word Origin

See more synonyms on [Thesaurus.com](https://www.thesaurus.com)

adjective

1. causing, tending to cause, or caused by disruption; disrupting :  
*the disruptive effect of their rioting.*

2. *Business.*

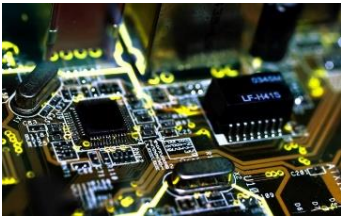
a. relating to or noting a new product, service, or idea that radically changes an industry or business strategy, especially by creating a new market and disrupting an existing one:

*disruptive innovations such as the cell phone and the two-year community college.*

b. relating to or noting a business executive or company that introduces or is receptive to such innovation:  
*disruptive CEOs with imagination and vision.*

a. relating to or noting a new product, service, or idea that radically changes an industry or business strategy, especially by creating a new market and disrupting an existing one:

Disruptions in Data Management always require two ingredients



...novel technology  
and hardware



...novel types of  
DB(!!!) applications



## Why was In-Memory Computing development disruptive?

### A Common Database Approach for OLTP and OLAP Using an In-Memory Column Database

Hasso Plattner  
Hasso Plattner Institute for IT Systems Engineering  
University of Potsdam  
Prof.-Dr.-Heimert-Str. 2-3  
14482 Potsdam, Germany  
hasso.plattner@hpi.uni-potsdam.de



#### Categories and Subject Descriptors

H.2.0 [Information Systems]: DATABASE MANAGEMENT—General

#### General Terms

Design, Performance

After the conversion of attributes into integers, processing becomes faster. More recently, the use of column store databases for analytics has become quite popular. Dictionary compression on the database level and reading only those columns necessary to process a query speed up query processing significantly in the column store case.

I always believed the introduction of so-called data warehouses was a compromise. The flexibility and speed we

→ enabler for HTAP = OLTP & OLAP

# UseCase - Hybrid Transactional Processing



Accounting, Reconciliation, and Reporting application

OLTP

OLAP

## STATISTICS

- 203m active accounts (1st Quarter 2017)
- Online payments in 200+ countries (1st Quarter 2017)
- 6.1 billion payment transactions in 2016

## TRANSACTIONAL DATA VOLUME

- 500 million FTs
- 500 million business partner
- 100 million transaction per day
- ~321 million sub ledger documents per day
- 6 million PDAs per day
- 6 million VDAs per day
- 150 million VDAs in incoming layer
- 150.000 cash entries per day
- ...

→ All on a **single** HANA box (48 TB)





## “Traditional Apps“

Accounting, Reconciliation,  
and Reporting

# Applications



## Data Management System



# Hardware



# The DB Sandwich



“Tradition  
Accounting, Re  
and Reporting

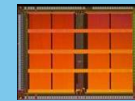
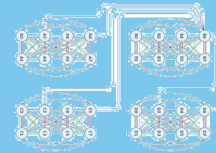
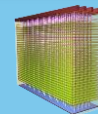
## Big Data Software: What's Next?

Michael Franklin (University of Chicago)

Wednesday, August 30th, 8:30 - 10:00

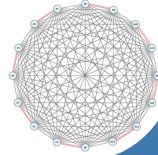
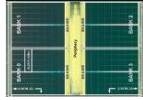
The Big Data revolution has been enabled in part by a wealth of innovation in software platforms for data storage, analytics, and machine learning. The design of Big Data platforms such as Hadoop and Spark focused on scalability, fault-tolerance and performance. As these and other systems increasingly become part of the mainstream, the next set of challenges are becoming clearer. Requirements for performance are changing as workloads evolve to include techniques such as hardware-accelerated deep learning. But more fundamentally, other issues are moving to the forefront. These include ease of use for a wide range of users, security, concerns about privacy and potential bias in results, and the perennial problems of data quality and integration from heterogeneous sources. Fortunately, the database community has much to say about all of these topics, and can and should take a leading role in addressing them. In this talk, I will give an overview of how we got here, with an emphasis on the development of the Apache Spark system. I will then focus on these emerging issues with an eye towards where the database community can most effectively engage.

Data Management  
System



# Outline

Characteristics / Opportunities / Challenges



Compute Units



Memory

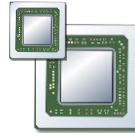


Networking



What is the design space? / What might be a hypothetical HW blueprint?

# Outline

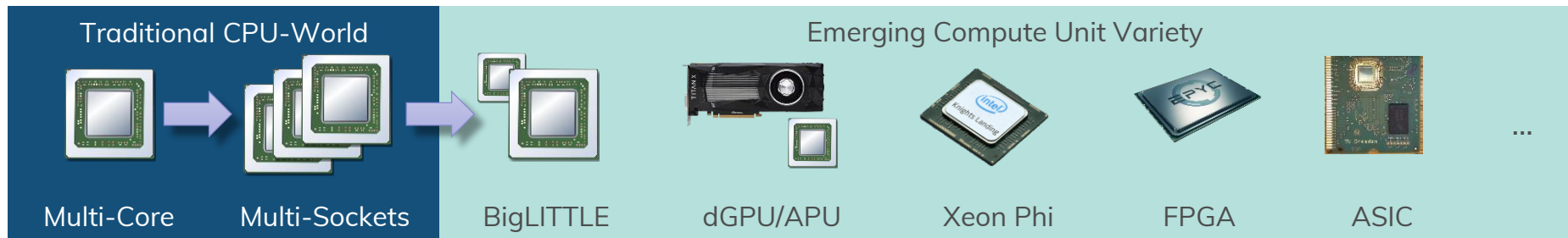


Compute Units

## Compute Unit Diversity

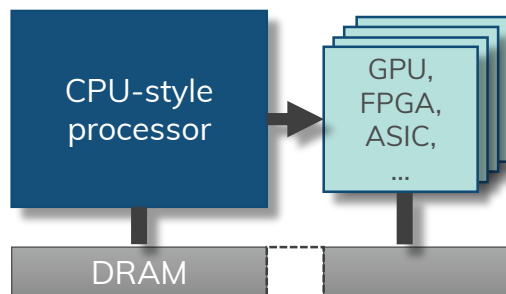
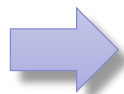


# Compute Unit Diversity



FOCUS ON

Heterogeneous Computing  
by offloading „simple tasks“

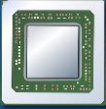


**PERFORMANCE, PERFORMANCE,  
PERFORMANCE!!!**

- specialized data structures and algorithms
- parallel programming models and compiler support
- data and operator placement strategies

# Compute Unit Diversity

Tradition



Multi-Core



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

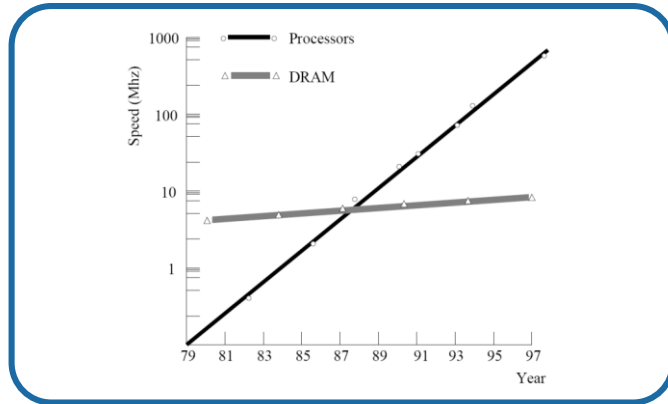


<https://www.top500.org/featured/systems/asci-white-lawrence-livermore-national-laboratory/>

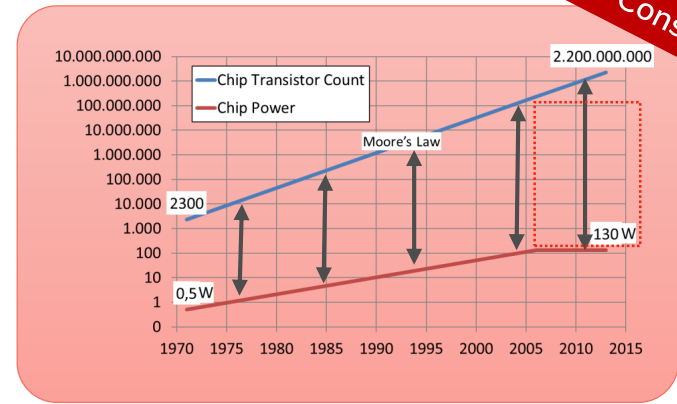
	Nvidia V100 (2017)	IBM ASCI White (2000)
Number of Processor Cores	3584	8192 (512 nodes x 16 IBM Power3)
Double-Precision Performance	7.5 TeraFLOPS	7.2 TeraFLOPS
NVIDIA NVLink™ v2 Interconnect Bandwidth	2x150 GB/s	N/A
PCIe x16 Interconnect Bandwidth	2x16 GB/s	N/A
Memory Capacity	16 GB	<b>6 TB DRAM</b>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

# ...but: we are hitting the „Energy Wall“

## Memory Wall



## Energy Wall



## QUESTIONS:

- 1) Does it matter and is there an impact on database systems (regarding energy savings without compromising performance)?
- 2) Why should the DB community care about it?

Appears in the Proceedings of the 38<sup>th</sup> International Symposium on Computer Architecture (ISCA '11)

## Dark Silicon and the End of Multicore Scaling

Hadi Esmaeilzadeh<sup>†</sup> Emily Blem<sup>‡</sup> Renée St. Amant<sup>§</sup> Karthikeyan Sankaralingam<sup>‡</sup> Doug Burger<sup>\*</sup>

<sup>†</sup>University of Washington <sup>‡</sup>University of Wisconsin-Madison

<sup>§</sup>The University of Texas at Austin <sup>\*</sup>Microsoft Research

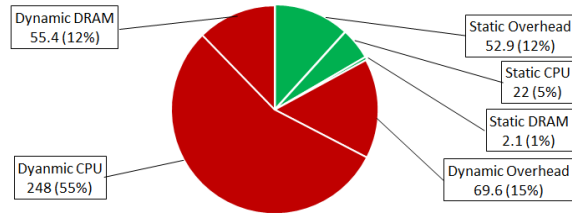
hadiane@cs.washington.edu blem@cs.wisc.edu stamant@cs.utexas.edu karu@cs.wisc.edu dburger@microsoft.com

# Energy Awareness

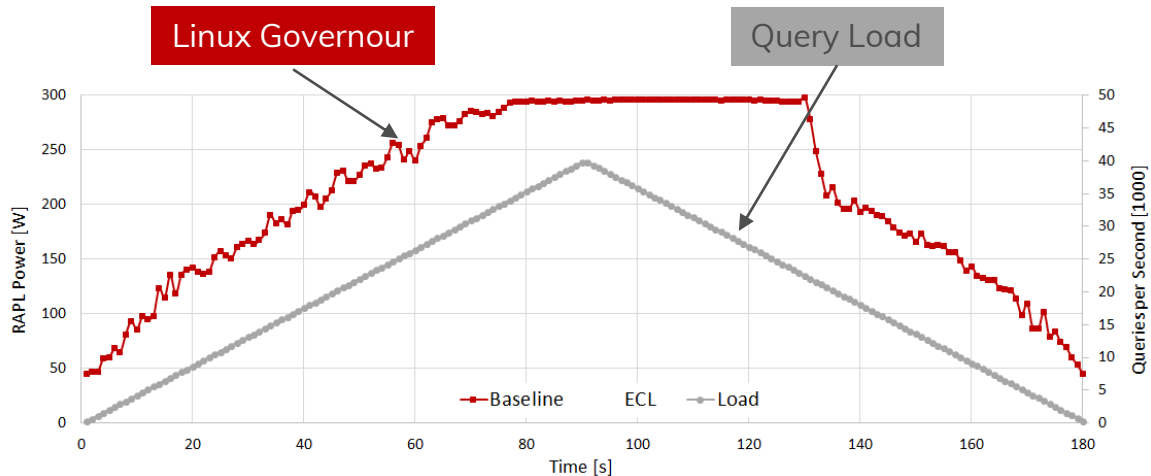
## POWER BREAKDOWN HASWELL-EP

- 19% static
- 81% dynamic

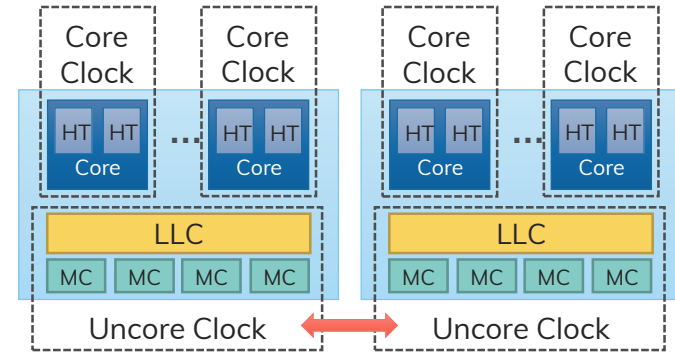
≅ load dependent



## INITIAL EVALUATION



## HARDWARE CONFIGURATION KNOBS

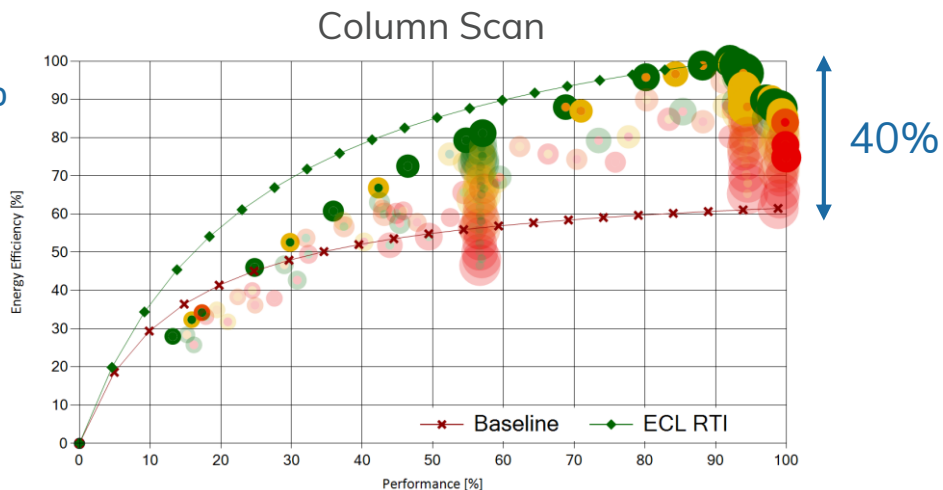
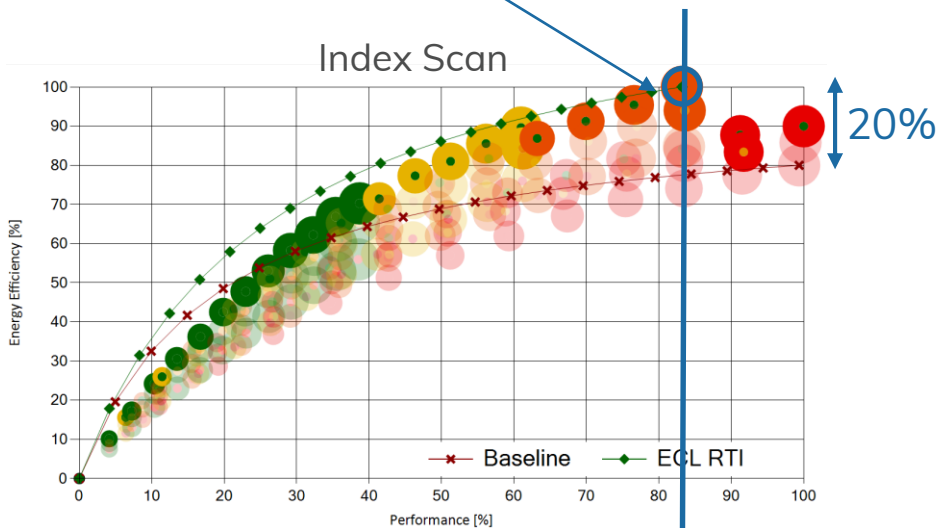
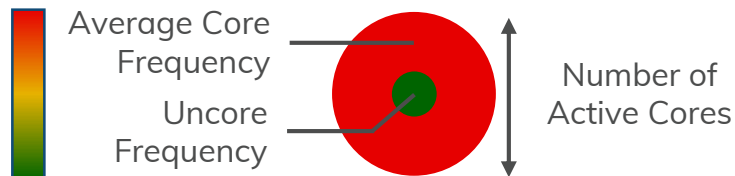


## OBSERVATIONS:

- 1) There are opportunities
- 2) There are many knobs to tune

# ...but: workload knowledge makes a difference

same performance / most energy efficient configuration



There is potential in energy saving without compromising performance!

