Virtuoso: An Open-Source, Comprehensive and Modular Simulation Framework for Virtual Memory Research

Konstantinos Kanellopoulos[1] Konstantinos Sgouras[2] Onur Mutlu[1]

ETH Zürich[1] University of Athens[2]

1 INTRODUCTION & BACKGROUND

Virtual memory is a cornerstone of modern computing systems. Introduced as one of the earliest instances of hardware-software co-design, VM facilitates programmer-transparent memory management, data sharing, process isolation and memory protection. In light of current application trends with large data sets and irregular memory access patterns [15, 17, 20, 33, 49] and as we transition to much larger address spaces [22] (e.g., hybrid memory systems with both volatile and non-volatile memories [2, 32]) and heterogeneous memory systems [31], the overheads of virtual memory are likely to increase [57]. Research span across various solutions to reduce the overheads of virtual memory in the context of (i) improving the efficiency of the TLB subsystem [4, 34, 40, 43, 53] (e.g., software-based TLB [43] etc.), (ii) the efficiency of page table structures [13, 39, 46, 47, 58] (e.g., hash-based page tables [46, 47]), (iii) accelerating address translation in virtualized environments [7, 12, 13, 35, 48], (iv) leveraging contiguity between virtual/physical addresses to enable offset-based address translation [3, 27, 57, 60] (e.g., range-based translation [25]), (v) enabling hash-based virtual-to-physical address mapping [14, 23, 42], (vi) enabling intermediate address spaces to perform address translation only upon LLC misses [6, 16, 18], (vii) designing efficient large page allocation techniques [10, 11, 29, 37, 60] (e.g., Transparent Huge Pages [10]) and (viii) reducing the overheads of minor page faults [30, 51].

2 LACK OF COMPREHENSIVE VIRTUAL MEMORY SIMULATION FRAMEWORK

Evaluating the efficiency of various virtual memory (VM) designs is crucial (i) given their significant impact on the system, including the CPU caches, the main memory, and the storage device and (ii) given that different system architectures might benefit from various VM techniques. Such an evaluation is not straightforward, as it heavily hinges on modeling the interplay between different VM techniques and the interactions of VM with the system architecture. To better illustrate the interplay between VM techniques, for example, the memory management policies can directly affect TLB performance [38], contiguity-aware VM solutions significantly influence memory fragmentation [3], affecting disk accesses, while the contiguity between virtual and physical addresses is pivotal for the efficiency of prefetching [54]. As also highlighted in prior works [39, 58], the design of the page table can negatively impact main memory efficiency. With respect to system architecture, mobile devices, with their resource constraints, might benefit from

Virtual memory components supported by existing simulators and Virtuoso

Simulator/ Component	TLB Subsystem	Page Table Design	Contiguity Schemes	Intermediate Address Space	Hash-based Translation	Metadata Management	Memory Management	Page Fault Modelling
SimpleScalar [50]	L1 D-TLB & L1 D-ITLB	✗	✗	✗	✗	✗	✗	✗
PTLSim [59]	L1 D-TLB & L1 D-ITLB	Simplistic Walker	✗	✗	✗	✗	✗	✓
Scarab [19]	✗	✗	✗	✗	✗	✗	✗	✗
Ramulator [28]	✗	✗	✗	✗	✗	✗	✗	✗
ZSim [45]	✗	✗	✗	✗	✗	✗	✗	✗
Multi2Sim [52]	L1 D-TLB & L1 D-ITLB	✗	✗	✗	✗	✗	✗	✗
Gem5 Syscall-Emu [8]	2-Level TLB Hierarchy	4-level Radix-Tree Walker	✗	✗	✗	✗	✗	✗
Gem5 Full system [8]	2-Level TLB Hierarchy	4-level Radix-Tree Walker	✗	✗	✗	✗	Transparent Huge Pages [10]	✓
Champsim [1]	2-Level TLB Hierarchy & TLB Prefetching	4-level Radix-Tree Walker	✗	✗	✗	✗	✗	✗
Sniper [9]	2-Level TLB Hierarchy	Static PTW latency	✗	✗	✗	✗	✗	✗
Virtuoso	Configurable TLB hierarchy	Memory-Efficient Hash Table [47]	Offset-based translation with Redundant Memory Mappings [25]	Virtual Block Interface with configurable past-LLC translation [44]	Hybrid address mapping with Utopia [23]	Memory tagging with XMem [55]	Memory Management Emulation	Bi-directional inter-process communication between Virtuoso and custom OS Software to emulate the functionality of PF and simulate the architectural events of PF
	Multi-page Size TLBs (Serial probing)	Hash Table with Open-Addressing and PTE Clustering [58]					Buddy Allocator with pre-created memory allocation snapshots	
	Page-size Predictors	Elastic Cuckoo Hash Table [46]					Eager Paging to allocate large contiguous blocks [25]	
	TLB prefetching techniques	Configurable Radix Table with PWCs	Direct Segments [5]	Midgard-based translation with trace-based VMA tracking [16]	Hash-based translation with NMT [41]	Mondriaan Data Protection [56]	Reservation-based THP [36]	
	Software-based TLBs [43]	Support for Nested MMUs (MMU with Nested TLB, Nested Walk etc.)					Artificial Fragmentation Generator	
	Storing TLB entries in data caches [24]							

