First Things First: A Discussion of Modelling Approaches for Disruptive Memory Technologies

Michael Müller
michael.mueller@uni-osnabrueck.de
Osnabrück University
Osnabrück, Germany

Daniel Kessener
dkessenerk@uni-osnabrueck.de
Osnabrück University
Osnabrück, Germany

Olaf Spinczyk
olaf.spinczyk@uni-osnabrueck.de
Osnabrück University
Osnabrück, Germany

ABSTRACT

Disruptive memory technologies (DMT) are dramatically changing the architecture of modern computer systems and affect important design decisions throughout the whole software stack. For their success it is crucial that developers of system software and applications find ways to fully exploit the potential of the novel hardware. Having appropriate DMT hardware models is the key to successful optimization in the world of system software and application development. Therefore, this paper introduces four relevant new DMTs and corresponding performance analyzes as well as modelling approaches. We conclude with the insight that there is a lack of system-wide models that are practically applicable by system software designers for proper optimization and, thus, an important domain for future research.

KEYWORDS

System software design, Non-Volatile Memory (NVM), Remote DMA (RDMA), Near-Memory Computing (NMC), Processing in Memory (PIM), High-Bandwidth Memory (HBM), System Knowledge Base, hwloc, MCTOP, NUMA topology

1 INTRODUCTION

The last decade saw a significant change in computer architectures and technologies. Today, multicore processors with non-uniform memory access (NUMA) have become the most common architecture for servers and mainframes with increasing heterogeneity. For a few years, new memory technologies, such as persistent memory, remote direct-memory-access, high-bandwidth memory, and near-memory computing, arose, which will transform the way memory and storage are about to be used in the computing systems of the future. These disruptive memory technologies raise new challenges for developers of applications and system software alike.

FGBS '21, September 21–22, 2021, Trondheim, Norway
© 2021 Copyright held by the authors.
https://doi.org/10.18420/fgbs2021h-02

As the transition from single processors with uniform memory access and simple memory hierarchies to complex NUMA architectures has already shown, exact knowledge of the underlying hardware topology is crucial to ensure performance and efficiency in today's computing systems [1–3, 5, 9]. However, it has also turned out that using this knowledge in the design of software systems is a double-edged sword, as one gains better performance and efficiency at the cost of portability. With the emerging disruptive memory technologies, one can see the same kind of development. Again, applications are tailored to specific hardware configurations at the price of portability.

Nevertheless, most developers are lured into a trap by faulty assumptions about the overall system. So a common assumption of developers of high-performance computing (HPC) and database software is that their application has exclusive usage of all hardware resources and that these resources are not subject to change during the lifetime of this software. Although this might be true for specialized applications in HPC, it is not for the data center, which exhibits a wide variety of applications, even whole virtual machines being deployed and assigned resources based on their actual utilization. In such systems the assumption that there is only one application running at a time is false, and the concurrency and interference of applications have to be taken into account, to ensure performance.

Thus, system software has to be in charge to manage disruptive memory technologies among concurrent applications and even virtual machines. However, to enable system software to make informed decisions regarding resource assignment, it needs a sophisticated model of the hardware resources to be managed that fulfills the following requirements. A sufficient system model

(1) should represent the overall hardware topology, including all disruptive memory technologies,
(2) should provide information about communication cost when moving data within the topology,
(3) should enable system software to update the model with monitoring results during runtime,

(4) should have a whole system view on all available resource and their allocations by all applications and system services
(5) and, based on the upper four, should provide performance predictions and compute optimized resource mappings that can be used by system software to improve performance and system load and reduce interference between concurrent applications

In the following section, we will present current state-of-the-art models for DMT and discuss their limitations. After that, we will describe existing system models for whole system optimization in Section 3 before concluding our paper in Section 4.

2 MODELS FOR DMT

Most studies on DMT focus on evaluating specific aspects of their respective technologies, such as performance analysis, system software design, programming models, or optimizations for specific hardware or applications. There has not been a concerted effort to develop a full-fledged software model capable of in-depth hardware simulation and application evaluation for any DMT. However, the studies examined in this paper provide a good foundation upon which such models could be developed.

2.1 Non-volatile Memory

While some performance models do exist, they only model very specific technologies, such as Optane [12, 28] and lack the ability to evaluate other technologies. Pohl et al. [21] introduced a cost model for optimizing PMEM access for data-stream processing, but this model is not suitable for applications other than stream processing. Some programming models aim to support developers seeking to take advantage of PMEM in their applications. George et al. [10] explore native support for persistent memory when developing in Go, Intel's *Persistent Memory Development Kit* (PMDK) [23] aims to supply developers with a development kit to ease the move to persistent memory and Köppen et al. [15] provides building blocks for the development of persistent data structures.