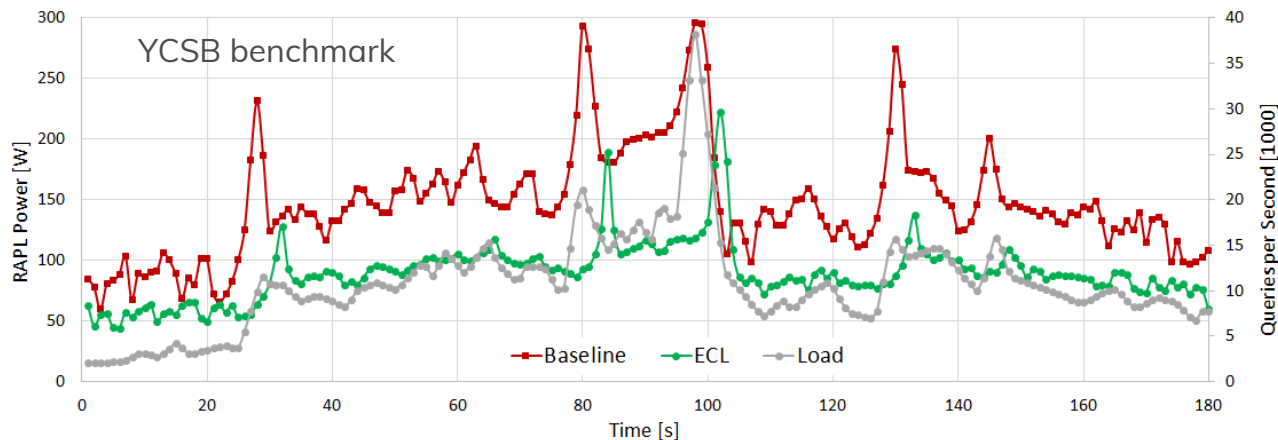
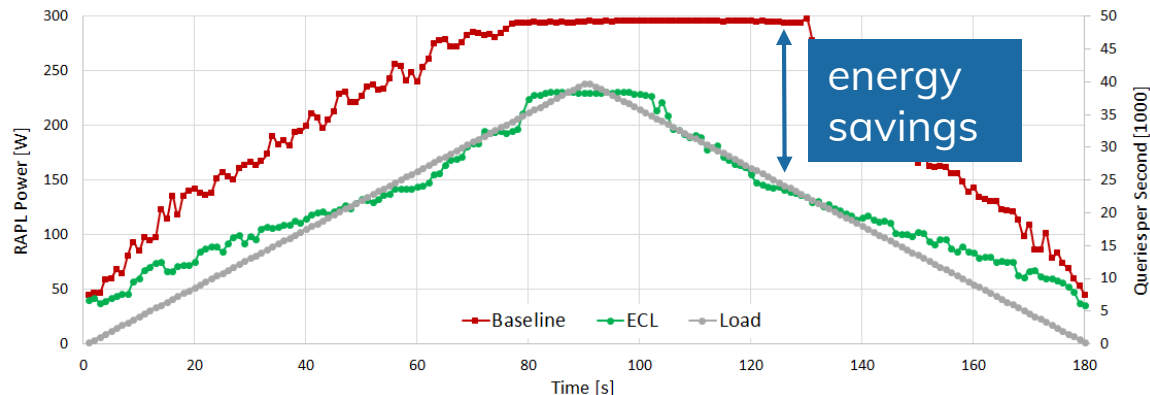


# Energy Savings

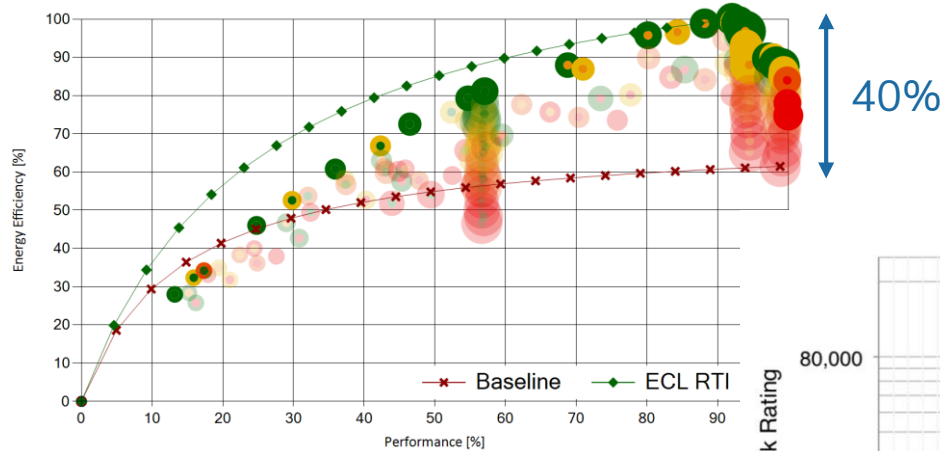
Query Load

Linux Governour

DB-controlled



# Energy Awareness



Is this disruptive?

How far can we push it?

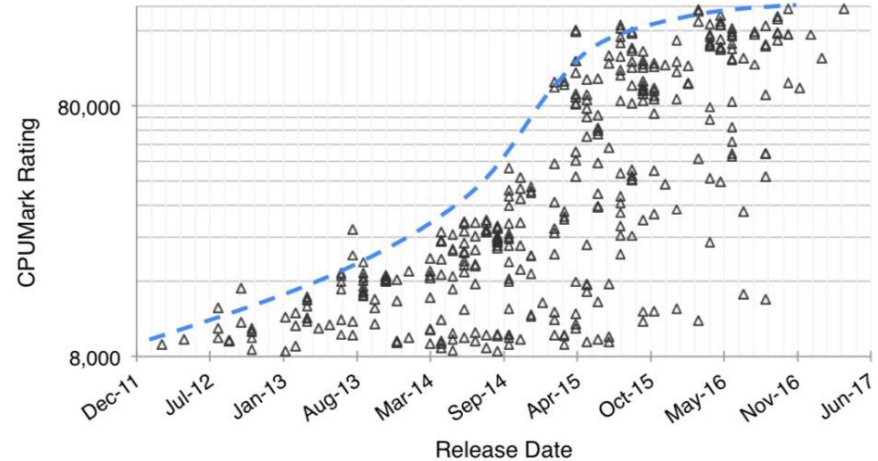
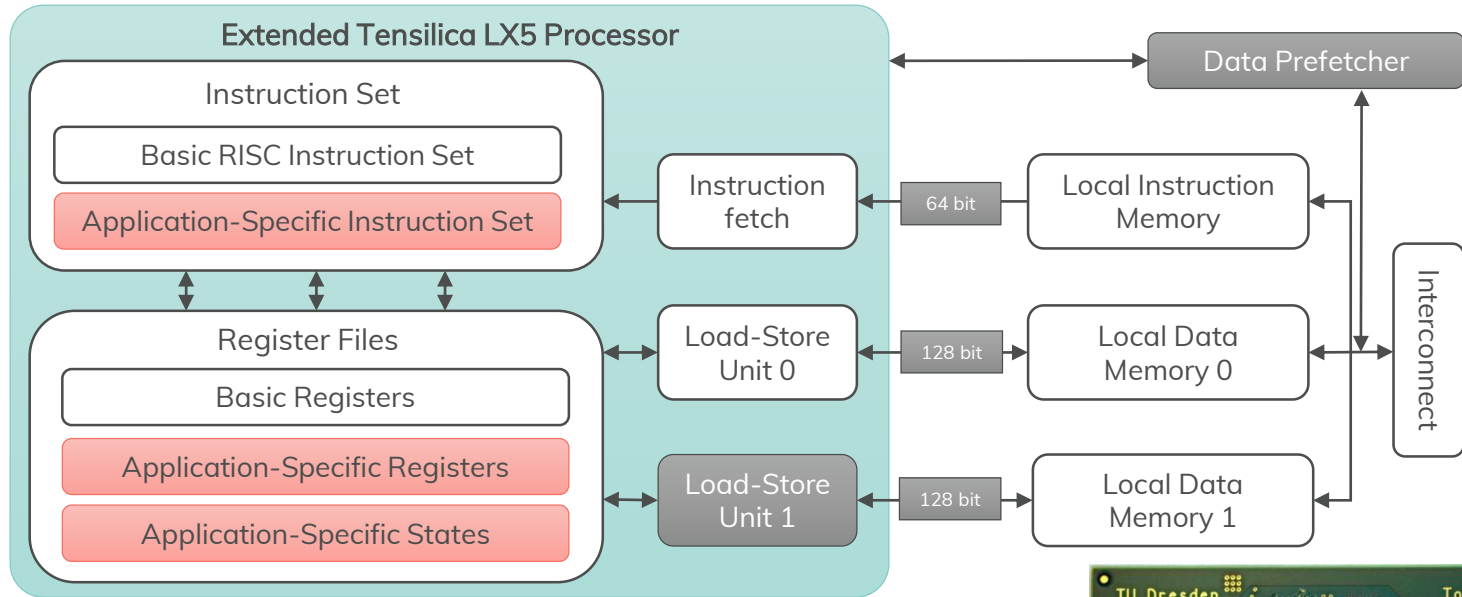


Figure 1: CPU performance trend for Android mobile devices.



- Offloading DB-specific instructions
- Energy efficient design for near-memory deployment

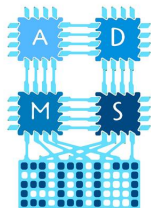


# Tomahawk DBA Primitives

	Bitmap Compression and Processing (AND, OR, XOR)			Hashing				Sorted Set Operations						
Primitives	WAH	PLWAH	COMPAX	Hash + Lookup	Hash + Insert	Hash Keys	Hash Sampling	CityHash32	Merge Sort	Intersection	Union	Difference	Sort-Merge Join	Sort-Merge Aggregation (SUM)

+ development is going on

...more on Friday  
11:40am – 12pm

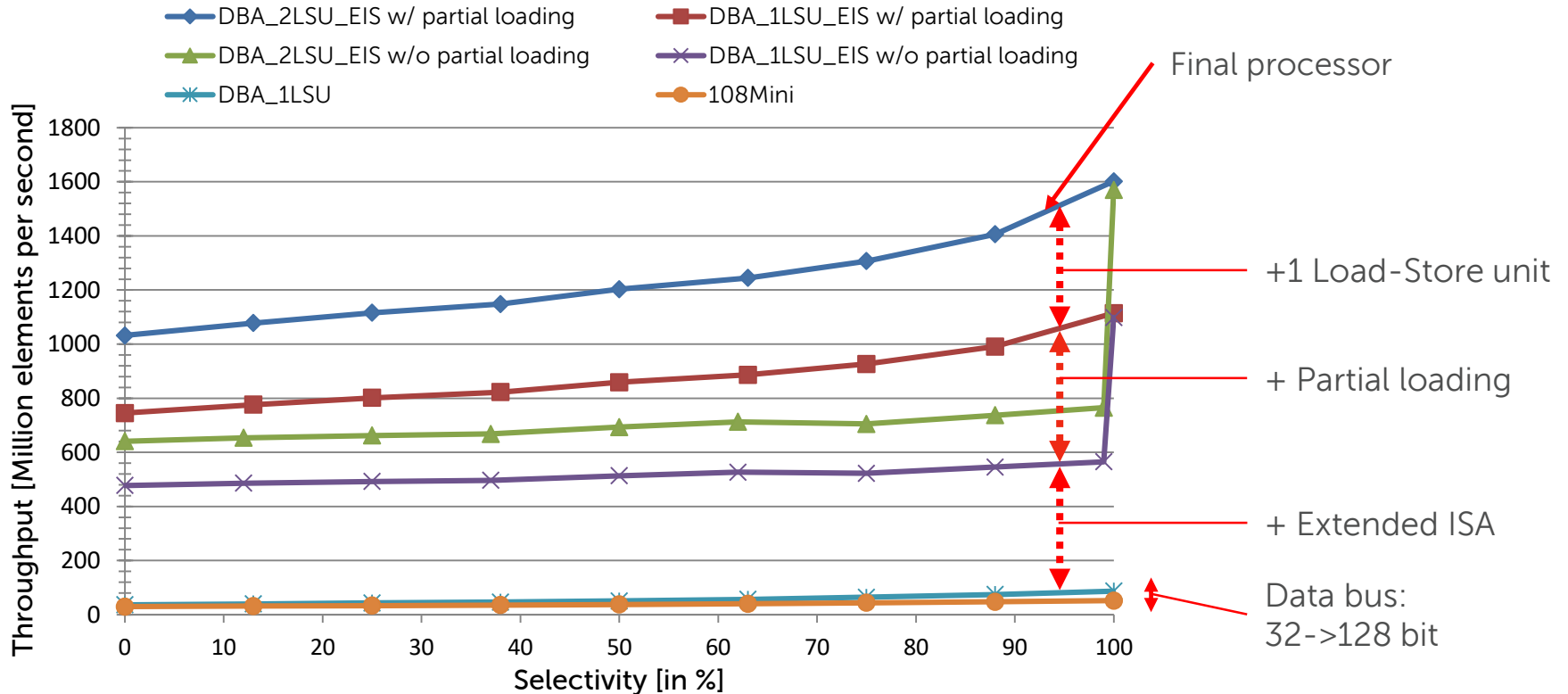


## Energy-Efficient Hash Join Implementations in Hardware-Accelerated MPSoCs

Sebastian Haas, Gerhard Fettweis  
Vodafone Chair Mobile Communications Systems  
Center for Advancing Electronics Dresden (cfaed)  
Technische Universität Dresden, Germany

sebastian.haas@tu-dresden.de, gerhard.fettweis@tu-dresden.de

# Tomahawk DBA: Sorted Set Intersection

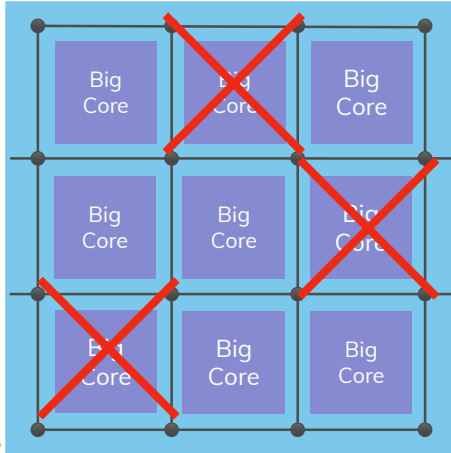


# Tomahawk DBA: Sorted Set Intersection

	INTEL I7-920	DBA_2LSU_EIS	
Throughput (elements/s)	1,100 mio	1,203 mio	→ ~ ±x%
Clock frequency	2.67 GHz	0.41 GHz	
Max. TDP	130 W	>> 0.135 W	
Cores/Threads	4/8	1/1	
Feature size	45 nm	65 nm	
Area (logic & memory)	263 mm <sup>2</sup>	>> 1.5 mm <sup>2</sup>	

## 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



Appears in the Proceedings of the 38<sup>th</sup> International Symposium on Computer Architecture (ISCA '11)

## Dark Silicon and the End of Multicore Scaling

Hadi Esmaeilzadeh<sup>†</sup> Emily Blem<sup>‡</sup> Renée St. Amant<sup>§</sup> Karthikeyan Sankaralingam<sup>¶</sup> Doug Burger<sup>¶</sup>