lightweight VM designs that minimize energy overheads. In contrast, cloud architectures, with abundant resources, might favour VM designs that maximize throughput and scalability [21]. In contrast, commercial multi-core architectures, with multiple threads running concurrently, might favour VM designs that reduce contention and ensure efficient memory bandwidth distribution among cores.

In the face of these intricacies, assessing the merits and drawbacks of old, modern and futuristic VM proposals becomes a highly-challenging task without a comprehensive simulation tool. Modern simulators, however, struggle to keep up with the rapid VM research developments, lacking the capability to model a wide range of contemporary VM techniques and their interactions. Table 1 summarizes the VM techniques that are supported by existing simulators. For example, Sniper [9], does not include a realistic model for the page table walk latency, neither emulates nor simulates the page fault handler and does not emulate memory management. The system call emulation mode of gem5 [8] does not include detailed TLB models (e.g., all TLBs are considered fully-associative), and neither emulates nor simulates page fault handling and large page allocation. At the same time, the full system simulation of gem5 emulates and simulates the actual OS including realistic memory management, large page allocation but at the cost of (i) high programmability effort, e.g., enabling contiguity-based translation requires modifying the kernel code in a functional manner and co-design it with the simulated hardware and (ii) high simulation time (e.g., simulating multi-programmed workloads in parallel is not supported). Given the wide range of research directions and the synergies between different VM components, our goal in this work is to provide researchers with an open-source, comprehensive and modular tool to study and evaluate various VM techniques and memory management schemes.

3 OUR APPROACH: VIRTUOSO

To this end, we present Virtuoso, an open-source, comprehensive and modular simulation framework that models various VM designs to establish a common ground for virtual memory research. Virtuoso is built on top of sniper [9] and, as show in Table 1, extends it with (i) state-of-the-art TLB techniques and organizations, (ii) four different page table designs from cutting-edge academic proposals, (iii) support for easily configurable Nested MMUs, required in virtualized environments(e.g., MMU with Nested TLB, Nested Walk etc.), (iv) translation techniques that exploit virtual and physical memory contiguity [5, 26] (v) support for address translation schemes that rely on intermediate address spaces [16, 18], (vi), metadata management schemes that enhance security and performance [55, 56], (vii) a complete memory management emulator that includes a reservation-based programmer-transparent large page allocator and (viii) a new simulation methodology that involves bi-directional inter-process communication between Virtuoso and software to estimate the performance of OS kernel code with a focus on the minor page fault handler. Even though Virtuoso is built on top of Sniper, it is highly-modular, avoiding the use of simulator-specific semantics. The only requirement to integrating Virtuoso in any architectural simulator, is attaching it to the simulator's memory subsystem model (e.g. interface to access the L1 data cache).

To demonstrate the versatility and utility of Virtuoso we examine four new example case studies.

- We analyze the performance, memory and cache footprint of different page table designs under different system workloads (e.g, memory fragmentation and memory bandwidth) and different execution environments (native vs virtualized execution).
- We perform a head-to-head comparison between large-page, intermediate address space and contiguity based schemes taking into consideration (i) address translation latency and (ii) impact on memory fragmentation.
- We examine the performance and microarchitectural impact of Virtuoso's reservation-based large page allocator across different fragmentation levels and compare it against Linux THP.
- We evaluate the microarchitectural impact caused by minor page faults in the presence of different page allocation policies and contiguity-aware translation approaches.

4 OUR CONTRIBUTIONS

In this work, we make the following contributions:

- We present Virtuoso, an open-source, comprehensive and modular simulator that models various virtual memory components and schemes to establish a common ground for virtual memory research.
- We demonstrate the versatility and the potential of Virtuoso with four new case studies. A minimal, alpha version of the simulator is used in two recent research works [23, 24], is currently under review for artifact evaluation and can be found under the following link: https://github.com/CMU-SAFARI/Victima. We will soon release a new version of Virtuoso.

