How Disruptive is Modern Hardware?

Wolfgang Lehner
Interest in Modern HW?

...IN RESEARCH

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

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

Active'17
First International Workshop on Data Management on Virtualized Active Systems.
Interest in Modern HW?
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?
Disruptions in Data Management always require two ingredients

...novel technology and hardware

...novel types of DB(!!!) applications
Why was In-Memory Computing development disruptive?

→ enabler for HTAP = OLTP & OLAP
UseCase - Hybrid Transactional Processing

**Statistika**

- **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)

---

Material provided by Tim Crum (PayPal) and Frank Renkes (SAP SE)
Data Management System

“Traditional Apps“
Accounting, Reconciliation, and Reporting

Applications

Hardware
The DB Sandwich

“Traditional Apps”
Accounting, Reconciliation, 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.
What is the design space? / What might be a hypothetical HW blueprint?
Outline

Compute Units

Compute Unit Diversity
Compute Unit Diversity

Heterogeneous Computing by offloading „simple tasks“

**Focus on**

**Performance, performance, performance!!!**

- specialized data structures and algorithms
- parallel programming models and compiler support
- data and operator placement strategies
# Compute Unit Diversity

## Traditional CPU-World vs. Emerging Compute Unit Variety

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

Material provided by Norman May (SAP SE)


https://www.top500.org/featured/systems/asci-white-lawrence-livermore-national-laboratory/
**...but: we are hitting the „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?
**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

Index Scan

20%

Column Scan

40%

There is potential in energy saving without compromising performance!
Energy Savings

Query Load

Linux Governor

DB-controlled
Energy Awareness

How far can we push it?

Is this disruptive?

Figure 1: CPU performance trend for Android mobile devices.

Mohammad Shahrad, David Wentzlaff: Towards Deploying Decommissioned Mobile Devices as Cheap Energy-Efficient Compute Nodes. HotCloud 2017
HW/SW-CoDesign

Extended Tensilica LX5 Processor

- Instruction Set
  - Basic RISC Instruction Set
  - Application-Specific Instruction Set

- Register Files
  - Basic Registers
  - Application-Specific Registers
  - Application-Specific States

- Instruction fetch
- Load-Store Unit 0
- Load-Store Unit 1

Data Prefetcher
- Local Instruction Memory
- Local Data Memory 0
- Local Data Memory 1

Interconnect

- Offloading DB-specific instructions
- Energy efficient design for near-memory deployment
## Tomahawk DBA Primitives

<table>
<thead>
<tr>
<th>Bitmap Compression and Processing (AND, OR, XOR)</th>
<th>Hashing</th>
<th>Sorted Set Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primitives</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WAH</td>
<td>Hash + Lookup</td>
<td>Intersection</td>
</tr>
<tr>
<td>PLWAH</td>
<td>Hash + Insert</td>
<td>Union</td>
</tr>
<tr>
<td>COMPAX</td>
<td>Hash Keys</td>
<td>Difference</td>
</tr>
<tr>
<td></td>
<td>Hash Sampling</td>
<td>Sort-Merge Join</td>
</tr>
<tr>
<td></td>
<td>CityHash32</td>
<td>Sort-Merge Aggregation (SUM)</td>
</tr>
</tbody>
</table>

+ 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 (Cted)  
Technische Universität Dresden, Germany  
sebastian.haas@tu-dresden.de, gerhard.fettweis@tu-dresden.de
Tomahawk DBA: Sorted Set Intersection

Throughput (Million elements per second) vs. Selectivity (in %)

- DBA_2LSU_EIS w/ partial loading
- DBA_2LSU_EIS w/o partial loading
- DBA_1LSU_EIS w/ partial loading
- DBA_1LSU_EIS w/o partial loading
- DBA_1LSU
- 108Mini

Final processor:
- +1 Load-Store unit
- + Partial loading
- + Extended ISA

Data bus: 32->128 bit
### Tomahawk DBA: Sorted Set Intersection

<table>
<thead>
<tr>
<th></th>
<th>Intel i7-920</th>
<th>DBA_2LSU_EIS</th>
<th>~ ±x%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throughput (elements/s)</td>
<td>1,100 mio</td>
<td>1,203 mio</td>
<td></td>
</tr>
<tr>
<td>Clock frequency</td>
<td>2.67 GHz</td>
<td>0.41 GHz</td>
<td></td>
</tr>
<tr>
<td>Max. TDP</td>
<td>130 W</td>
<td>&gt;&gt; 0.135 W</td>
<td></td>
</tr>
<tr>
<td>Cores/Threads</td>
<td>4/8</td>
<td>1/1</td>
<td></td>
</tr>
<tr>
<td>Feature size</td>
<td>45 nm</td>
<td>65 nm</td>
<td></td>
</tr>
<tr>
<td>Area (logic &amp; memory)</td>
<td>263 mm²</td>
<td>&gt;&gt; 1.5 mm²</td>
<td></td>
</tr>
</tbody>
</table>