2.2 Remote DMA

MacArthur et al. [18] have done a performance study evaluating different RDMA programming techniques, aiming to provide a guide showcasing "best practice" RDMA programming decisions. On a similar note, Kalia et al. [13] have compiled design guidelines for high performance RDMA systems. Nelson et al. [20] have performed a study on the consequences of combining *Non-Uniform Memory Access* (NUMA) with RDMA, concluding that applications should be designed around both technologies to avoid steep performance penalties. Wei et al. [29] provide optimization hints for developers working with mixed RDMA and NVM systems while Shen et al. [25] and Qiu et al. [22] analyzed scheduling problems that arise with RDMA and the latter proposed and implemented a management model to improve native RDMA.

2.3 Near-Memory Computing

Singh et al. [26] surveyed the current state-of-the-art NMC technologies, identifying key challenges for future research efforts. Hsieh et al. [11], Mutlu et al. [19] and Khan et al. [14] have all researched techniques and practices to facilitate instruction offloading to a dedicated GPU to alleviate memory bandwidth bottlenecks. Singh et al. [27] introduce NAPEL, a high-level performance and energy estimation framework for NMC architectures, utilizing machine learning techniques. But each application to be evaluated with NAPEL requires extensive training data that takes a considerable amount of time to collect. NAPEL is also not suitable for applications using online scheduling algorithms. Corda et al. [6] introduce NMPO, a high-level framework capable of predicting NMC offloading suitability. It also utilizes machine learning. Lee et al. [17] developed a framework that allows for data-intensive database operations to be offloaded to near-memory computation engines and implemented a Proof-of-Concept. Corda et al. [7] identified metrics for characterizing NMC performance, which may aid in the development of performance models. Hsieh et al. [11] also proposes mechanisms for automated offloading and mapping of NMC kernels and shows a measurable performance increase.

2.4 High-Bandwidth Memory

Das et al. [8] propose an algorithmic foundation for automated management of memory systems that use HBM. It presents an $O(1)$ algorithm for replacing pages in HBM to use as an additional cache layer next to DRAM. Laghari et al. [16] propose an object placement algorithm to move data objects into either DRAM or HBM to maximize performance when the capacity of fast memory is insufficient. The algorithm is, however, only able to evaluate a single application at a time.

3 SYSTEM-LEVEL MODELS IN RESEARCH

For almost a decade, models that enable in-depth system-level optimization have been explored for representing NUMA systems with multicore processors. The following section discusses the most important of these models and their suitability for modeling disruptive memory technologies by evaluating their compliance with the requirements for a whole system model from Section 1.

3.1 System Knowledge Base

An example of such a model and how it can be used in operating systems to optimize management policies regarding complex memory hierarchies is the work of Schüpbach [24] for the *Barrelfish* operating system [1]. The *system knowledge base* (SKB) of Barrelfish is an OS service that stores all information about the hardware topology as well as the current utilization of each hardware resource by applications. For this, the SKB is tightly integrated with the OS, providing information about the hardware and current resource allocations.

Furthermore, the SKB provides an interface to applications to query system information and tweak system management policies. This enables to compute resource mappings that e.g. minimize communication latencies between threads, minimize interference between applications or maximize memory throughput.

However, as disruptive memory technologies, such as near-memory computing and persistent memory, were not available when the SKB was designed, it is still an open question how the SKB could represent those technologies in Barrelfish. Thus, although the SKB meets our requirements 2 to 5, it does not meet requirement 1 fully as it lacks support for DMTs.

3.2 MCTOP

Chatzopoulos et al. take another approach to a system-level model with MCTOP [5]. MCTOP is a user-space library providing detailed information on the NUMA topology of a system, such as interconnect bandwidth and latencies as well as cache and memory sizes. Here, two graphs represent the hardware topology: the first graph abstracting the intra-socket connections between cores of the same CPU, the second storing information about each socket and their interconnections together with their bandwidth and latencies.

Furthermore, MCTOP provides additional libraries that allow applications to allocate system resources corresponding to a given policy, determine the cheapest communication paths between sockets or calculate the expected latency and achievable throughput for a given memory layout and thread mapping.

The strength of MCTOP compared to traditional approaches, such as *libnuma*, is that the developer only needs to state her desired placement policy and does not need to specify precise core mappings, which are too often not portable.