REFERENCES

[1] ChampSim. https://github.com/ChampSim/ChampSim.

[2] Chloe Alverti, Vasileios Karakostas, Nikhita Kunati, Georgios Goumas, and Michael Swift. DaxVM: Stressing the Limits of Memory as a File Interface. In *MICRO 2022*.

[3] Chloe Alverti, Stratos Psomadakis, Vasileios Karakostas, Jayneel Gandhi, Konstantinos Nikas, Georgios Goumas, et al. Enhancing and Exploiting Contiguity for Fast Memory Virtualization. In *ISCA*. 2020.

[4] Thomas W. Barr, Alan L. Cox, and Scott Rixner. SpecTLB: A Mechanism for Speculative Address Translation. In *ISCA*. 2011.

[5] Arkaprava Basu, Jayneel Gandhi, Jichuan Chang, Mark D. Hill, and Michael M. Swift. Efficient Virtual Memory for Big Memory Servers. In *ISCA 2013*.

[6] Arkaprava Basu, Mark D. Hill, and Michael M. Swift. Reducing Memory Reference Energy with Opportunistic Virtual Caching. In *ISCA*. 2012.

[7] Ravi Bhargava, Benjamin Serebrin, Francesco Spadini, and Srilatha Manne. Accelerating Two-Dimensional Page Walks for Virtualized Systems. In *ASPLOS*. 2008.

[8] Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K. Reinhardt, Ali Saidi, Arkaprava Basu, et al. The gem5 Simulator. *Comput. Archit. News*. 2011.

[9] Trevor E. Carlson, Wim Heirman, and Lieven Eeckhout. Sniper: Exploring the Level of Abstraction for Scalable and Accurate Parallel Multi-Core Simulations. In *SC 2011*.

[10] Jonathan Corbet. Transparent Huge Pages in 2.6.38. https://lwn.net/Articles/423584/. 2011.

[11] Yu Du, Miao Zhou, Bruce R Childers, Daniel Mossé, and Rami Melhem. Supporting superpages in non-contiguous physical memory. In *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 223–234. 2015.

[12] Jayneel Gandhi, Arkaprava Basu, Mark D. Hill, and Michael M. Swift. Efficient Memory Virtualization: Reducing Dimensionality of Nested Page Walks. In *MICRO*. 2014.

[13] Jayneel Gandhi, Mark D. Hill, and Michael M. Swift. Agile Paging: Exceeding the Best of Nested and Shadow Paging. In *ISCA '16*.

[14] Krishnan Gosakan, Jaehyun Han, William (Massachusetts Inst. of Technology) Kuszmaul, Ibrahim Nael Mubarek, Nirjhar Mukherjee, Guido Tagliavini, et al. Mosaic Pages: Big TLB Reach with Small Pages. In *ASPLOS 2023*.

[15] Graph 500. Large-Scale Benchmarks. http://www.graph500.org/.

[16] Siddharth Gupta, Atri Bhattacharyya, Yunho Oh, Abhishek Bhattacharjee, Babak Falsafi, and Mathias Payer. Rebooting Virtual Memory with Midgard. In *ISCA*. 2021.

[17] Udit Gupta, Xiaodong Wang, Maxim Naumov, Carole-Jean Wu, Brandon Reagen, David Brooks, et al. The Architectural Implications of Facebook's DNN-based Personalized Recommendation. In *arXiv 2019*.

[18] Nastaran Hajinazar, Pratyush Patel, Minesh Patel, Konstantinos Kanellopoulos, Saugata Ghose, Rachata Ausavarungnirun, et al. The Virtual Block Interface: A Flexible Alternative to the Conventional Virtual Memory Framework. In *ISCA*. 2020.

[19] Hpsresearchgroup. "hpsresearchgroup/scarab: Joint hps and eth repository to work towards open sourcing scarab and ramulator.". https://github.com/hpsresearchgroup/scarab.

[20] R. Hwang, T. Kim, Y. Kwon, and M. Rhu. Centaur: A Chiplet-based, Hybrid Sparse-Dense Accelerator for Personalized Recommendations. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. 2020.

[21] Bongjoon Hyun, Youngeun Kwon, Yujeong Choi, John Kim, and Minsoo Rhu. NeuMMU: Architectural Support for Efficient Address Translations in Neural Processing Units. In *ASPLOS '20*.