#### Relative Area Consumption (DBA_2LSU_EIS)

<table>
<thead>
<tr>
<th>Part</th>
<th>Area [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic Core</td>
<td>20.5</td>
</tr>
<tr>
<td>Decoding/Muxing</td>
<td>14.4</td>
</tr>
<tr>
<td>States</td>
<td>14.7</td>
</tr>
<tr>
<td>Op: All</td>
<td>11.3</td>
</tr>
<tr>
<td>Op: Intersection</td>
<td>6.8</td>
</tr>
<tr>
<td>Op: Difference</td>
<td>9.0</td>
</tr>
<tr>
<td>Op: Union</td>
<td>17.6</td>
</tr>
<tr>
<td>Op: Merge-Sort</td>
<td>5.7</td>
</tr>
<tr>
<td>SUM</td>
<td>100</td>
</tr>
</tbody>
</table>

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

Dark Silicon and the End of Multicore Scaling

Hadi Esmaeilzadeh, Emily Blem, Renée St. Amant, Karthikeyan Sankaralingam, Doug Burger

University of Washington, University of Wisconsin-Madison, The University of Texas at Austin, Microsoft Research

hadi@cs.washington.edu, blem@cs.wisc.edu, stamant@cs.utexas.edu, karu@cs.wisc.edu, dburger@microsoft.com
Tomahawk DBA: Sorted Set Intersection

<table>
<thead>
<tr>
<th></th>
<th>Intel i7-920</th>
<th>DBA_2LSU_EIS</th>
<th>~ ±x%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throughput (elements/s)</td>
<td>1,100 mio</td>
<td>1,203 mio</td>
<td></td>
</tr>
<tr>
<td>Clock frequency</td>
<td>2.67 GHz</td>
<td>0.41 GHz</td>
<td></td>
</tr>
<tr>
<td>Max. TDP</td>
<td>130 W</td>
<td>&gt; &gt; 0.135 W</td>
<td></td>
</tr>
<tr>
<td>Cores/Threads</td>
<td>4/8</td>
<td>1/1</td>
<td></td>
</tr>
<tr>
<td>Feature size</td>
<td>45 nm</td>
<td>65 nm</td>
<td></td>
</tr>
<tr>
<td>Area (logic &amp; memory)</td>
<td>263 mm²</td>
<td>&gt; &gt; 1.5 mm²</td>
<td></td>
</tr>
</tbody>
</table>

Relative Area Consumption (DBA_2LSU_EIS)

<table>
<thead>
<tr>
<th>Part</th>
<th>Area[%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic Core</td>
<td>20.5</td>
</tr>
<tr>
<td>Decoding/Muxing</td>
<td>14.4</td>
</tr>
<tr>
<td>States</td>
<td>14.7</td>
</tr>
<tr>
<td>Op: All</td>
<td>11.3</td>
</tr>
<tr>
<td>Op: Intersection</td>
<td>6.8</td>
</tr>
<tr>
<td>Op: Difference</td>
<td>9.0</td>
</tr>
<tr>
<td>Op: Union</td>
<td>17.6</td>
</tr>
<tr>
<td>Op: Merge-Sort</td>
<td>5.7</td>
</tr>
</tbody>
</table>
Summary - Compute Unit Diversity

Traditional CPU-World
- Multi-Core
- Multi-Sockets

Emerging Compute Unit Variety
- BigLITTLE
- dGPU/APU
- Xeon Phi
- FPGA
- ASIC

Opportunities
- Performance through parallelism

Challenges
- Life cycle
- Energy Awareness
- Programming Model
- Increased system complexity
- ...
Memory Diversity
Memory Diversity

- **SRAM**
  - Latency: 1X
  - Capacity: 1X

- **DRAM**
  - Latency: 10X
  - Capacity: 100X

**Types of Memory:***
- Caches
- MCDRAM
- DRAM DIMM
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

Material provided by David Wang (Samsung)
Memory Diversity

<table>
<thead>
<tr>
<th>Caches</th>
<th>MCDRAM</th>
<th>DRAM DIMM</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRAM</td>
<td>DRAM</td>
<td></td>
</tr>
<tr>
<td>1X</td>
<td>10X</td>
<td></td>
</tr>
<tr>
<td>1X</td>
<td>100X</td>
<td></td>
</tr>
</tbody>
</table>

Latency

Capacity

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

Caches | MCDRAM | DRAM DIMM | 3D XPoint | FRAM | MRAM | PRAM

SRAM | DRAM | NVRAM

1X | 10X | 100X

Latency

1X | 100X | 1000X

Capacity

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
### 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