However, since MCTOP runs entirely in user-space, it can only apply optimizations for a single application. This lack of a system-wide view can easily lead to interference and bad performance when two applications use MCTOP with the same resource allocations. Thus, it does not comply with requirement 4. Furthermore, MCTOP focused on providing information and optimized mapping strategies for NUMA-only. So far, MCTOP does not model disruptive memory technologies, nor does it provide optimizations for them. Hence, it does not meet the requirements 1 and 5, even though it can optimize thread mappings and memory allocations for a single application on ccNUMA systems without DMT.

3.3 hwloc

Less sophisticated but widely used for static optimization for a given machine is *hwloc* [4] by Broquedis et al., which is a platform-independent user-space library running on a variety of operating systems. The information that *hwloc* includes the whole memory hierarchy and each core of the system down to single hardware contexts, as well as information about PCI devices and which NUMA region they are associated with. Hence, *hwloc* provides the developer with detailed information about her computing system for manual optimization.

But unfortunately *hwloc* does not provide latency costs and bandwidth for intra- and inter-socket connections. Thus, it cannot be used to perform performance predictions and optimization for resource mappings of applications. Therefore while meeting the requirement 1, it does not meet the remaining requirements.

As one can see from our survey, none of the discussed models for whole system optimization meet all system software requirements to manage DMT resources efficiently. Nevertheless, the SKB, with its integration in Barrelfish, gets close to meeting all requirements. However, how Schüpbach [24] pointed out in his dissertation, the SKB as it is now, is only single-threaded and because it stores all system information exclusively, it will easily become a bottleneck if highly contended. Furthermore, the usage of a constraint-logic programs to query system information from the SKB is problematic to performance and ease of use. Although using constraint-logic programming makes the SKB very sophisticated and powerful it can reduce performance drastically and even put the whole system on halt when other applications or OS services are delayed by long-taking CPL queries.

Hence, although the model behind the SKB and its integration within the OS is promising, there are still open questions in how a system model can be integrated into an OS, such that it can be queried quickly as well as can be used by application (developers) easily. A representation like MCTOP would likely improve the usability and performance of querying such a model.

Thus, the open questions, we see, are how performance models for DMTs can be designed accurately and integrated into a whole-system model on one hand, and how to design the system- and application-interface for such a model on the other hand. Additionally, it is still to debate how a system model shall be integrated into the overall system. Shall it be a part of the OS as a system service like the SKB, or a user-space library like MCTOP? Or even something else?

4 CONCLUSION

This paper discussed the need for sophisticated system models for disruptive memory technologies to enable system software to manage these efficiently. Furthermore, we presented the current state of research regarding the modeling of such technologies. We pointed out that although there are already models for specific technologies, such as NMC and PMEM, these models still lack the generality and holistic view needed by system software and application developers to build efficient and portable software systems.

Moreover, this paper described existing system models and how these meet the requirements from Section 1. Almost none of the discussed models, except for *hwloc*, include DMTs which itself is unsuitable for performance predictions and whole-system optimization, as it lacks critical information about communication latencies and bandwidth as well as utilization by applications.

Hence, we conclude our paper with the finding that there is a severe lack of whole system models which include DMTs in performance prediction and optimization, leaving system software alone without means to optimize the usage of those emerging technologies.

We are confident that investigating whole system models for disruptive memory technologies open up exciting research opportunities for years to come. These will foster research on new ways to leverage the potential that the emerging disruptive memory technologies provide.

ACKNOWLEDGMENTS

The work on this paper has been supported by Deutsche Forschugsgemeinschaft (DFG) under grant no. SP 986/9-1.

REFERENCES

[1] Andrew Baumann, Paul Barham, Pierre-Evariste Dagand, Tim Harris, Rebecca Isaacs, Simon Peter, Timothy Roscoe, Adrian Schüpbach, and Akhilesh Singhania. 2009. The multikernel: a new OS architecture for scalable multicore systems. In *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*. ACM, 29–44.

[2] Silas Boyd-Wickizer, Haibo Chen, Rong Chen, Yandong Mao, M Frans Kaashoek, Robert Morris, Aleksey Pesterev, Lex Stein, Ming Wu, Yuehua Dai, and others. 2008. Corey: An Operating System for Many Cores.. In *OSDI*, Vol. 8. 43–57.

[3] Silas Boyd-Wickizer, Austin T. Clements, Yandong Mao, Aleksey Pesterev, M. Frans Kaashoek, Robert Morris, and Nickolai Zeldovich. 2010. An Analysis of Linux Scalability to Many Cores. In *9th USENIX Symposium on Operating Systems Design and Implementation (OSDI 10)*. USENIX Association, Vancouver, BC. https://www.usenix.org/conference/osdi10/analysis-linux-scalability-many-cores

[4] Francois Broquedis, Jérôme Clet-Ortega, Stéphanie Moreaud, Nathalie Furmento, Brice Goglin, Guillaume Mercier, Samuel Thibault, and Raymond Namyst. 2010. hwloc: A Generic Framework for Managing Hardware Affinities in HPC Applications. In *18th Euromicro Conference on Parallel, Distributed and Network-based Processing*. 180–186. https://doi.org/10.1109/PDP.2010.67 ISSN: 2377-5750.