[22] Intel Corp. 3rd Generation Intel® Xeon® Scalable processore. https://www.intel.com/content/www/us/en/products/docs/processors/embedded/3rd-gen-xeon-scalable-iot-product-brief.html.

[23] Konstantinos Kanellopoulos, Rahul Bera, Kosta Stojiljkovic, Nisa Bostanci, Can Firtina, Rachata Ausavarungnirun, et al. Utopia: Efficient Address Translation using Hybrid Virtual-to-Physical Address Mapping.

[24] Konstantinos Kanellopoulos, Hong Chul Nam, Nisa Bostanci, Rahul Bera, Mohammad Sadrosadati, Rakesh Kumar, et al. Victima: Drastically Increasing Address Translation Reach by Leveraging Underutilized Cache Resources. In *MICRO 2023*.

[25] Vasileios Karakostas, Jayneel Gandhi, Furkan Ayar, Adrián Cristal, Mark D. Hill, Kathryn S. McKinley, et al. Redundant Memory Mappings for Fast Access to Large Memories. In *ISCA*. 2015.

[26] V. Karakostas, J. Gandhi, F. Ayar, A. Cristal, M. D. Hill, K. S. McKinley, et al. Redundant Memory Mappings for Fast Access to Large Memories. In *ISCA*. 2015.

[27] Vasileios Karakostas, Jayneel Gandhi, Adrián Cristal, Mark D. Hill, Kathryn S. McKinley, Mario Nemirovsky, et al. Energy-Efficient Address Translation. In *HPCA*. 2016.

[28] Yoongu Kim, Weikun Yang, and Onur Mutlu. Ramulator: A Fast and Extensible DRAM Simulator. *CAL*. 2015.

[29] Youngjin Kwon, Hangchen Yu, Simon Peter, Christopher J. Rossbach, and Emmett Witchel. Coordinated and Efficient Huge Page Management with Ingens. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*.

[30] Gyusun Lee, Wenjing Jin, Wonsuk Song, Jeonghun Gong, Jonghyun Bae, Tae Jun Ham, et al. A Case for Hardware-Based Demand Paging. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. 1103–1116. 2020. https://doi.org/10.1109/ISCA45697.2020.00093

[31] Yang Li, Saugata Ghose, Jongmoo Choi, Jin Sun, Hui Wang, and Onur Mutlu. Utility-Based Hybrid Memory Management. In *CLUSTER*. 2017.

[32] Sihang Liu, Korakit Seemakhupt, Gennady Pekhimenko, Aasheesh Kolli, and Samira Khan. Janus: Optimizing Memory and Storage Support for Non-Volatile Memory Systems. In *ISCA*. 2019.

[33] Piotr R Luszczek, David H Bailey, Jack J Dongarra, Jeremy Kepner, Robert F Lucas, Rolf Rabenseifner, et al. The HPC Challenge (HPCC) Benchmark Suite. In *SC*. 2006.

[34] Yashwant Marathe, Nagendra Gulur, Jee Ho Ryoo, Shuang Song, and Lizy K. John. CSALT: Context Switch Aware Large TLB. In *MICRO*. 2017.

[35] Artemiy Margaritov, Dmitrii Ustiugov, Amna Shahab, and Boris Grot. PTEMagnet: Fine-grained physical memory reservation for faster page walks in public clouds. In *ASPLOS*. 2021.

[36] Juan Navarro, Sitaram Iyer, Peter Druschel, and Alan Cox. Practical, Transparent Operating System Support for Superpages. In *OSDI*. 2002.

[37] Ashish Panwar, Sorav Bansal, and K Gopinath. Hawkeye: Efficient fine-grained os support for huge pages. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*. 2019.

[38] Chang Hyun Park, Taekyung Heo, Jungi Jeong, and Jaehyuk Huh. Hybrid TLB Coalescing: Improving TLB Translation Coverage Under Diverse Fragmented Memory Allocations. In *ISCA*. 2017.

[39] Chang Hyun Park, Ilias Vougioukas, Andreas Sandberg, and David Black-Schaffer. Every Walk's a Hit: Making Page Walks Single-Access Cache Hits. In *ASPLOS '22*.

[40] Binh Pham, Viswanathan Vaidyanathan, Aamer Jaleel, and Abhishek Bhattacharjee. CoLT: Coalesced Large-Reach TLBs. In *MICRO*. 2012.

[41] Javier Picorel, Djordje Jevdjic, and Babak Falsafi. Near-Memory Address Translation. In *PACT*. 2017.