<sup>†</sup>University of Washington <sup>‡</sup>University of Wisconsin-Madison

<sup>§</sup>The University of Texas at Austin <sup>¶</sup>Microsoft Research

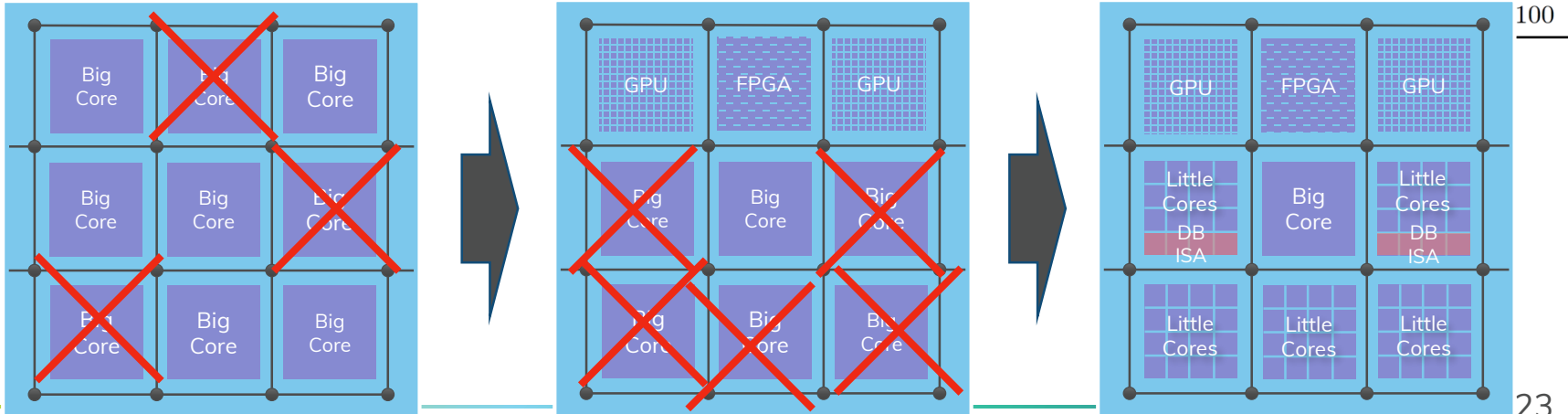
hadiane@cs.washington.edu blem@cs.wisc.edu stamant@cs.utexas.edu karu@cs.wisc.edu dburger@microsoft.com

# Tomahawk DBA: Sorted Set Intersection

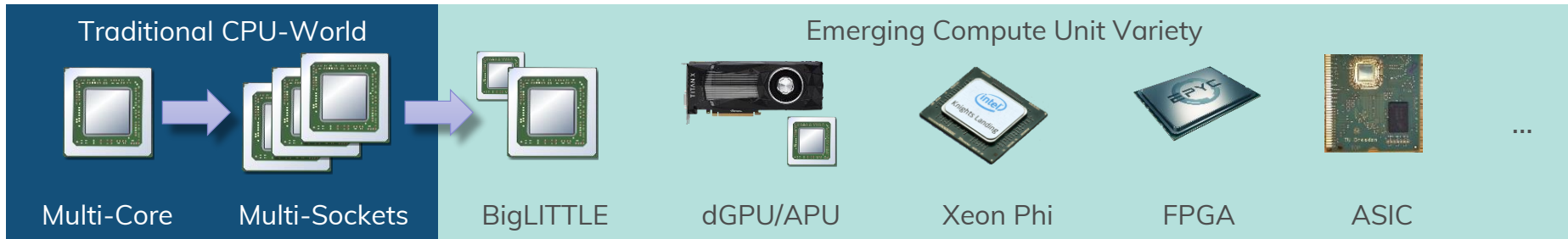
	INTEL I7-920	DBA_2LSU_EIS	
Throughput (elements/s)	1,100 mio	1,203 mio	→ ~ ±x%
Clock frequency	2.67 GHz	0.41 GHz	
Max. TDP	130 W	>> 0.135 W	
Cores/Threads	4/8	1/1	
Feature size	45 nm	65 nm	
Area (logic & memory)	263 mm <sup>2</sup>	>> 1.5 mm <sup>2</sup>	

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



# Summary - Compute Unit Diversity



## Opportunities

Performance  
through parallelism



## Challenges

Life cycle

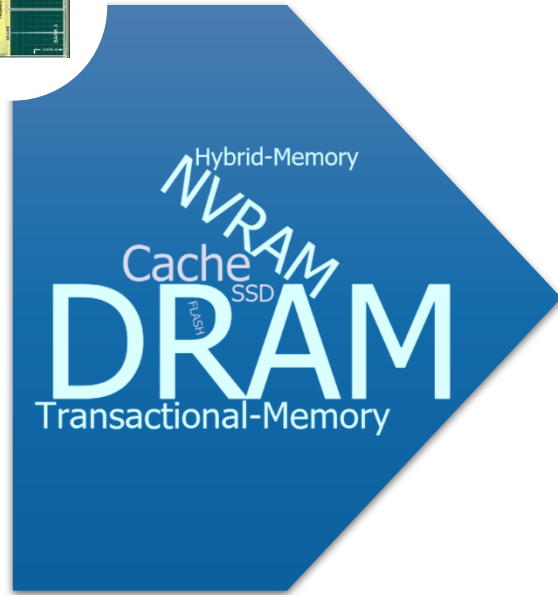
Energy Awareness

Programming Model

Increased system complexity

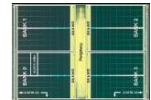
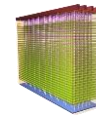
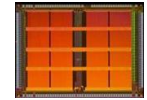
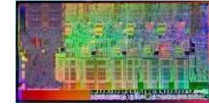
...



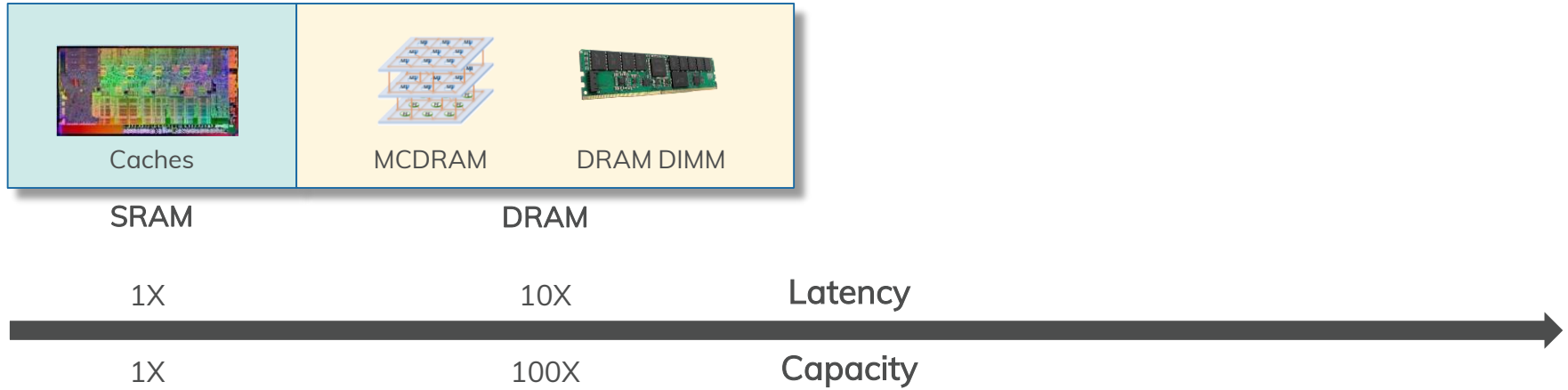


Memory

## Memory Diversity



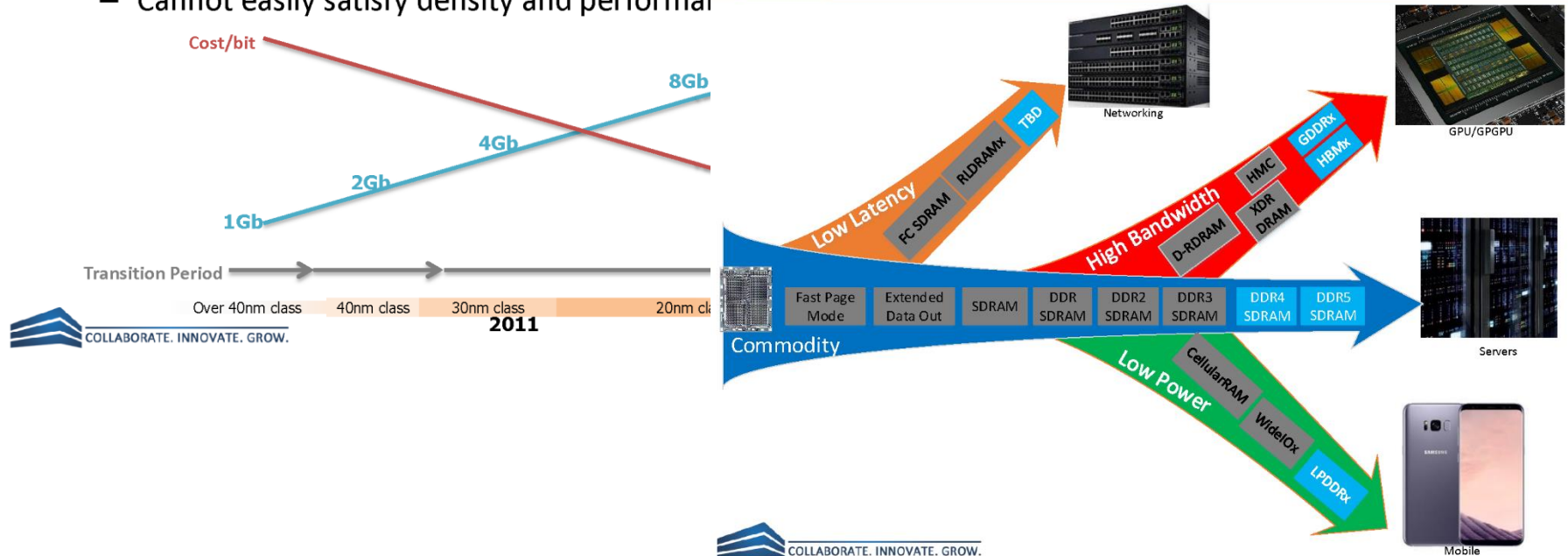
# Memory Diversity



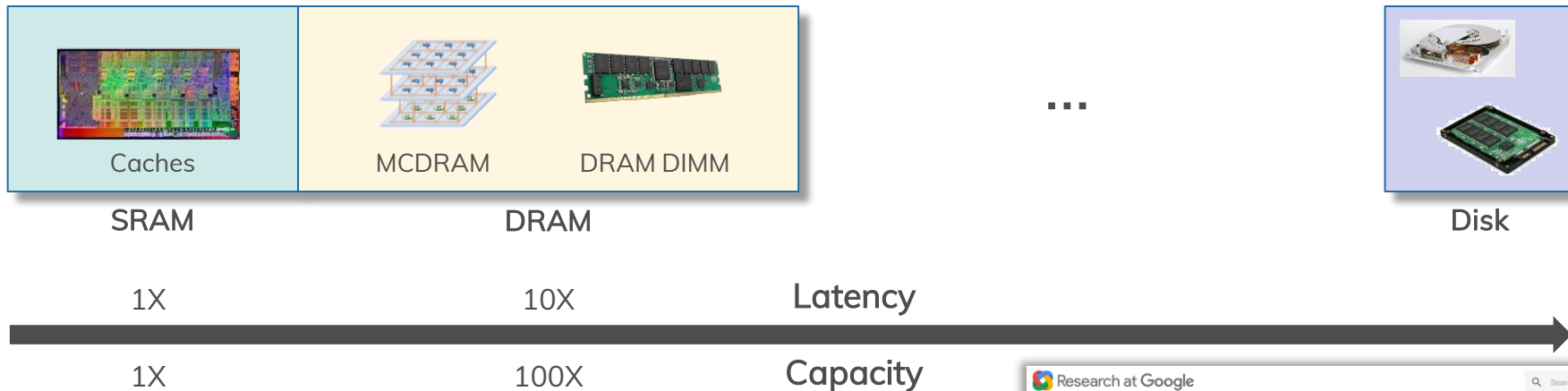
# DRAM Process Scaling Challenge

- DRAM process scaling is slowing significantly
  - Approaching physical limit
  - Migrating to new process difficult and requires large investment
  - Core parameters (ex. Refresh,  $t_{WR}$ , VRT) are getting worse
    - ⇒ increasing fail bit count
  - Cannot easily satisfy density and performance

## DRAM Family Tree and Applications



# Memory Diversity

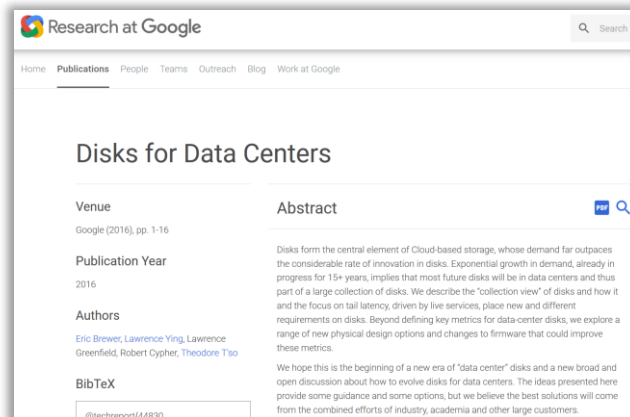


## DEVELOPMENTS

HDD: huge demand for extremely cheap cloud storage  
(→ e.g. new form factors)

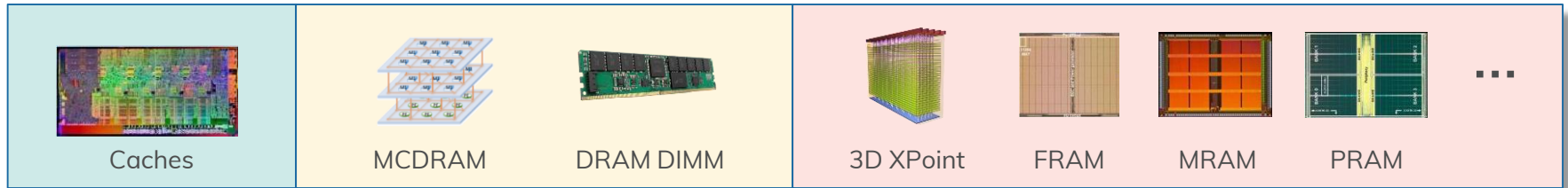
SDD:

- large capacity (> 1PByte) and (relatively) high bandwidth
- significant development ahead
- still (relatively) poor latency



# Memory Diversity

## Merging Point between Storage and Memory



SRAM

DRAM

NVRAM

1X

10X

Latency

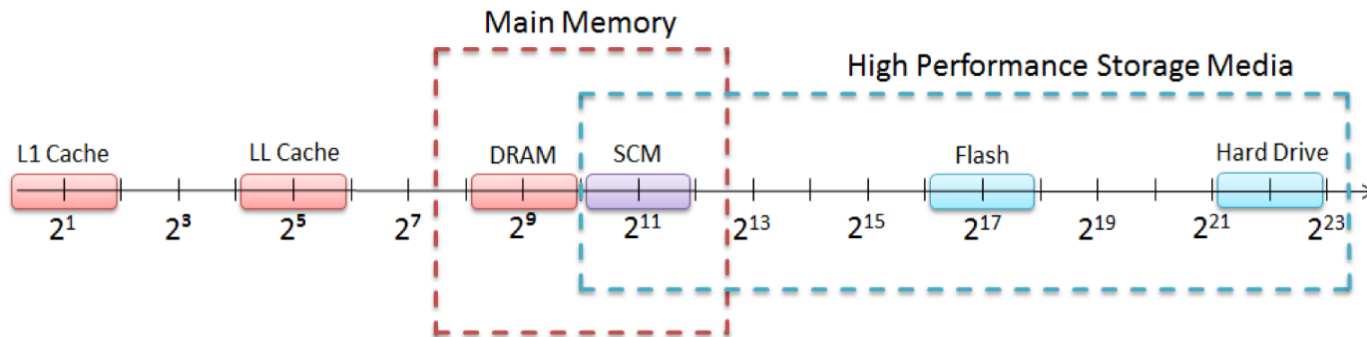
100X

1X

100X

Capacity

1000X



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

# Game Changer? Non-Volatile Memory (NVRAM)



## ADVANTAGES

- ... does not consume energy if not used
- ... is persistent, byte-addressable
- ... x-times denser than DRAM

## DRAWBACKS

- ... has higher latency than DRAM
  - Read latency ~2x slower than DRAM
  - Write latency ~10x slower than DRAM
- Number of writes is limited

	MRAM	DRAM	PCM	ReRAM	TLC NAND
Cost per Bit	~5	1	>0.5	>0.5	0.05
Read Latency	~1	1	10	100	1000
Write Latency	~1	1	50	1000	10000
Volatility	no	yes	no	no	no
Endurance	>1E15	> 1E16	1E6~1E8	1E6	1E3
Write Energy (J/bit)	0.1~100	1	0.1~10	0.1~10	10

Estimates provided by David Wang (Samsung)

IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTING SYSTEMS

## A Survey of Software Techniques for Using Non-Volatile Memories for Storage and Main Memory Systems

Sparsh Mittal, Member, IEEE, and Jeffrey S. Vetter, Senior Member, IEEE

**Abstract**—Non-volatile memory (NVM) devices, such as Flash, phase change RAM, spin transfer torque RAM, and resistive RAM, offer several advantages and challenges when compared to conventional memory technologies, such as DRAM and magnetic hard disk drives (HDDs). In this paper, we present a survey of software techniques that have been proposed to exploit the advantages and mitigate the disadvantages of NVMs when used for designing memory systems, and, in particular, secondary storage (e.g., solid state drive) and main memory. We classify these software techniques along several dimensions to highlight their similarities and differences. Given that NVMs are growing in popularity, we believe that this survey will motivate further research in the field of software technology for NVMs.