---

### Table: Comparison of Memory Technologies

<table>
<thead>
<tr>
<th></th>
<th>MRAM</th>
<th>DRAM</th>
<th>PCM</th>
<th>ReRAM</th>
<th>TLC NAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost per Bit</td>
<td>~5</td>
<td>1</td>
<td>&gt;0.5</td>
<td>&gt;0.5</td>
<td>0.05</td>
</tr>
<tr>
<td>Read Latency</td>
<td>~1</td>
<td>1</td>
<td>10</td>
<td>100</td>
<td>1000</td>
</tr>
<tr>
<td>Write Latency</td>
<td>~1</td>
<td>1</td>
<td>50</td>
<td>1000</td>
<td>10000</td>
</tr>
<tr>
<td>Volatility</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Endurance</td>
<td>&gt;1E15</td>
<td>&gt;1E16</td>
<td>1E6</td>
<td>1E6</td>
<td>1E3</td>
</tr>
<tr>
<td>Write Energy (J/bit)</td>
<td>0.1~100</td>
<td>1</td>
<td>0.1~10</td>
<td>0.1~10</td>
<td>10</td>
</tr>
</tbody>
</table>

---

### Estimates provided by David Wang (Samsung)
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
(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

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

**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

What is in NVM?

<table>
<thead>
<tr>
<th>m_size</th>
<th>m_array</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Corrupt!</td>
</tr>
<tr>
<td>0</td>
<td>2017</td>
</tr>
<tr>
<td>1</td>
<td>2017</td>
</tr>
</tbody>
</table>

À la software transactional memory

à la software transactional memory

pmem.io
Persistent Memory Programming

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!
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

Material provided by Ismail Oukid (SAP SE)
Persistent Memory Leaks

Novel class of memory leaks resulting from failures

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

```c
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);
}
```

Failure-induced persistent memory leak!
Persistent Memory Leaks

Novel class of memory leaks resulting from failures in persistent storage.

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

```c
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!

New types of bugs and additional testing overhead.
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
Network Diversity
Network Diversity

- Ring Network
- Fully Connected
- Fat Tree
- Infiniband, etc.

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

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

Network vs. Memory Bandwidth:

⇒ Network bandwidth is not a bottleneck anymore (Latency is still 10x higher for remote access)
Example: Distributed Radix Join

Workload:
- Data: 512M records per table x 2 on 4 servers (1Gb Ethernet + FDR 4x IB)

Listing 1: Remote Radix-Partitioning

