The Data Center under your Desk – How Disruptive is Modern Hardware for DB System Design?

Wolfgang Lehner
Technische Universität Dresden
01062 Dresden, Germany
wolfgang.lehner@tu-dresden.de

ABSTRACT

While we are already used to see more than 1,000 cores within a single machine, the next processing platforms for database engines will be heterogeneous with built-in GPU-style processors as well as specialized FPGAs or chips with domain-specific instruction sets. Moreover, the traditional volatile as well as the upcoming non-volatile RAM with capacities in the 100s of TBytes per machine will provide great opportunities for storage engines but also call for radical changes on the architecture of such systems. Finally, the emergence of economically affordable, high-speed/low-latency interconnects as a basis for rack-scale computing is questioning long-standing folklore algorithmic assumptions but will certainly play an important role in the big picture of building modern data management platforms. In this talk, we will try to classify and review existing approaches from a performance, robustness, as well as energy efficiency perspective and pinpoint interesting starting points for further research activities.

1. INTRODUCTION

Traditional database system design is based on hardware assumptions that were valid 30 years back. Conceptually, that includes a working area with CPU and associated (transient) DRAM plus some sort of stable storage (disk) and networking capabilities to communicate with remote engines. In order to hide the low bandwidth and high latency as much as possible, the database system explicitly requires control over the individual interactions with these "external devices". A decade back, main-memory centric approaches changed the data hierarchy by emphasizing cache-awareness of algorithms and data structures, optimizing the relationship between CPU (i.e. CPU caches) and DRAM. By relying on large (and expensive) main memory capacities, main-memory centric approaches overcome many limitations, but still assume the traditional hardware model with disk IO for logging and an explicit network protocol for distributed query processing.

Proceedings of the VLDB Endowment, Vol. 10, No. 12

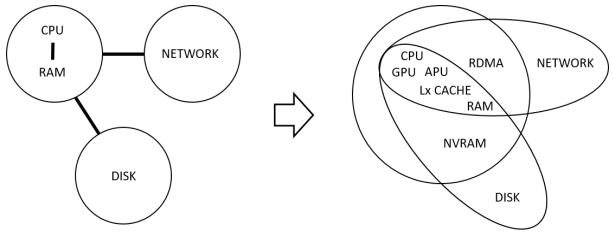
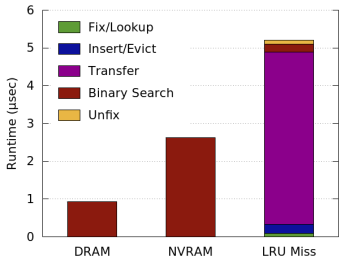


Figure 1: Change of the HW Design Space

However, especially with the advent of rack-scale computing platforms, the design space of the underlying hardware environment has changed dramatically in multiple directions. For example, non-volatile RAM (NVRAM) allows direct access on stable storage in a byte-addressable manner, blurring the separation of the (transient) working copy and the (persistent) state of the database. In the same way, techniques like RDMA cause to blur the boundaries of individual nodes by allowing direct access to the main memory of other nodes in a cluster.

Moreover, significant progress has been made with respect to processing units. After the wave of multi-cores currently resulting in machines with more than 1,000 general purpose cores (e.g. HPE SGI UV300), we will be faced with potentially wildly heterogeneous computing units specialized on individual tasks. On the GPU side, we currently see – on the one hand – a move towards more standardized memory models as a step towards general programing models (e.g. current NVIDIA Volta [1]) going hand-in-hand with extremely efficient communication paths (e.g. NVLink). On the other hand, we have APUs – GPU and regular CPU on the same die – as tightly connected system overcoming PCI-bus induced data transfer limitations. Within the domain of FPGAs – a platform considered extremely beneficial for compute-intensive applications but highly neglected for data-intensive systems – we already have boards providing the programmable logic area in combination with multiple ARM cores to allow offloading of complex algorithmic snippets (e.g. [2]). We also see tightly integrated systems such as those within the Intel-Altera Heterogeneous Architecture Research Platform Program (HARP), consisting of a regular XEON CPU with an FPGA on the same die.