**Index Terms**—Review, classification, non-volatile memory (NVM) (NVRAM), flash memory, phase change RAM (PCM) (PCRAM), spin transfer torque RAM (STT-RAM) (STT-MRAM), resistive RAM (ReRAM) (RRAM), storage class memory (SCM), Solid State Drive (SSD).

### 1 INTRODUCTION

FOR all computing systems ranging from hand-held embedded systems to massive supercomputers, memory systems play the primary role in determining their power consumption, reliability, and, unquestionably, application performance. The ever-increasing data-intensive nature of state-of-the-art applications

catches to main memory to secondary storage. The potential benefits of these NVMs stem from their projected physical properties that may allow them to both consume very low power and provide much higher density than projected traditional technologies. For example, the size of a typical SRAM cell is in the range of 125–200F<sup>2</sup>, while that of a PCM and Flash cell is the range of 1–10F<sup>2</sup> and 1–6F<sup>2</sup>, respectively.

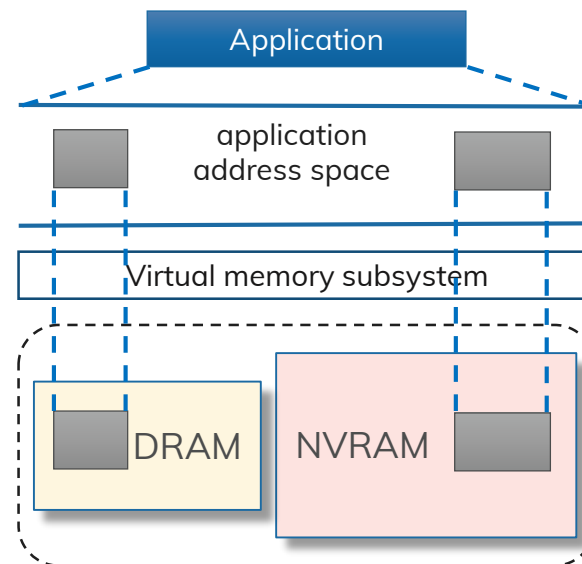
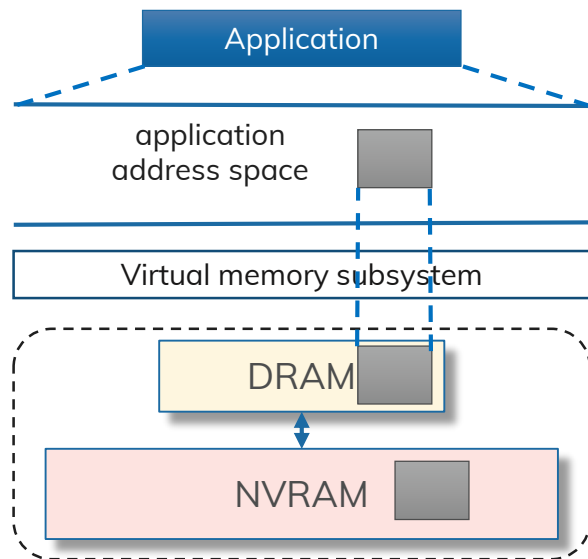
Parameter	NAND	DRAM	PCM	STT-RAM
Read Latency	25 $\mu$ s	50 ns	50 ns	10 ns
Write Latency	500 $\mu$ s	50 ns	500 ns	50 ns
Byte-addressable	No	Yes	Yes	Yes
Endurance	10 <sup>4</sup> –10 <sup>5</sup>	>10 <sup>15</sup>	10 <sup>8</sup> –10 <sup>9</sup>	>10 <sup>15</sup>

# NVRAM as Transient Main Memory

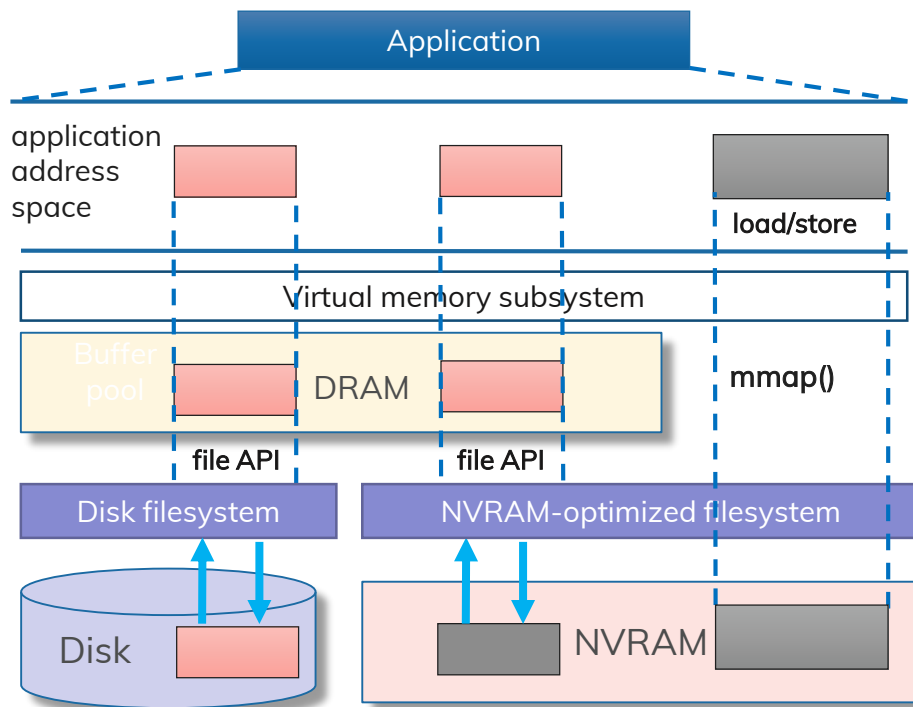
**NVRAM operates in two modes**

DRAM as hardware-managed  
cache for NVRAM

NVRAM next to DRAM



# NVRAM as Persistent Main Memory



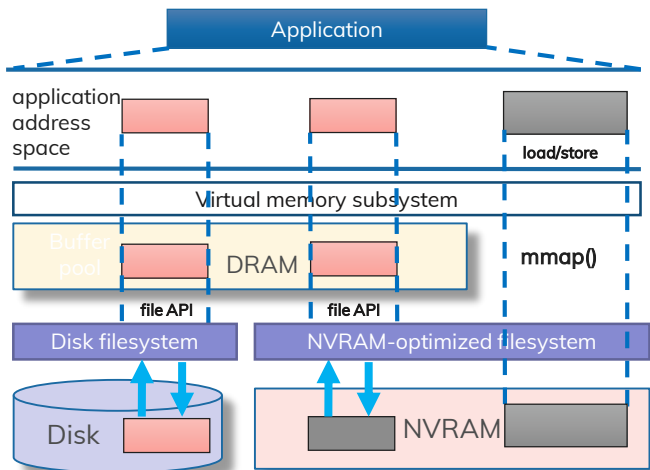
- SNIA recommends to access NVRAM via file `mmap()`
- NVRAM-optimized filesystem provides zero-copy `mmap()`, bypassing the OS page cache
- Linux ext4 and xfs already provide Direct Access support



may result in single-level database, i.e. **the persistent version == the working copy**

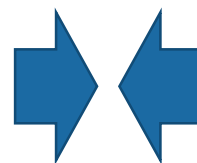
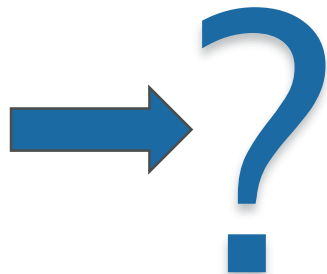


# NVRAM as Universal Memory



“...not fast enough to replace main memory...not cheap enough to replace flash” M. Stonebraker  
<https://www.nextplatform.com/2017/08/15/hardware-drives-shape-databases-come/>

Is this disruptive?



# NVRAM as Universal Memory: Pros and Cons

## CONS / THREADS

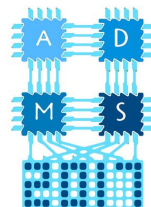
- NVRAM too expensive to fill the gap between DRAM and SSDs
- higher latency is directly visible for state-of-the-art data structures
- little control over when data is persisted due to CPU cache eviction policy or memory reordering
- testing methods required to cover novel types of bugs



## PROS / OPPORTUNITIES

- DRAM may be hitting scalability limits soon
- fits nicely into rack-scale architectural blueprints
- very limited performance degradations for the right data structure with matching access patterns
- provides near instant recovery!  
(loading an X-TeraByte database into Main Memory is a pain!)

...more on Friday  
11:40am – 12pm



### Adaptive Recovery for SCM-Enabled Databases

Ismail Oukid<sup>†\*</sup>

Anisoara Nica<sup>†</sup>

Daniel Dos Santos  
Bossle<sup>†</sup>

Wolfgang Lehner<sup>†</sup>

Peter Bumbulis<sup>†</sup>

Thomas Willhalm<sup>\*</sup>

<sup>†</sup>TU Dresden  
first.last@tu-dresden.de

<sup>†</sup>SAP SE  
first.last@sap.com

<sup>\*</sup>Intel Deutschland GmbH  
first.last@intel.com

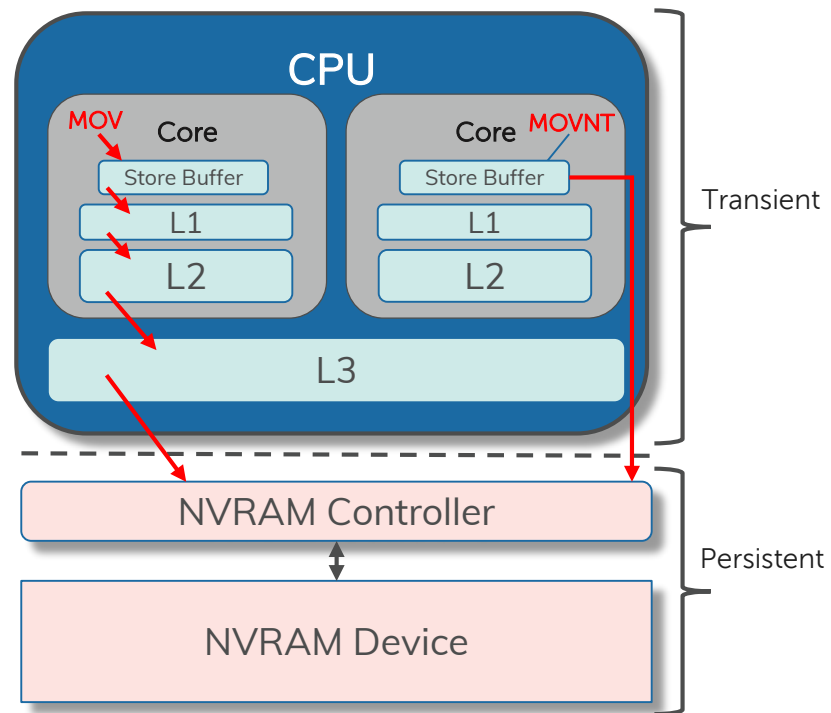
# Programming and Testing Challenges

## (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 resulting in novel failure scenarios



How to ensure consistency of data structures 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();
}
Array.push_back(2017);
```

What is in NVM?

m_size	m_array	
0		✗
1	Corrupt!	✗
0	2017	✗
1	2017	✓

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

à la software  
transactional memory

**pmem.io**  
Persistent Memory Programming

[Home](#) [Glossary](#) [Documents](#) [NVM Library](#) [Blog](#) [About](#)

This site is focused on making *persistent memory programming* easier. The current focus is on the NVM Library, which is a library (set of libraries, actually) designed to provide some useful APIs for server applications wanting to use persistent memory. You can [read more about the NVM Library](#) or [go directly to the source](#). Contributions are welcome!

[Recent Blog Posts](#)

[Using Standard Library](#)

**PROS:**

- low-level optimizations possible

**CONS:**

- programmer must reason about the application state  
→ harder to use and error prone

**PROS:**

- easy to use and to reason about

**CONS:**

- overhead due to systematic logging
- low-level optimizations not possible

# NVM Performance Challenges

## WHAT IS THE COST OF FLUSHING INSTRUCTIONS?

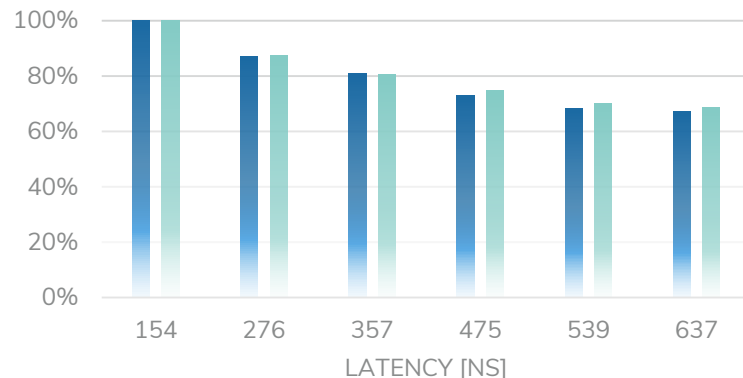
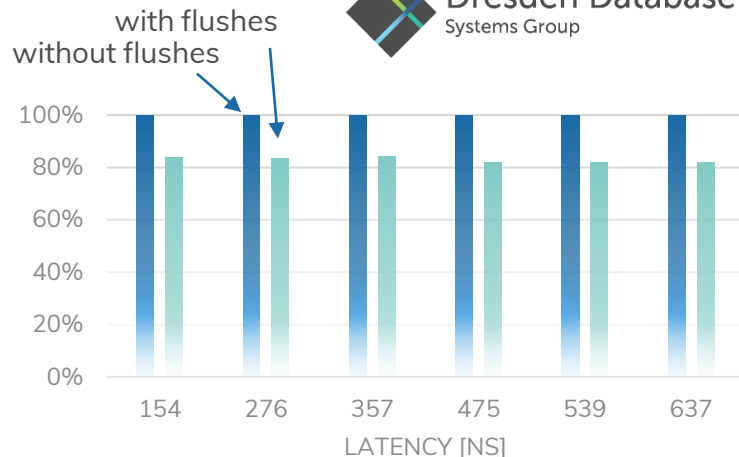
- Prototype hybrid NVM-DRAM storage engine
- TPC-C throughput relative to “without flushes”
- Flushes incur ~18% performance overhead

**Flushes are expensive but agnostic to latency**

## WHAT IS THE EFFECT OF HIGHER NVM LATENCIES?

- TPC-C throughput relative to “baseline NVM latency” (154ns)
- 4x higher latency  $\rightarrow$  ~32% performance penalty with or without flushes

**NVM latency is the main performance-deciding factor**



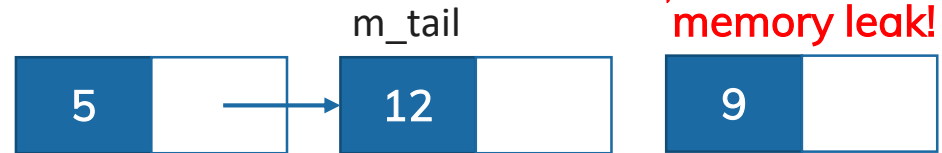
# Persistent Memory Leaks

Novel class of memory leaks resulting from failures

**Example:** crash during a linked-list insertion