```python
for each tuple r in R do{
    h = radix-hash(key(r));
    buffer[h].append(r);
    if(buffer.isFull()){
        copy_counter[h]++;
        if(partition[h] is local){
            memcpy(buffer[h], partition[h]);
        } else{
            signalled = (counter[h]==N);
            RDMA_WRITE(buffer[h], partition[h], signalled);
        }
    } else{
        buffer.empty();
    }
}
```
Example: Memory Extensions

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

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 Djürken  Hasso Plattner
Matthias Uflacker  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

What is next?

Is this disruptive?
Network Diversity

- On-Chip: Fully Connected, Infiniband, etc.
- On-Board: Fully Connected, Fat Tree
- Cross-Board: Infiniband, etc.
- Cross-Node: Fully Connected, Fat Tree

- Ring Network: Haswell-EP
- 2D Mesh: Skylake-X, 3D Mesh, 2D Mesh Photonics
- HP The Machine
- Intel RSA Rack-scale architecture
Technology Advances in Network Technology (cross-board)

Antenna array

(2.5mm x 3.0mm) + 3D stacking

mm-wave ICs

processor / memory

broadband circuits

integrated photonics

single mode onboard waveguides
Technology Advances in Network Technology (cross-board)

antenna array

antenna arrays

mm-wave ICs
processor / memory
broadband circuits
integrated photonics

single mode onboard waveguides

(cost-based) configurable topology

3D Hyper-Fat Tree
Short-Range Wireless

Butler Matrix
Dear Architects,

I am sick of your shit.

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 also architects. Now all these architects were doing was designing fancy clothes that look a bit like glass dishes. And these real architects knew other real architects and now the only people I know are architects. And they all design glass dishes that I will never own or live in and serve only to demoralize my view of New Jersey.

Do not get me wrong, architects. I like you as a person. I think you are nice, smart, good most of the time, and I like your glasses. You have lovely 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:

- burntos
- hedgehog
- coffee

Annie Choi
(An Open Letter)
Design Space for DB Architectures

- 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

- **Variety** (heterogeneity)
  - Impact on query optimization
  - Impact on runtime
  - Dealing with non-relational operators / application code

- Millions of cores?
- Scheduling a zoo?
### Design Space for DB Architectures

- **Volume (scalability)**
  - Single query vs. overall system performance
  - Scheduling & data placement
  - Concurrency control

- **Variety (heterogeneity)**
  - Impact on query optimization
  - Impact on runtime
  - Dealing with non-relational operators / application code

- ** Millions of cores? **
- ** Scheduling a zoo? **

---

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

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

**ABSTRACT**

The hardware landscape is currently characterized by heterogeneous multicomputer systems that are equipped with many different computing units of different characteristics. This trend is expected to continue. How can we best utilize the computing power of a system to improve the overall performance? One key to answer this question is to design a data placement strategy that optimizes the performance of the database.

---

**Today at 2:00PM**
Design Space for DB Architectures

Volume (scalability)

⚠️ Millions of cores?

⚠️ Scheduling a zoo?

⚠️ Energy Awareness?

Variety (heterogeneity)

Chip Transistor Count vs. Chip Power over years.
Design Space for DB Architectures

Volume (scalability)

⚠️ Millions of cores?

⚠️ Scheduling a zoo?

⚠️ Energy Constraints? Resilience Constraints?

Variety (heterogeneity)

Heat

Manufacturing Process & Aging

Increasing (Multi-) Bit Flip Error Rates

Radiation

Disturbance Errors

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

Yizhe Liu, Sozhen Gogia, Bhashkar Sharma, Mark Samuelson, Justin Mora, Ahmad Kouril, Ji Liu, Badriprasad Bhalodia, Kasugra Yadav, Omar Muttu
Carnegie Mellon University, yizheliu@cs.cmu.edu, jliu, ms@cmu.edu
Microsoft Corporation, jkouril, bahsne, mbhalodia, kandyav, omarm@live.com

1. Summary
Recent analysis estimates that server-cost contributions that exceed 35% of the total cost of ownership (TCO) of a datacenter [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-error rates. Errors in the field [33, 31, 20, 27, 34, 36], the need to design a framework to control the occurrence of memory errors in an application's data, and the limited work done to date to address the issue. Second, we wanted an approach to measure how an application actually uses the data. Third, we wanted our framework to be easily adaptable to other workloads or system configurations.
Design Space for DB Architectures

- Volume (scalability)
- Variability (reconfiguration @ runtime)
- Variety (heterogeneity)
- Millions of cores?
- Scheduling a zoo?
- Energy Constraints?
- Resilience Constraints?

software-defined infrastructure
Database Design Principles

- Data-Centric Design
- Fine-Grained Adaptivity
- Self-Adaptation
DB Design Principle: **Data Centric Architecture**

**Scalability Limiters**

- Latches in Data Structures
- Remote Memory Accesses

**Transaction-Oriented Architecture**

- Bandwidth
- Latency

**Latency Penalty**

- GPUs
- FPGAs
- Little Cores
- Big Core

**Board**

- DRAM

**Socket**
DB Design Principle: **Data Centric Architecture**

**Scalability Limiters**
- Latches in Data Structures
- Remote Memory Accesses

**Scalability Enablers**
- Data-centric Architecture
- Factoring out Common Services
- Partitions as 1st-Class Citizens
- Hierarchical Communication

**Data-Oriented Transaction Execution**
- Ippokratis Pandis\(^1,2\)
ipandis@ece.cmu.edu
- Ryan Johnson\(^1,2\)
ryanjohn@ece.cmu.edu
- Nikos Hardavellas\(^3\)
nikos@northwestern.edu
- Anastasias Ailamaki\(^2,1\)
natassa@epfl.ch

\(^1\)Carnegie Mellon University
\(^2\)Ecole Polytechnique Fédérale de Lausanne
\(^3\)Northwestern University

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

**Morsel-Driven Parallelism: A NUMA-Aware Query Evaluation Framework for the Many-Core Age**
- Viktor Leis* Peter Boncz† Alfons Kemper* Thomas Neumann*
  *Technische Universität Dresden
  †IBM

... and some more!
DB Design Principle: **Fine Grained Adaptivity**

Clear abstraction between

Application Code / Operator Code

DB Engine Runtime

Physical Data Representation
DB Design Principle: **Fine Grained Adaptivity**

Clear abstraction between ... allows for

- Individual code variants
- Individual data representation

**Physical Data Representation**

**DB Engine Runtime**

**Application Code / Operator Code**
DB Design Principle: System Adaptation

- Storage Format
- Compute Unit Selection
- Resilience
- Compression
- Data Placement
- Energy Management
- Hardware Reconfiguration

PELOTON

What Is Peloton?
- A highly efficient SQL database management system
- Optimizes access to data
- Supports advanced query optimization
- Integrated artificial intelligence component that can enable autonomous optimization
- Native support for low addressable memory (LAM) storage technology

Many more are needed!!!
... the End!
Conclusion

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!