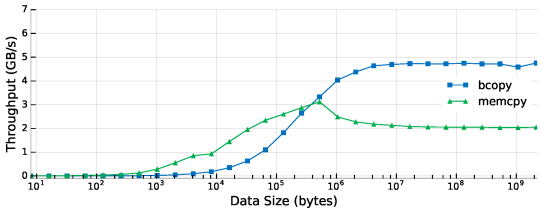
The plethora of opportunities is finally extended by integrating domain-specific instruction sets into regular processors in order to exploit the "Dark Silicon" effect (e.g. [3]) so far mostly dominated by text and image processing scenarios. Considering all these developments in the hardware sector, the overall question has to be asked:



(a) Caching Policies for NVRAM

distance	bandwidth (GB/s)	latency (ns)
local	36.2	81
2nd processor	9.5	400
1 hop NUMALink	7.5	505 - 515
2 hop NUMALink	7.5	625 - 635
3 hop NUMALink	7.1	745 - 755
4 hop NUMALink	6.5	870

(b) Severe NUMA effects



(c) Data Movement in NUMA Machines

Figure 2: Opportunities and Impact of Modern Hardware Developments

How Disruptive is Modern Hardware for DB System Design?

Instead of providing a comprehensive answer to this question, let us give two examples of how modern hardware may have an impact on the design of database data structures and algorithms.

Example 1: Revisiting Buffer Management

The advent of non-volatile RAM has the potential to revolutionize the way storage systems are designed to provide efficient access and consistency at the same time [4]. A very obvious way of using NVRAM is to treat it as a super fast disk by defining a file system (like tmpfs) over NVRAM regions and bypassing the OS page cache, e.g. using Direct Access (DAX) support in ext4fs in Linux, and running the database system as it is. Since storage engines usually implement their own caching strategy, the system might interestingly get slower compared to operating directly on NVRAM and completely ignoring the existence of an intermediate cache. In contrast to a traditional cache behavior with a "Hit" (DRAM) or "Miss" (HDD/SSD/...), NVM can be directly accessed by the CPU and therefore add a new dimension to replacement strategies: "Miss-Without-Transfer", which is significantly cheaper than a regular "Miss" (including a read and a write). Figure 2a shows the results of a micro benchmark running a search on 4k blocks assuming 4x the latency of NVRAM compared to regular DRAM. The right bar in figure 2a shows the runtime using a regular LRU strategy. In this setting, a "Miss" is 2x more expensive than executing the binary search directly in NVRAM without copying the data into lower latency DRAM before accessing. Obviously, hierarchical memory scenarios in NUMA environments or configurations with memory regions accessible via RDMA increase the complexity even more.

Example 2: Overcoming NUMA Effects

Large machines are based on local memory attached to individual processors which again are connected by an intra-machine network (e.g. Intel QPI) resulting in a NUMA-style memory model. Although a global and cache-coherent virtual memory is provided for the individual applications, significant communication overhead within a single machine may result in severe overall performance impact. As can be seen in figure 2b for a HPE SGI UV2000, such NUMA effects may result in severe bandwidth and latency penalties, if data access is "remote", e.g. ends up in the memory of a processor "at the other end" of the machine. While preserving data locality is a well-studied problem, modern hardware may bring in an additional dimension for a cost-based data placement decision: For example, the HPE SGI-series provides an ASIC (HARP) to connect individual processors, maintaining cache coherency and providing a common address space across different processors. Additionally, the HARP employs a Global Reference Unit (GRU) facilitating a proprietary API to accelerate and thereby offload memory operations. Unfortunately, as can be seen in figure 2c, the use of the GRU internal `gru_bcopy()` mechanism to transfer memory chunks asynchronously and outside of the processor requires a decision regarding the traditional `memcpy()`-approach with respect to data chunk size, communication path, latency optimization goal, and/or regarding the ability to use the freed-up processor capacity for other jobs.

2. SUMMARY AND CONCLUSIONS

While these examples do not reflect all of the opportunities and challenges of modern hardware components and platforms, they may serve to spark the discussion on **"how disruptive is modern hardware for DB system design"**. Within the talk, we will therefore specifically discuss approaches already exploiting modern hardware capabilities from an academic research as well as commercial application point of view. For some cases, simply throwing an existing system on top of new hardware and expecting it to run faster can lead to disappointment.

Acknowledgments

A big "Thank you" goes to my research group at TU Dresden – they are challenging and driving me every day with great ideas and interesting problems. I also want to explicitly thank the whole SAP HANA data management team in Walldorf, Seoul, and Waterloo providing me with the opportunity to push database research in the context of a commercial data management platform.

3. REFERENCES

[1] NVIDIA Volta. https://www.nvidia.com/en-us/data-center/volta-gpu-architecture/.

[2] UltraScale. https://www.xilinx.com/products/silicon-devices/soc/zynq-ultrascale-mpsoc.html.

[3] H. E. e.a. Dark silicon and the end of multicore scaling. In *ISCA 2011*, pages 365–376, 2011.

[4] S. M. e.a. A survey of software techniques for using non-volatile memories for storage and main memory systems. *IEEE Transactions on Parallel and Distributed Systems*, 27(5):1537–1550, 2016.