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

Konstantinos Kanellopoulos¹  Konstantinos Sgouras²  Onur Mutlu¹

ETH Zürich¹ University of Athens²

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 [5], 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

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SimpleScalar [59]</td>
<td>L1 D-TLB &amp; L1 D-ITLB</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>PTLSim [59]</td>
<td>L1 D-TLB &amp; L1 D-ITLB</td>
<td>Simplicistic Walker</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Scarab [19]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Ramulator [28]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>ZSim [46]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Multi2Sim [52]</td>
<td>L1 D-TLB &amp; L1 D-ITLB</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Gem5 Full system [8]</td>
<td>2-Level TLB Hierarchy</td>
<td>4-level Radix-Tree Walker</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Gem5 Syscall-Emu [8]</td>
<td>2-Level TLB Hierarchy</td>
<td>4-level Radix-Tree Walker</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Champsim [1]</td>
<td>2-Level TLB Hierarchy &amp; TLB Prefetching</td>
<td>4-level Radix-Tree Walker</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Sniper [9]</td>
<td>2-Level TLB Hierarchy</td>
<td>State PTW latency</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Virtual memory components supported by existing simulators and Virtuoso

- Configurable TLB hierarchy
- Multi-page Size TLBs (Serial probing)
- Page-size Predictors
- TLB Prefetching techniques
- Storing TLBs in data caches [24]
- Memory-Efficient Hash Table [47]
- Offset-based translation with Redundant Memory Mappings [25]
- Hybrid address mapping with Utopia [23]
- Hybrid Memory tagging with XMem [55]
- Memory Management Emulation
- Buddy Allocate with pre-created memory allocation snapshots
- Eager Paging to allocate large contiguous blocks [26]
- Reservation-based THP [16]
- Bi-directional inter-process communication between Virtuoso and custom OS
- Software to emulate the functionality of PF and simulate the architectural events of PF

1

Konstantinos Kanellopoulos
Konstantinos Sgouras
Onur Mutlu

1 ETH Zürich
2 University of Athens

1

Konstantinos Sgouras

1 University of Athens

1

University of Athens
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 shown 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


Piotr R Luszczek, David H Bailey, Jack J Dongarra, Jeremy Kepner, Robert F Yang Li, Saugata Ghose, Jongmoo Choi, Jin Sun, Hui Wang, and Onur Mutlu.

Konstantinos Kanellopoulos, Hong Chul Nam, Nisa Bostanci, Rahul Bera, Moham-

Hpsresearchgroup. "hpsresearchgroup/scarab: Joint hps and eth repository

Udit Gupta, Xiaodong Wang, Maxim Naumov, Carole-Jean Wu, Brandon Reagen,

Jayneel Gandhi, Arkaprava Basu, Mark D. Hill, and Michael M. Swift. Efficient

Yoongu Kim, Weikun Yang, and Onur Mutlu. Ramulator: A Fast and Extensible

Vasileios Karakostas, Jayneel Gandhi, Furkan Ayar, Adrian Cristal, Mark D. Hill,

Nastaran Hajinazar, Pratyshe Patel, Miness Patel, Konstantinos Kanellopoulos,

Samira Khan. Janus: Optimizing Memory and Storage Support for Non-Volatile


Zi Yan, Daniel Lustig, David Nellans, and Abhishek Bhattachar-


Idan Yizyp and Dan Tsafrir. Hash, Don’t Cache (the Page Table). In SIGMETRICS. 337–350. 2016.


Binh Pham, Vishwanathan Vaidyanathan, Aamer Jaleel, and Abhishek Bhattachar-

ColT: Coalesced Large-Reach TLBs. In Micro. 2012.

Javier Picorel, Djordje Jevdjc, and Babak Falsafi. Near-Memory Address Trans-


Javier Picorel, Djordje Jevdjc, and Babak Falsafi. Near-Memory Address Trans-


Daniel Sanchez and Christos Kozyrakis. ZSim: Fast and Accurate Microarchitec-


Nastaran Hajinazar, Pratyush Patel, Minesh Patel, Konstantinos Kanellopoulos,


Konstantinos Kanellopoulos, Hong Chai Nam, Nisa Bostanci, Rahul Bera, Moham-