```
void append(int val){  
    node *newNode = new node();  
    newNode->value = val;  
    persist(&(newNode->value));  
    m_tail->next = newNode;  
    persist(m_tail);  
    m_tail = newNode;  
    persist(&m_tail);  
}
```

**persistent allocation**



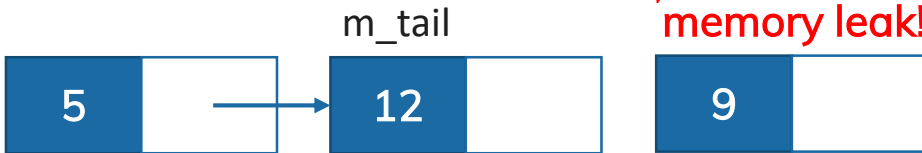
**Failure-induced  
persistent  
memory leak!**

# Persistent Memory Leaks

Novel class of memory leaks resulting from

Example: crash during a linked-list insertion

```
void append(int val){  
    node *newNode = new node();  
    newNode->value = val;  
    persist(&(newNode->value));  
    m_tail->next = newNode;  
    persist(m_tail);  
    m_tail = newNode;  
    persist(&m_tail);  
}  
List.append(9);
```



Failure-induced  
persistent  
memory leak!

novel types of bugs and additional testing overhead

### Memory Management Techniques for Large-Scale Persistent-Main-Memory Systems

Ismail Oukid  
TU Dresden & SAP SE  
ismail.oukid@sap.com

Daniel Booss  
SAP SE  
daniel.booss@sap.com

Adrien Lespinasse  
Independent  
adrien.lespinasse@gmail.com

Grégoire Gomes  
Grenoble INP  
gregoire.gomes@gmail.com

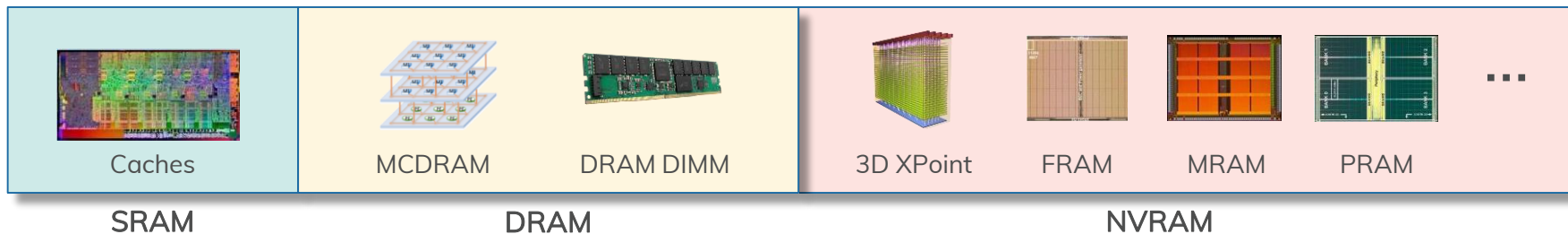
Thursday at 3:30PM

**ABSTRACT** Storage Class Memory (SCM) is a novel class of memory technologies that promise to revolutionize database architectures. SCM is byte-addressable and exhibits latencies similar to those of DRAM, while being non-volatile. Hence, SCM could replace both main memory and storage, enabling a novel single-level database architecture without the traditional I/O bottleneck. Fail-safe persistent SCM allocation can be considered *conditio sine qua non* for enabling this novel architecture paradigm for database management systems. In this paper we present FAllocator, a fail-safe persistent SCM allocator whose design

Parameter	Candidates [18]			
	NAND	DRAM	PCM	STT-RAM
Read Latency	25 $\mu$ s	50 ns	50 ns	10 ns
Write Latency	500 $\mu$ s	50 ns	500 ns	50 ns
Byte-addressable	No	Yes	Yes	Yes
Endurance	$10^4$ - $10^5$	$>10^{15}$	$10^8$ - $10^9$	$>10^{15}$

As the technology matures, we expect the first few generations of SCM to exhibit higher latencies than DRAM, especially for writes. Other promising SCM candidates that were subject to industry announcements include Intel and Micron's 3D XPoint, Resistive RAM (RRAM) [13], and HP's Memristors [26]. Given

# Summary - Memory Diversity



## Opportunities

Performance through more In-Memory data

Enables In-Memory DBs beyond the 100TByte range

Provide in-Memory solution for hard-to-partition OLTP databases, graph algorithms, ...



## Challenges

Cost

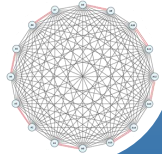
Write Endurance

R/W Symmetry

Testing

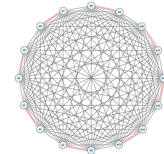
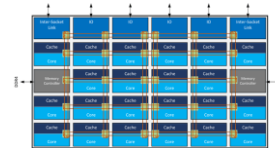
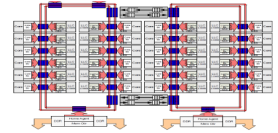
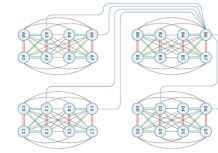
Increased algorithmic complexity



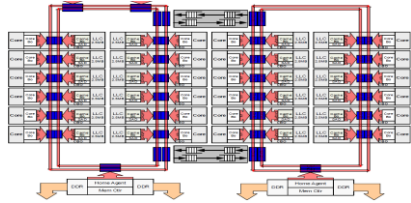


Networking

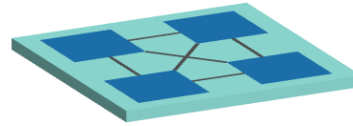
## Network Diversity



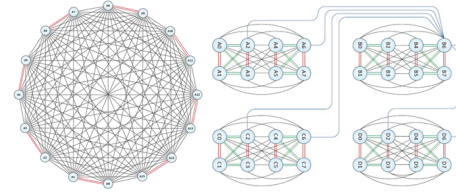
# Network Diversity



Ring Network  
*Haswell-EP*



Fully Connected



Fully Connected

Fat Tree



Infiniband, etc.



data locality is king, moving data is evil!!!



**...most critical component for data-centric systems**

- key for separation of compute and memory
- core prerequisite for providing elasticity in database systems

# Fast Networks: Infiniband & RDMA

## The End of Slow Networks: It's Time for a Redesign

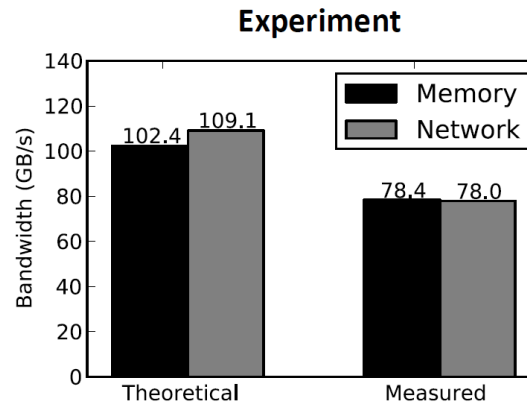
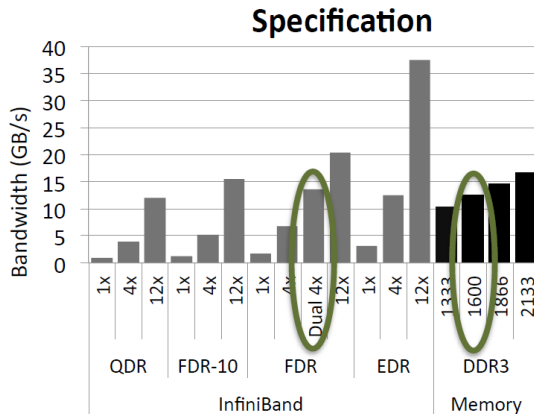
Carsten Binnig Andrew Crotty Alex Galakatos Tim Kraska Erfan Zamanian

Department of Computer Science, Brown University

{firstname\_lastname}@brown.edu

PVLDB 2016

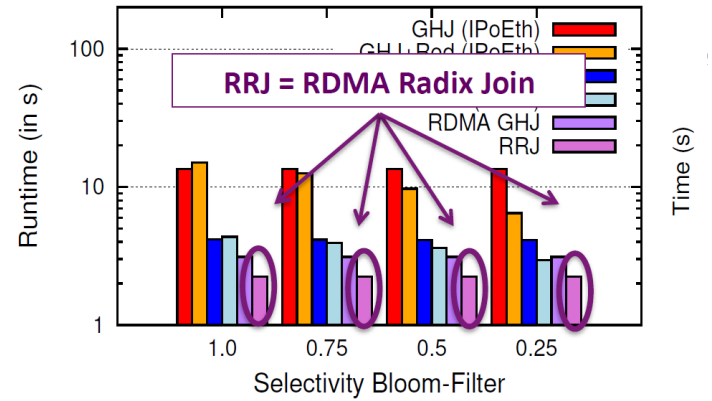
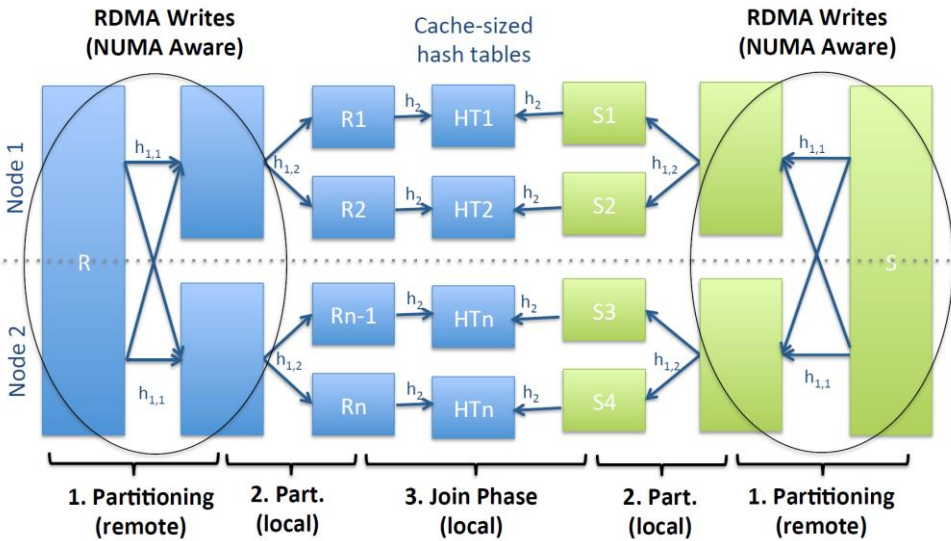
## Network vs. Memory Bandwidth:



Machine: 2 Sockets, 4 NICs

⇒ Network bandwidth is not a bottleneck anymore  
(Latency is still 10x higher for remote access)

# Example: Distributed Radix Join



## Workload:

- **Joins:** Classic Dist. Join (Slow Net), Classic Dist. Join (Fast Net), RDMA Joins (Fast Net)
- **Data:** 512M records per table x 2 on 4 servers (1Gb Ethernet + FDR 4x IB)

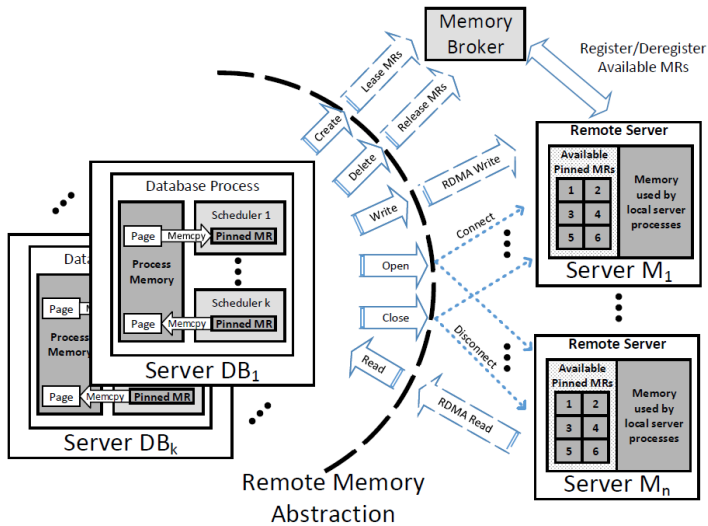
## Listing 1: Remote Radix-Partitioning