[5] Georgios Chatzopoulos, Rachid Guerraoui, Tim Harris, and Vasileios Trigonakis. 2017. Abstracting Multi-Core Topologies with MCTOP. In *Proceedings of the Twelfth European Conference on Computer Systems*. ACM, Belgrade Serbia, 544–559. https://doi.org/10.1145/3064176.3064194

[6] Stefano Corda, Madhurya Kumaraswamy, Ahsan Javed Awan, Roel Jordans, Akash Kumar, and Henk Corporaal. 2021. NMPO: Near-Memory Computing Profiling and Offloading. *arXiv:2106.15284 [cs]* (June 2021). http://arxiv.org/abs/2106.15284 arXiv: 2106.15284.

[7] Stefano Corda, Gagandeep Singh, Ahsan Jawed Awan, Roel Jordans, and Henk Corporaal. 2019. Platform Independent Software Analysis for Near Memory Computing. In *2019 22nd Euromicro Conference on Digital System Design (DSD)*. 606–609. https://doi.org/10.1109/DSD.2019.00093

[8] Rathish Das, Kunal Agrawal, Michael A. Bender, Jonathan Berry, Benjamin Moseley, and Cynthia A. Phillips. 2020. How to Manage High-Bandwidth Memory Automatically. In *Proceedings of the 32nd ACM Symposium on Parallelism in Algorithms and Architectures (SPAA '20)*. Association for Computing Machinery, New York, NY, USA, 187–199. https://doi.org/10.1145/3350755.3400233

[9] Tudor David, Rachid Guerraoui, and Vasileios Trigonakis. 2013. Everything You Always Wanted to Know About Synchronization but Were Afraid to Ask. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles (SOSP '13)*. ACM, New York, NY, USA, 33–48. https://doi.org/10.1145/2517349.2522714 event-place: Farminton, Pennsylvania.

[10] Jerrin Shaji George, Mohit Verma, Rajesh Venkatasubramanian, and Pratap Subrahmanyam. 2020. go-pmem: Native Support for Programming Persistent Memory in Go. 859–872. https://www.usenix.org/conference/atc20/presentation/george

[11] Kevin Hsieh, Eiman Ebrahim, Gwangsun Kim, Niladrish Chatterjee, Mike O'Connor, Nandita Vijaykumar, Onur Mutlu, and Stephen W. Keckler. 2016. Transparent Offloading and Mapping (TOM): Enabling Programmer-Transparent Near-Data Processing in GPU Systems. In *ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*. 204–216. https://doi.org/10.1109/ISCA.2016.27 ISSN: 1063-6897.

[12] Joseph Izraelevitz, Jian Yang, Lu Zhang, Juno Kim, Xiao Liu, Amirsaman Memaripour, Yun Joon Soh, Zixuan Wang, Yi Xu, Subramanya R. Dulloor, Jishen Zhao, and Steven Swanson. 2019. Basic Performance Measurements of the Intel Optane DC Persistent Memory Module. *arXiv:1903.05714 [cs]* (Aug. 2019). http://arxiv.org/abs/1903.05714 arXiv: 1903.05714.

[13] Anuj Kalia, Michael Kaminsky, and David G. Andersen. 2016. Design Guidelines for High Performance {RDMA} Systems. In *Proceedings of the 2016 {USENIX} Annual Technical Conference ({USENIX} {ATC} 16)*. 437–450. https://www.usenix.org/conference/atc16/technical-sessions/presentation/kalia

[14] Kamil Khan, Sudeep Pasricha, and Ryan Gary Kim. 2020. A Survey of Resource Management for Processing-In-Memory and Near-Memory Processing Architectures. *Journal of Low Power Electronics and Applications* 10, 4 (Dec. 2020), 30. https://doi.org/10.3390/jlpea10040030 Number: 4 Publisher: Multidisciplinary Digital Publishing Institute.