[42] Javier Picorel, Djordje Jevdjic, and Babak Falsafi. Near-Memory Address Translation. In *PACT*. 2017.

[43] Jee Ho Ryoo, Nagendra Gulur, Shuang Song, and Lizy K. John. Rethinking TLB designs in virtualized environments: A very large part-of-memory TLB. In *ISCA*. 469–480. 2017.

[44] SAFARI Research Group. Ramulator-VBI — GitHub Repository. https://github.com/CMU-SAFARI/Ramulator-VBI.git.

[45] Daniel Sanchez and Christos Kozyrakis. ZSim: Fast and Accurate Microarchitectural Simulation of Thousand-Core Systems. In *ISCA*. 2013.

[46] Dimitrios Skarlatos, Apostolos Kokolis, Tianyin Xu, and Josep Torrellas. Elastic Cuckoo Page Tables: Rethinking Virtual Memory Translation for Parallelism. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '20)*. Association for Computing Machinery, New York, NY, USA, 1093–1108. 2020. https://doi.org/10.1145/3373376.3378493

[47] Jovan Stojkovic, Namrata Mantri, Dimitrios Skarlatos, Tianyin Xu, and Josep Torrellas. Memory-Efficient Hashed Page Tables. In *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. 1221–1235. 2023. https://doi.org/10.1109/HPCA56546.2023.10071061

[48] Jovan Stojkovic, Dimitrios Skarlatos, Apostolos Kokolis, Tianyin Xu, and Josep Torrellas. Parallel Virtualized Memory Translation with Nested Elastic Cuckoo Page Tables. In *ASPLOS*. 84–97. 2022.

[49] Arun Subramaniyan, Yufeng Gu, Timothy Dunn, Somnath Paul, Md. Vasimuddin, Sanchit Misra, et al. GenomicsBench: A Benchmark Suite for Genomics. In *ISPASS*. 2021.

[50] D. Ernst T. Austin, E. Larson. SimpleScalar: an infrastructure for computer system modeling. In *Computer, (Volume: 35, Issue: 2, February 2002)*. IEEE, 59–67. 2002.

[51] Chandrahas Tirumalasetty, Chih Chieh Chou, Narasimha Reddy, Paul Gratz, and Ayman Abouelwafa. Reducing Minor Page Fault Overheads through Enhanced Page Walker. *ACM Trans. Archit. Code Optim.* 2022.

[52] Rafael Ubal, Byunghyun Jang, Perhaad Mistry, Dana Schaa, and David Kaeli. Multi2Sim: A Simulation Framework for CPU-GPU Computing *(PACT '12)*.

[53] Georgios Vavouliotis, Lluc Alvarez, Vasileios Karakostas, Konstantinos Nikas, Nectarios Koziris, Daniel A. Jiménez, et al. Exploiting Page Table Locality for Agile TLB Prefetching. In *ISCA*. 85–98. 2021. https://doi.org/10.1109/ISCA52012.2021.00016

[54] Georgios Vavouliotis, Gino Chacon, Lluc Alvarez, Paul V. Gratz, Daniel A. Jiménez, and Marc Casas. Page Size Aware Cache Prefetching. In *MICRO 2022*.

[55] Nandita Vijaykumar, Abhilasha Jain, Diptesh Majumdar, Kevin Hsieh, Gennady Pekhimenko, Eiman Ebrahimi, et al. A Case for Richer Cross-Layer Abstractions: Bridging the Semantic Gap with Expressive Memory. In *ISCA*. 2018.

[56] Emmett Witchel, Josh Cates, and Krste Asanović. Mondrian Memory Protection. In *ASPLOS*. 2002.

[57] Zi Yan, Daniel Lustig, David Nellans, and Abhishek Bhattacharjee. Translation Ranger: Operating System Support for Contiguity-Aware TLBs. In *ISCA*. 2019.

[58] Idan Yaniv and Dan Tsafrir. Hash, Don't Cache (the Page Table). In *SIGMETRICS*. 337–350. 2016.

[59] M.T. Yourst. PTLsim: A Cycle Accurate Full System x86-64 Microarchitectural Simulator *(IEEE International Symposium on Performance Analysis of Systems Software)*. 2007.

[60] Kaiyang Zhao, Kaiwen Xue, Ziqi Wang, Dan Schatzberg, Leon Yang, Antonis Manousis, et al. Contiguitas: The Pursuit of Physical Memory Contiguity in Datacenters. In *ISCA '23*.