```

1 for each tuple r in R do{
2   h = radix-hash(key(r));
3   buffer[h].append(r);
4
5   if (buffer.isFull()){
6     copy_counter[h]++;
7     if (partition[h] is local){
8       memcpy(buffer[h], partition[h]);
9     }
10    }
11    else{
12      signalled = (counter[h]==N);
13      RDMA_WRITE(buffer[h], partition[h], signalled);
14    }
15    buffer.empty();
16 }

```

# Example: Memory Extensions



## Accelerating Relational Databases by Leveraging Remote Memory and RDMA

SIGMOD 2016

Feng Li    Sudipto Das    Manoj Syamala    Vivek R. Narasayya

Microsoft Research  
Redmond, WA 98052, USA  
{fenl, sudiptod, manoj, viveknar}@microsoft.com

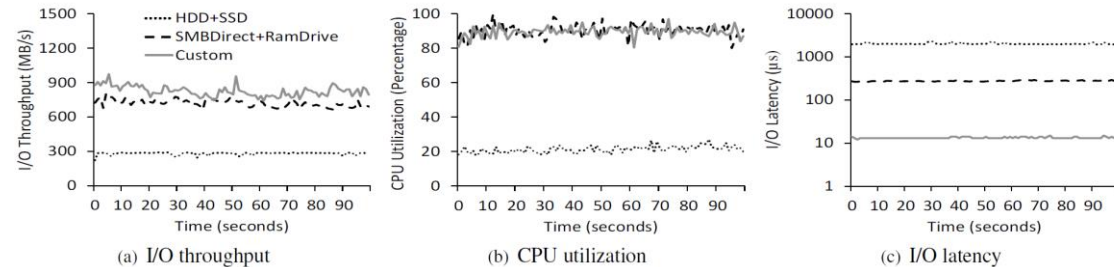


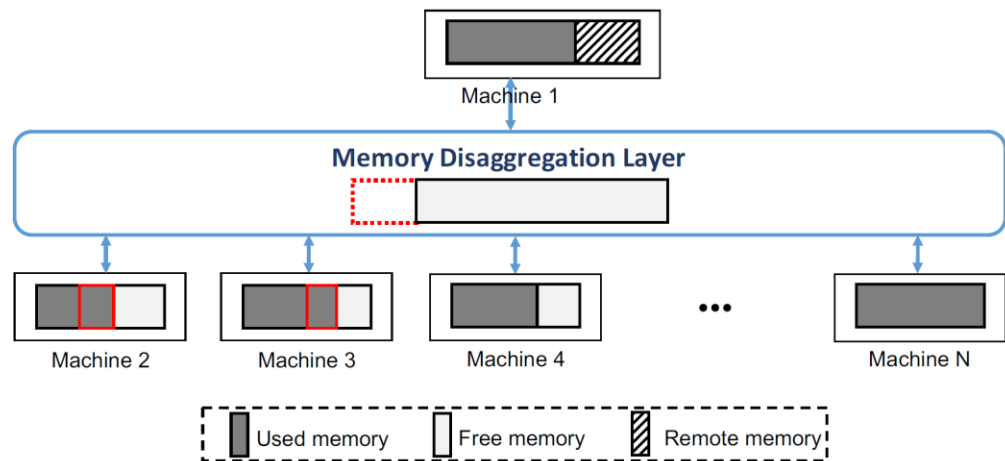
Figure 1: Integrating remote memory into an RDBMS.

... many more similar approaches...

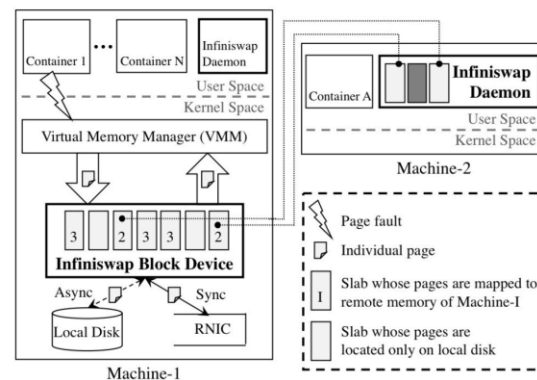
# ...even more Memory Extensions

## Efficient Memory Disaggregation with INFINISWAP

Juncheng Gu, Youngmoon Lee, Yiwen Zhang, Mosharaf Chowdhury, Kang G. Shin  
University of Michigan



NSDI 2017

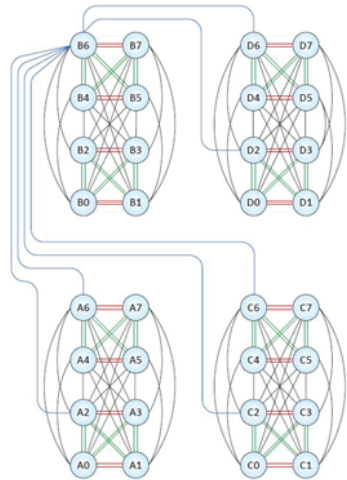


**Figure 3:** INFINISWAP architecture. Each machine loads a block device as a kernel module (set as swap device) and runs an INFINISWAP daemon. The block device divides its address space into slabs and transparently maps them across many machines' remote memory; paging happens at page granularity via RDMA.

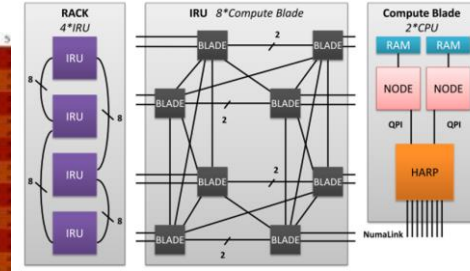
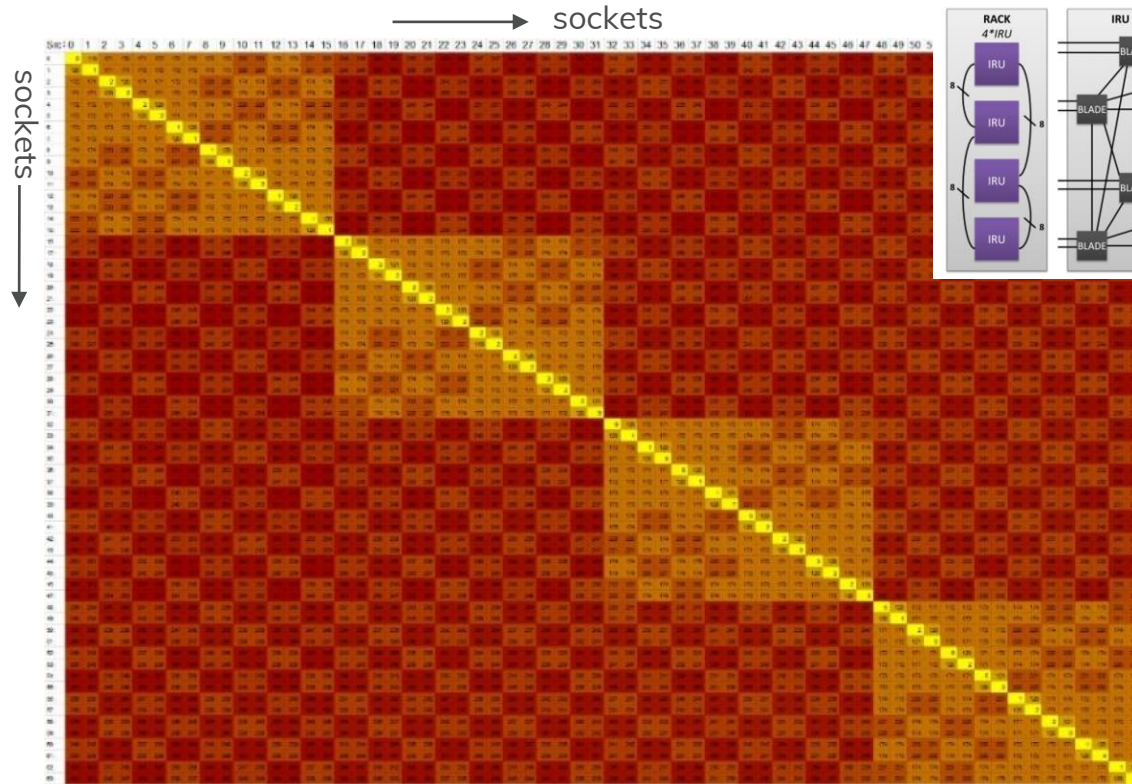
➔ pathfinding projects towards Rack-scale computing

# ...but still: Latency matters!!!

HPE SGI UV 3000

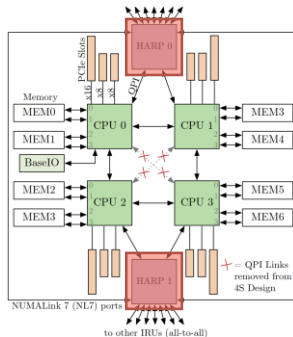


Fat Tree  
Topology



 Up to 82% bandwidth penalty & factor of 10 latency penalty

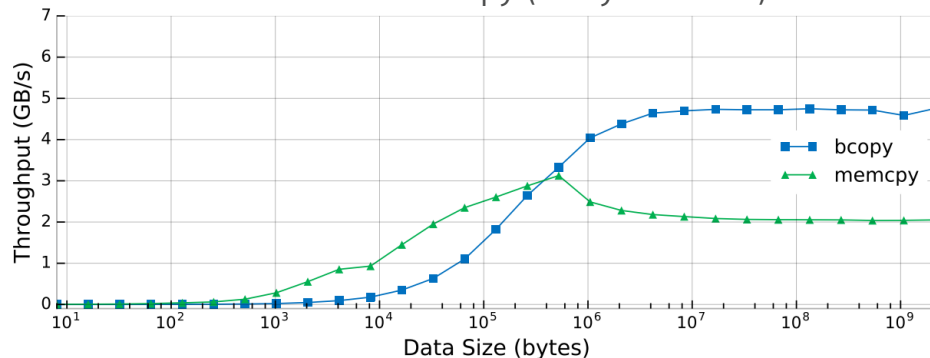
# Solution ? - Accelerated Memory Operations



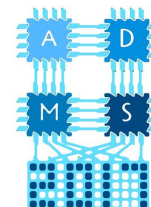
## SGI Global Reference Unit (GRU)

- ▶ Global Shared Memory & Cache Coherency
- ▶ Explicit Offloading

Socket-to-Socket Copy (4GByte chunks)



 **requires cost-based decision during runtime!**



...more on Friday  
1.10 pm - 1.35 pm

## Hardware-Accelerated Memory Operations on Large-Scale NUMA Systems

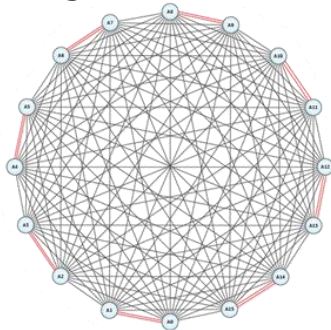
Markus Dreseler Timo Dürken  
Matthias Uflacker Hasso Plattner  
Hasso Plattner Institute  
Potsdam, Germany  
{first.last}@hpi.uni-potsdam.de

Thomas Kissinger Eric Lübke  
Dirk Habich Wolfgang Lehner  
Database Systems Group  
Technische Universität Dresden  
{first.last}@tu-dresden.de

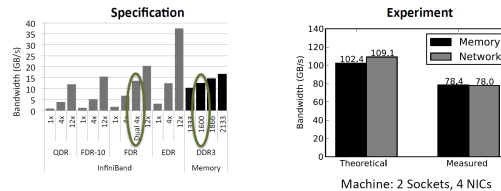


# Network Developments

All-to-All topology,  
e.g. NUMalink 7

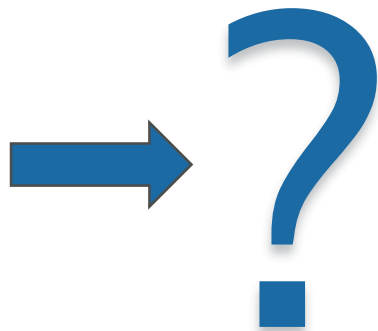


Network vs. Memory Bandwidth:



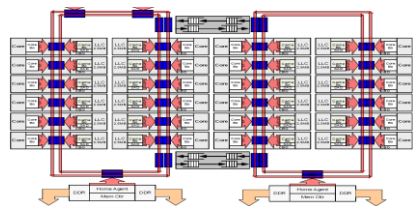
⇒ Network bandwidth is not a bottleneck anymore  
(Latency is still 10x higher for remote access)

Is this disruptive?

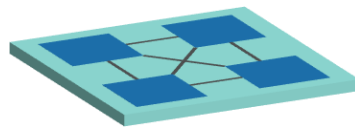


What is next?

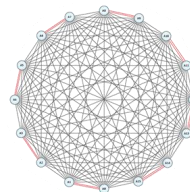
# Network Diversity



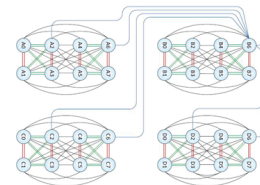
Ring Network  
*Haswell-EP*



Fully Connected



Fully Connected

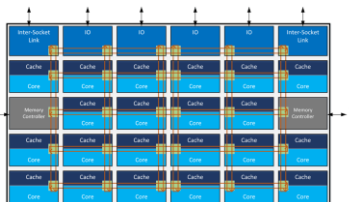


Fat Tree

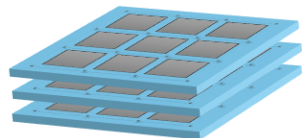


Infiniband, etc.

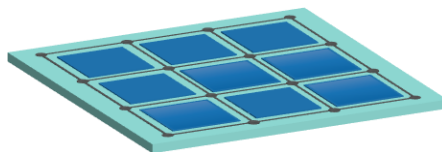
On-Chip → On-Board → Cross-Board → Cross-Node



2D Mesh  
*Skylake-X*



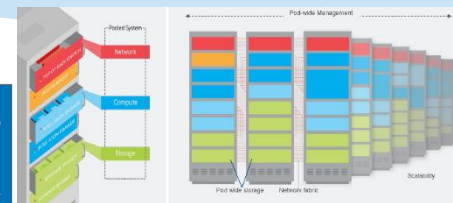
3D Mesh



2D Mesh  
*Photonics*

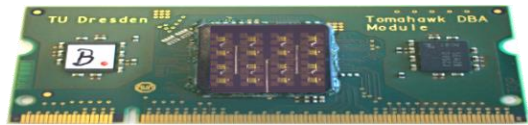
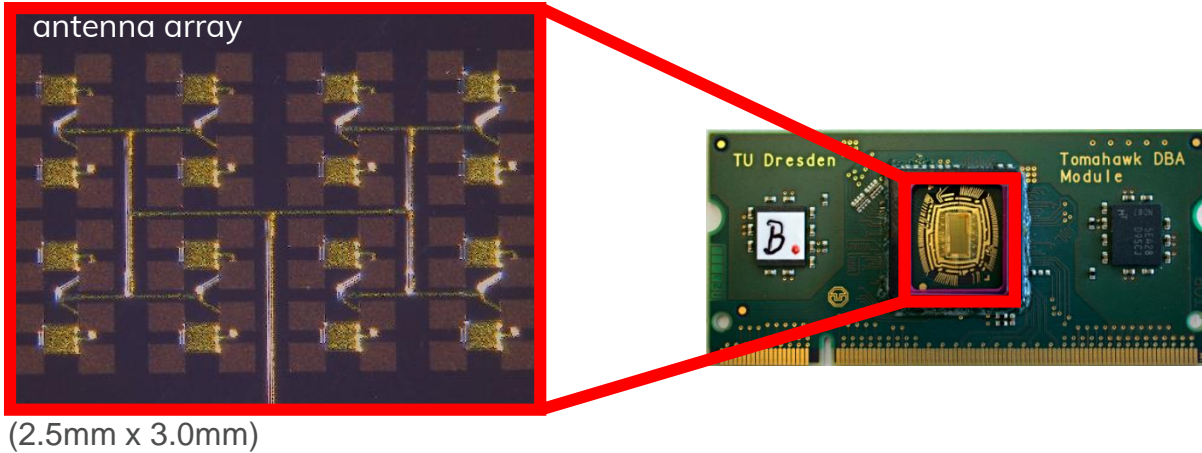


HP  
*The Machine*

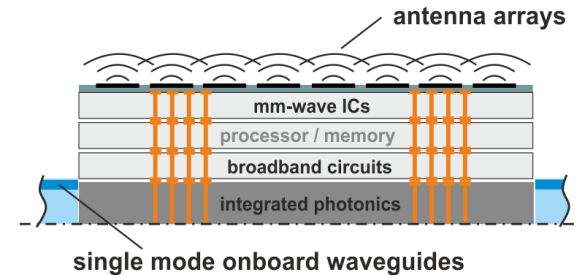


Intel RSA  
*Rack-scale architecture*

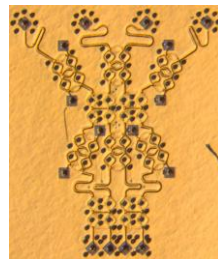
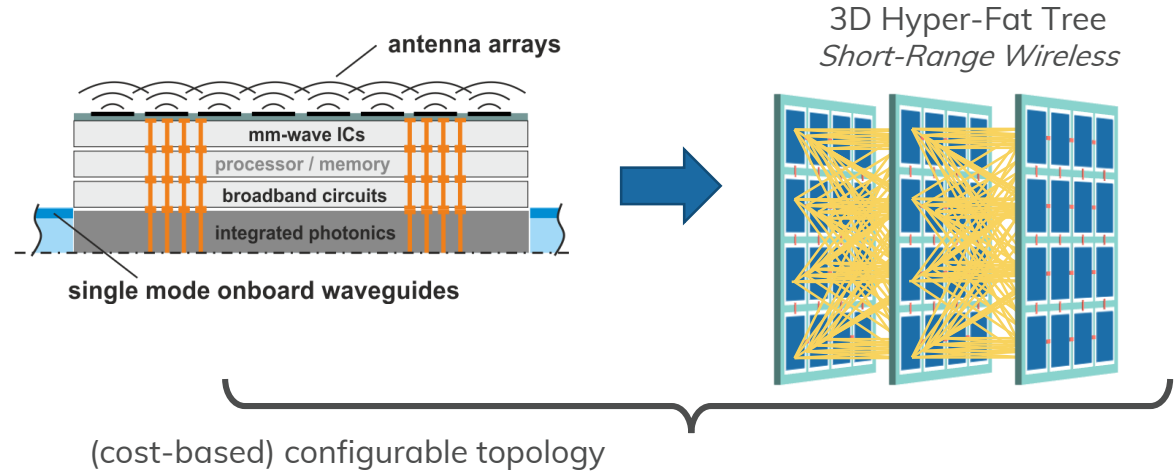
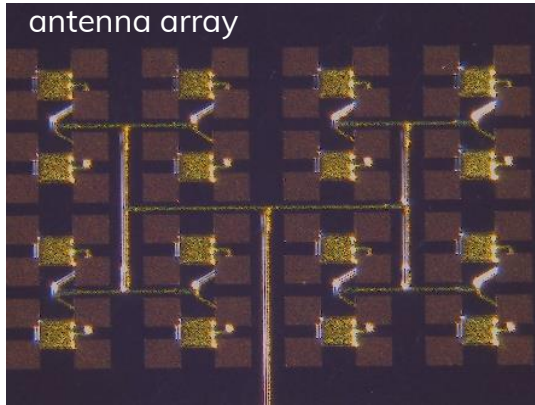
# Technology Advances in Network Technology (cross-board)



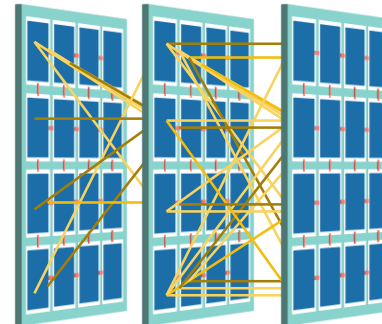
+ 3D stacking



# Technology Advances in Network Technology (cross-board)



Butler Matrix



?

# Dear Architects, I am sick of your shit.

**The 3 Vs !!!**

Annie Choi  
(An Open Letter)

*Once, a long time ago in the days of yore, I had a friend who was studying architecture to become, presumably, an architect.*

*This friend introduced me to other friends, who were also studying architecture. Then these friends had other friends who were architects - real architects doing real architecture like designing fancy condos that look a lot like glass slides. And these real architects knew other real architects and now the only people I know are architects. And they all design glass slides that I will never walk or live in and serve only to obstruct my view of New Jersey.*

*Do not get me wrong, architects, I like you as a person, I think you are nice, smell good most of the time, and I like your glasses. You have crazy hair, and if you are lucky, most of it is on your head. But I do not care about architecture. It is true. This is what I do care about.*