[15] Marcel Köppen, Jana Traue, Christoph Borchert, Jörg Nolte, and Olaf Spinczyk. 2019. Cache-Line Transactions: Building Blocks for Persistent Kernel Data Structures Enabled by AspectC++. In *Proceedings of the 10th Workshop on Programming Languages and Operating Systems*

(PLOS'19). Association for Computing Machinery, New York, NY, USA, 38–44. https://doi.org/10.1145/3365137.3365396

[16] Mohammad Laghari and Didem Unat. 2017. Object Placement for High Bandwidth Memory Augmented with High Capacity Memory. In *Proceedings of the 29th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*. 129–136. https://doi.org/10.1109/SBAC-PAD.2017.24

[17] Donghun Lee, Minseon Ahn, Jungmin Kim, Kangwoo Cho, Oliver Rebholz, Andrew Chang, Jongmin Gim, Jaemin Jung, Vincent Pham, Krishna Malladi, and Yang Seok. 2020. Optimizing Data Movement with Near-Memory Acceleration of In-memory DBMS. *EDBT* (April 2020), 371–374. https://doi.org/10.5441/002/edbt.2020.35

[18] Patrick MacArthur and Robert D. Russell. 2012. A Performance Study to Guide RDMA Programming Decisions. In *Proceedings of the 14th International Conference on High Performance Computing and Communication 2012 IEEE 9th International Conference on Embedded Software and Systems*. 778–785. https://doi.org/10.1109/HPCC.2012.110

[19] Onur Mutlu, Saugata Ghose, Juan Gómez-Luna, and Rachata Ausavarungnirun. 2020. A Modern Primer on Processing in Memory. *CoRR* abs/2012.03112 (2020). https://arxiv.org/abs/2012.03112

[20] Jacob Nelson and Roberto Palmieri. 2019. Understanding RDMA Behavior in NUMA Systems. In *Proceedings of the IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*. 273–274. https://doi.org/10.1109/CGO.2019.8661190

[21] Constantin Pohl and Kai-Uwe Sattler. A Cost Model for Data Stream Processing on Modern Hardware. (????), 8.

[22] Haonan Qiu, Xiaoliang Wang, Tianchen Jin, Zhuzhong Qian, Baoliu Ye, Bin Tang, Wenzhong Li, and Sanglu Lu. 2018. Toward Effective and Fair RDMA Resource Sharing. In *Proceedings of the 2nd Asia-Pacific Workshop on Networking (APNet '18)*. Association for Computing Machinery, New York, NY, USA, 8–14. https://doi.org/10.1145/3232565.3232636

[23] Steve Scargall. 2020. Introducing the Persistent Memory Development Kit. In *Programming Persistent Memory: A Comprehensive Guide for Developers*, Steve Scargall (Ed.). Apress, Berkeley, CA, 63–72. https://doi.org/10.1007/978-1-4842-4932-1_5

[24] Adrian L. Schüpbach. 2012. *Tackling OS Complexity with Declarative Techniques*. PhD Thesis. ETH Zurich.

[25] Dian Shen, Junzhou Luo, Fang Dong, Xiaolin Guo, Kai Wang, and John C. S. Lui. 2020. Distributed and Optimal RDMA Resource Scheduling in Shared Data Center Networks. In *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*. 606–615. https://doi.org/10.1109/INFOCOM41043.2020.9155533 ISSN: 2641-9874.

[26] Gagandeep Singh, Lorenzo Chelini, Stefano Corda, Ahsan Javed Awan, Sander Stuijk, Roel Jordans, Henk Corporaal, and Albert-Jan Boonstra. 2019. Near-memory computing: Past, present, and future. *Microprocessors and Microsystems* 71 (Nov. 2019), 102868. https://doi.org/10.1016/j.micpro.2019.102868

[27] Gagandeep Singh, Juan Gómez-Luna, Giovanni Mariani, Geraldo F. Oliveira, Stefano Corda, Sander Stuijk, Onur Mutlu, and Henk Corporaal. 2019. NAPEL: Near-Memory Computing Application Performance Prediction via Ensemble Learning. In *Proceedings of the 56th ACM/IEEE Design Automation Conference (DAC)*. 1–6. ISSN: 0738-100X.

[28] Zixuan Wang, Xiao Liu, Jian Yang, Theodore Michailidis, Steven Swanson, and Jishen Zhao. 2020. Characterizing and Modeling Non-Volatile Memory Systems. In *Proceedings of the 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 496–508. https://doi.org/10.1109/MICRO50266.2020.00049

[29] Xingda Wei, Xiating Xie, Rong Chen, Haibo Chen, and Binyu Zang. 2021. Characterizing and Optimizing Remote Persistent Memory with {RDMA} and {NVM}. In *Proceedings of the 2021 {USENIX} Annual Technical Conference ({USENIX} {ATC} 21)*. 523–536. https://www.usenix.org/conference/atc21/presentation/wei