- burritos
- hedgehog
- coffee

# Design Space for DB Architectures

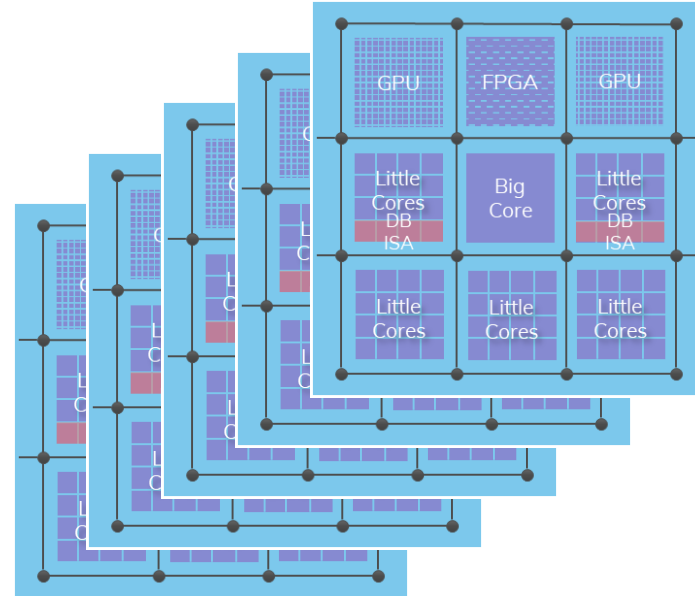


Volume  
(scalability)

- Single query vs. overall system performance
- Scheduling & data placement
- Concurrency control



Millions of cores?



# Design Space for DB Architectures



Volume  
(scalability)

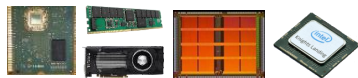
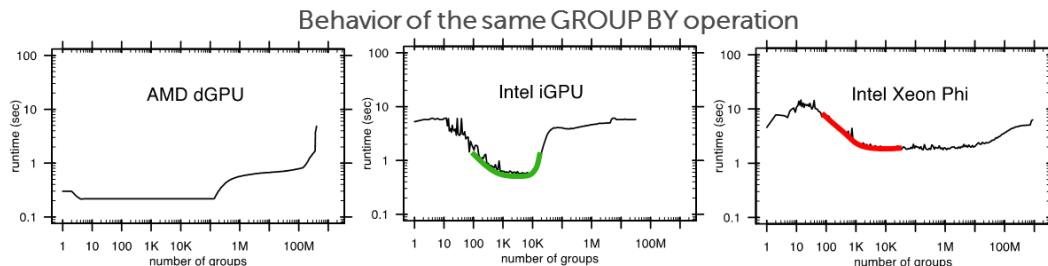
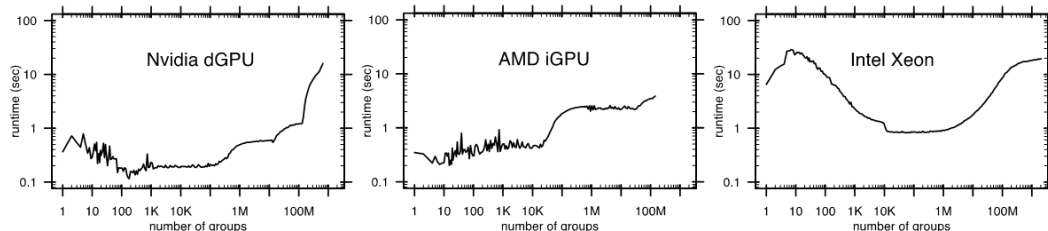
- Single query vs. overall system performance
- Scheduling & data placement
- Concurrency control



Millions of cores?



Scheduling a zoo?



Variety  
(heterogeneity)

- Impact on query optimization
- Impact on runtime
- Dealing with non-relational operators / application code

# Design Space for DB Architectures

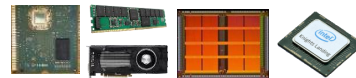
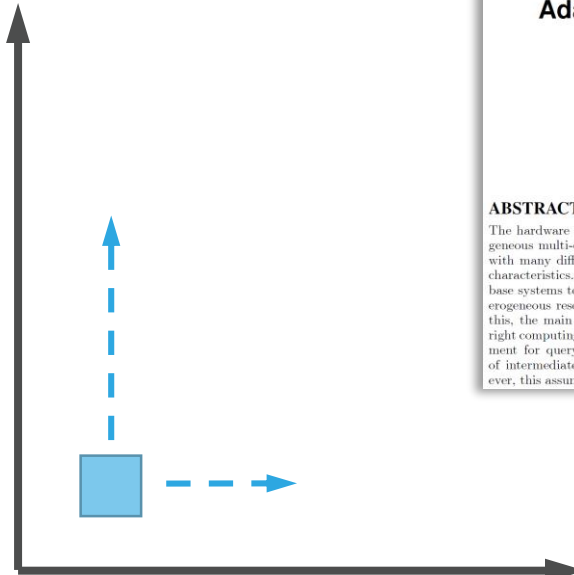


Volume  
(scalability)

- Single query vs. overall system performance
- Scheduling & data placement
- Concurrency control

 Millions of cores?

 Scheduling a zoo?



Variety  
(heterogeneity)

**Adaptive Work Placement for Query Processing on Heterogeneous Computing Resources**

Tomas Karnagel, Dirk Habich, Wolfgang Lehner  
Database Systems Group  
Technische Universität Dresden  
Dresden, Germany  
{first.last}@tu-dresden.de

**ABSTRACT**

The hardware landscape is currently changing from homogeneous multi-core systems towards heterogeneous systems with many different computing units, each with different characteristics. This trend is a great opportunity for database systems to increase the overall performance by utilizing heterogeneous resources more efficiently. However, this, the main challenge is to place the right computing unit. Current approaches for query processing assume that the cost of intermediate results can be correctly estimated. However, this assumption does not hold for complex queries. To

GPUs [12], the Xeon Phi [16], and FPGAs [23], showing great results for isolated workloads. However, for arbitrary workloads, the data placement and scheduling has to be adapted to the heterogeneous resources and in particular to the different computing units. Current optimization approaches for query processing assume that the cost of intermediate results can be correctly estimated. However, this assumption does not hold for complex queries. To

**Today at 2:00PM**

- Impact on query optimization
- Impact on runtime
- Dealing with non-relational operators / application code

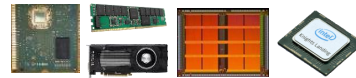
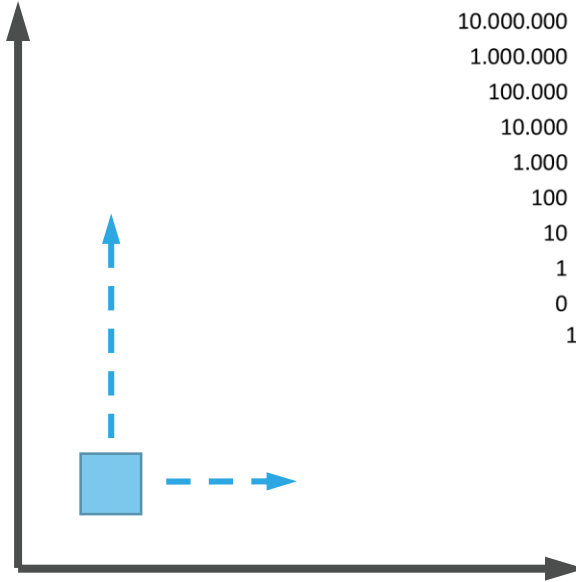


# Design Space for DB Architectures

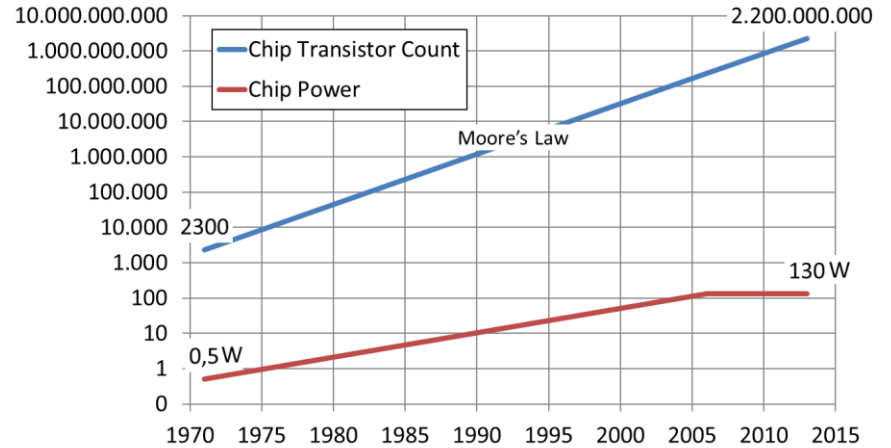


Volume (scalability)

- ⚠ Millions of cores?
- ⚠ Scheduling a zoo?
- ⚠ Energy Awareness?



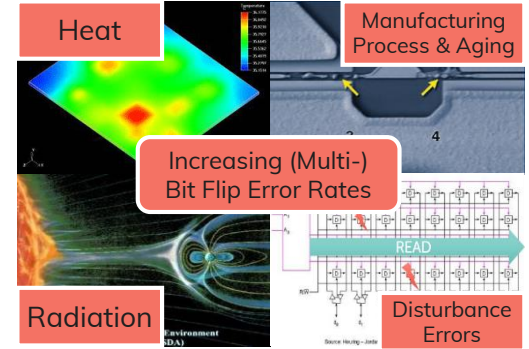
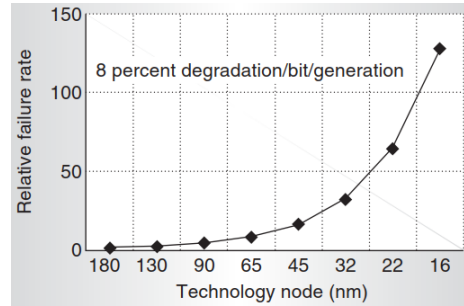
Variety (heterogeneity)



# Design Space for DB Architectures



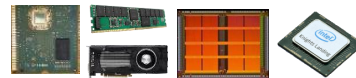
Volume  
(scalability)



⚠ Millions of cores?

⚠ Scheduling a zoo?

⚠ Energy Constraints?  
⚠ Resilience Constraints?



Variety  
(heterogeneity)

**Heterogeneous-Reliability Memory:  
Exploiting Application-Level Memory Error Tolerance**

Yixin Luo<sup>1</sup> Sriram Govindan<sup>1</sup> Bikash Sharma<sup>1</sup> Mark Santaniello<sup>1</sup> Justin Meza  
Aman Kansal<sup>1</sup> Jie Liu<sup>1</sup> Badridine Khessib<sup>1</sup> Kushagra Vaid<sup>1</sup> Onur Mutlu  
Carnegie Mellon University, {yxluo, srgovin, bsharma, marksan, kansal, jie.liu, bkhebbib, kvaid}@cmu.edu  
<sup>1</sup>Microsoft Corporation, {srgovin, bsharma, marksan, kansal, jie.liu, bkhebbib, kvaid}@microsoft.com

**1. Summary**  
Recent studies estimate that server cost contributes to as much as 57% of the total cost of ownership (TCO) of a data-center [1]. One key contributor to this high server cost is the procurement of memory devices such as DRAMs, especially for data-intensive datacenter cloud applications that need low errors in the field [40, 33, 41, 20, 27, 18, 36]. We wanted to design a framework to emulate the occurrence of a memory error in an application's data in a *controlled* manner. Second, we wanted an *efficient* way to measure how an application accesses its data. Third, we wanted our framework to be easily *adaptable* to other workloads or system configurations.

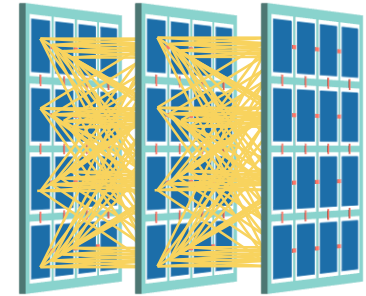
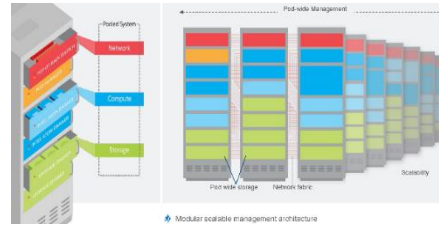
Krishna, Lohman, & Ram, 2013

# Design Space for DB Architectures



Volume  
(scalability)

*software-defined infrastructure*



Variability  
(reconfiguration @ runtime)



Millions of cores?



Scheduling a zoo?



Energy Constraints?  
Resilience Constraints?

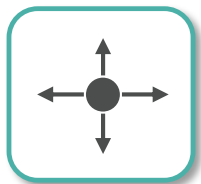


Variety  
(heterogeneity)

# Database Design Principles



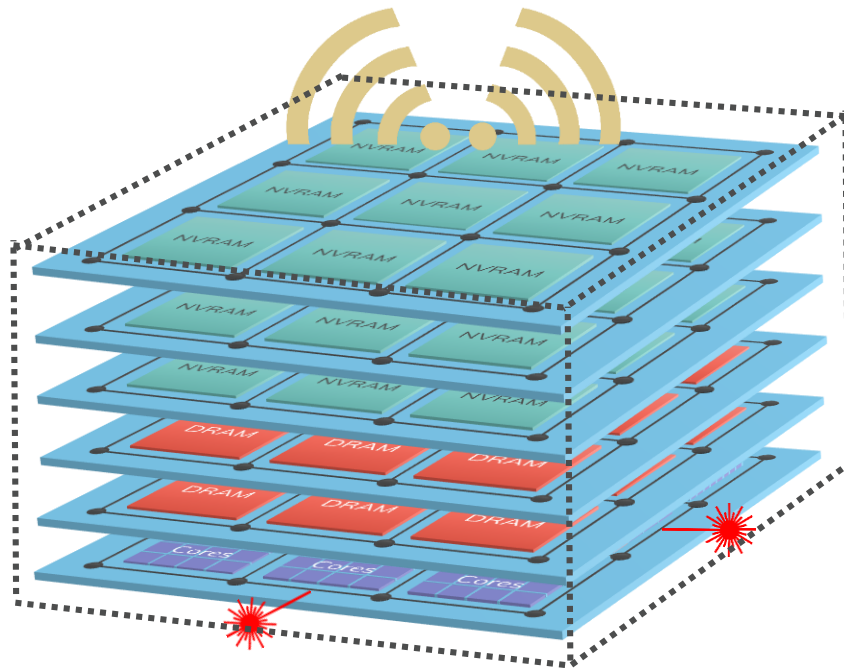
Data-Centric Design



Fine-Grained Adaptivity



Self-Adaptation

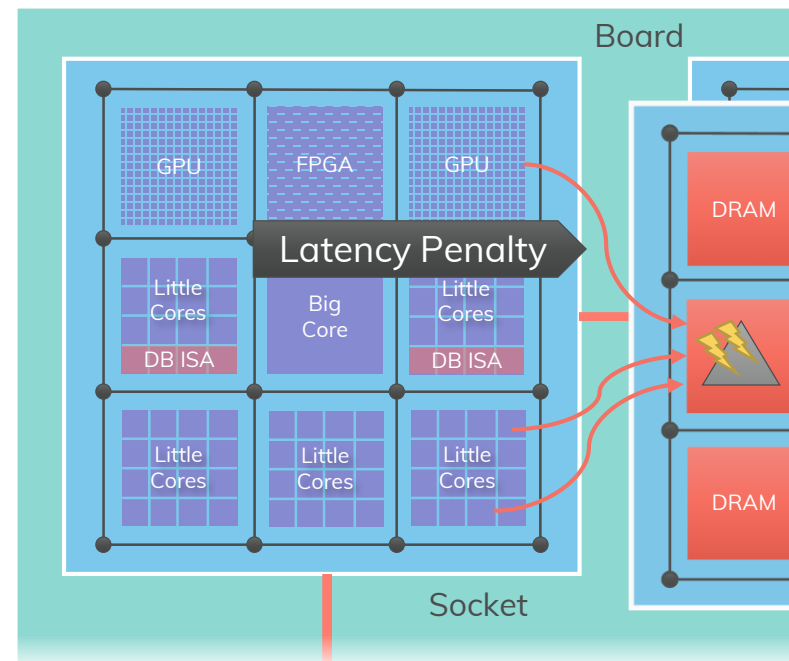
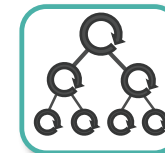
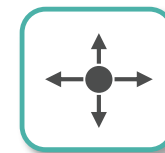
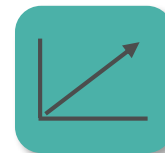
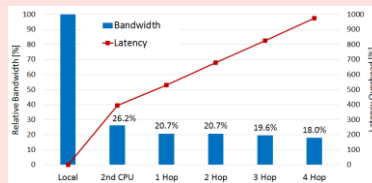


# DB Design Principle: Data Centric Architecture

## Scalability Limiters

### Transaction-Oriented Architecture

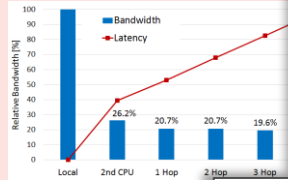
- ▶ Latches in Data Structures
- ▶ Remote Memory Accesses



# DB Design Principle: Data Centric Architecture

## Scalability Limiters

- ▶ Latches in Data Structures
- ▶ Remote Memory Accesses



## Transaction-Oriented Architecture

## Scalability Enablers

### Data-centric Architecture

- ▶ Factoring out Common Services
- ▶ Partitions as 1st-Class Citizens
- ▶ Hierarchical Communication

### Data-Oriented Transaction Execution

Ippokratis Pandis<sup>1,2</sup> Ryan Johnson<sup>1,2</sup> Nikos Hardavellas<sup>3</sup> Anastasia Ailamaki<sup>2,1</sup>  
ipandis@ece.cmu.edu ryanjohn@ece.cmu.edu nikos@northwestern.edu natassa@epfl.ch

<sup>1</sup>Carnegie Mellon University, Pittsburgh, PA, USA    <sup>2</sup>École Polytechnique Fédérale de Lausanne, Lausanne, VD, Switzerland    <sup>3</sup>Northwestern University, Evanston, IL, USA

### Adaptive NUMA-aware data placement and task scheduling for analytical workloads in main-memory column-stores

Iraklis Psaroudakis\* ‡ Tobias Scheuer‡ Norman May‡  
Abdelkader Sellami† Anastasia Ailamaki\*

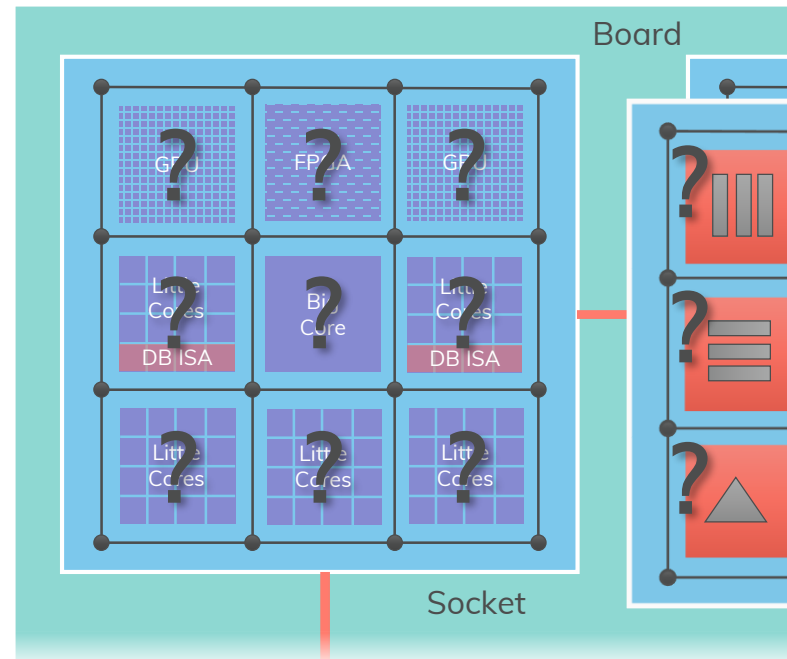
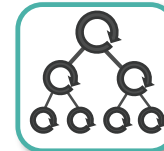
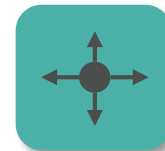
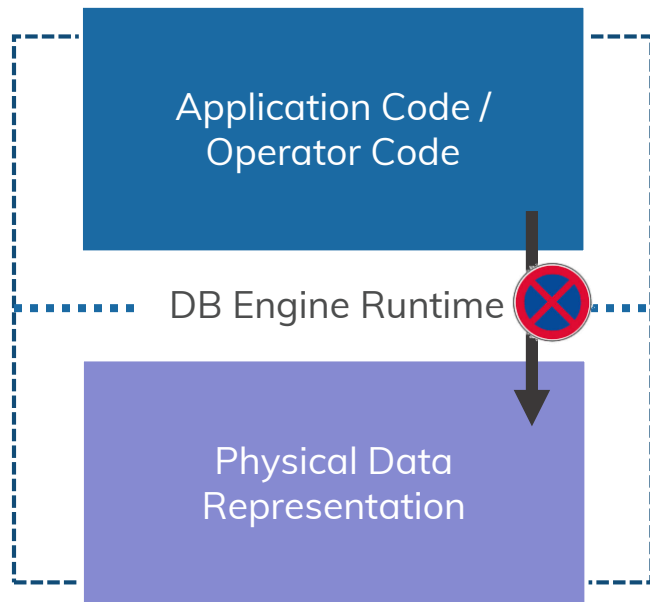
### Morsel-Driven Parallelism: A NUMA-Aware Query Evaluation Framework for the Many-Core Age

Viktor Leis\* Peter Boncz† Alfons Kemper\* Thomas Neumann\*

... and some more!

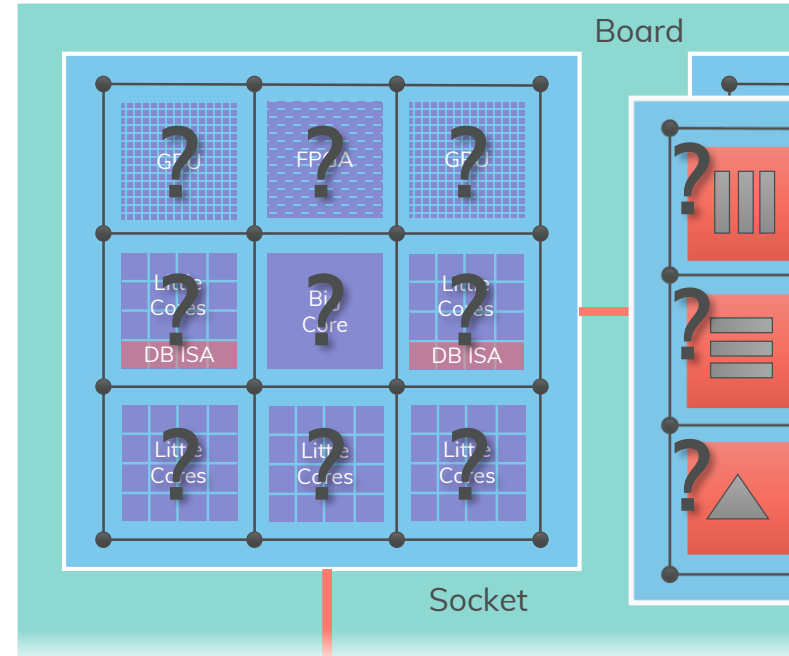
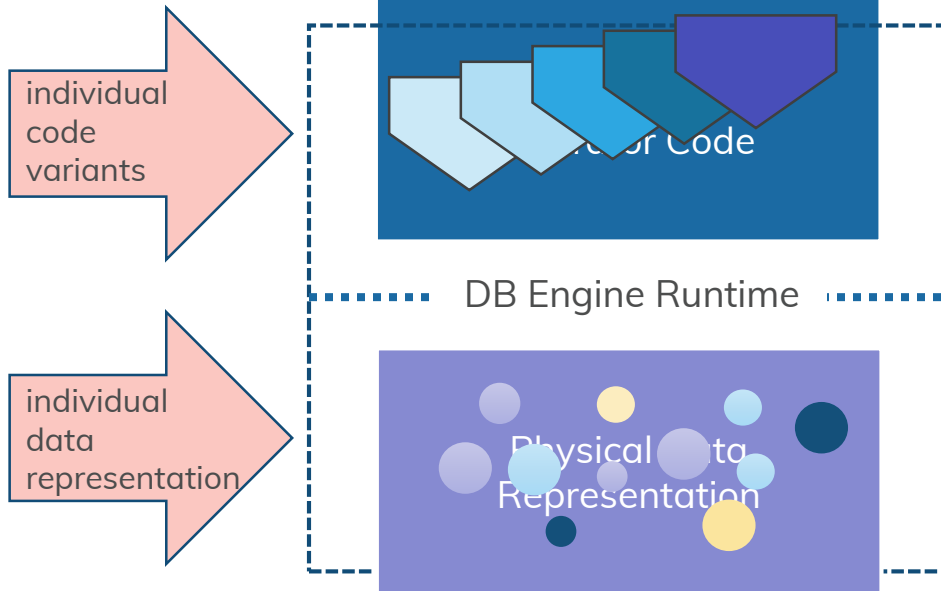
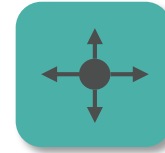
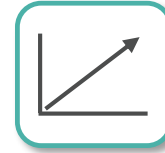
# DB Design Principle: **Fine Grained Adaptivity**

Clear abstraction between



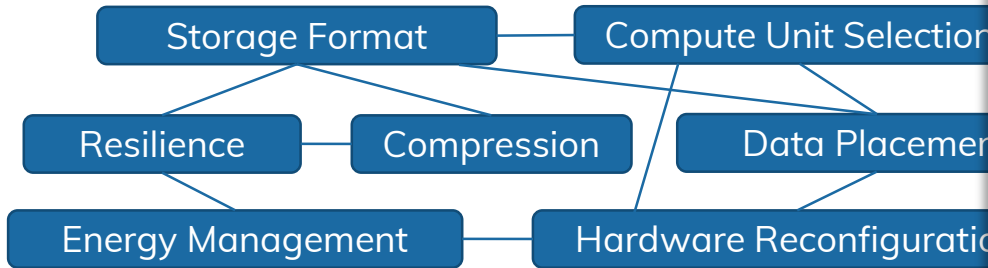
# DB Design Principle: **Fine Grained Adaptivity**

Clear abstraction between ... allows for





# DB Design Principle: System Adaptation



**What Is Peloton?**

- A self-driving SQL database management system.
- Integrated artificial intelligence components that enable autonomous optimization.
- Native support for byte-addressable non-volatile memory (NVM) storage technology.
- Lock-free multi-version
- Postgres network-protocol
- High-performance, lock
- 100% Open-Source (Ap



**What Problem Do**

During last two decades, re and physical design. This wo reactionary measures that fi

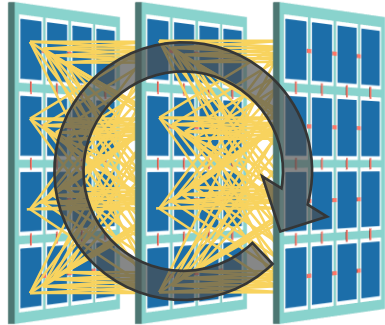
A new architecture is needed autonomous operations. This integrated planning compon trends which lets the system way, and reduces time taken of these systems has surpas

**About**

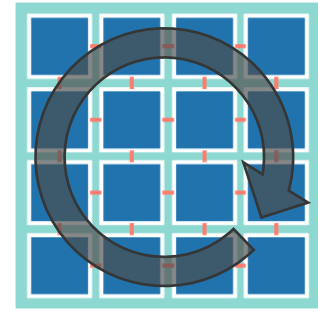
The goal of the ERIS project is to construct a highly adaptive in-memory storage engine that is able to scale-up on modern and future single-box server systems that expose massive NUMA effects. To achieve these goals, ERIS is designed in a modular way. The core component is the storage engine that is programmable via a low-level programming interface, which is based on C++ that is enhanced by specialized library functions. On top of this programming interface, users can run several domain-specific languages (DSL) such as the traditional declarative SQL. To cope with the variety of data formats of todays applications, ERIS allows to plug-in additional storage format modules to ensure that the data of different data models can be stored an processed efficiently.

Another central a to changing work engine to be elas efficiency as well collocate data and be exchanged at control loop, whi to adapt the sys

many more are needed!!!

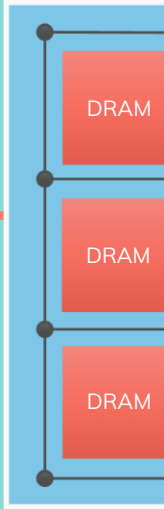


Box



Board

Board



Socket

... the End!

Hardware developments are pushing system software development

## OPPORTUNITIES ARE MANYFOLD

- We don't have a choice!
  - modern HW will be exploited for efficient data management  
→if not by „us“, then by other communities
- Extremely interesting research questions, but
  - „there is no free lunch“ still holds!
  - requires interdisciplinary research activities beyond DB system engine design

## RECAP: DISRUPTIONS ALWAYS REQUIRE TWO INGREDIENTS:

- Novel technology
- Novel types of DB(!!!) applications



Now we have both!!!

Now it's time for the next disruption!



# How Disruptive is Modern Hardware?

Wolfgang Lehner

Thanks!