Towards Fast and Power-Efficient System Suspend

Yussuf Khalil*
Karlsruhe Institute of Technology

Felix Zimmer†
Karlsruhe Institute of Technology

Fabian Meyer†
Karlsruhe Institute of Technology

Frank Bellosa
Karlsruhe Institute of Technology

Abstract

Suspend is a common operating system feature that puts a system into a state with minimal power consumption while it is not being actively used with the aim of being able to resume to the previous state as quickly as possible. Two common approaches are *suspend-to-memory* (e.g., ACPI S3, S0ix) and *suspend-to-disk* (e.g., ACPI S4). The former allows for fast resume times by keeping system memory (i.e., DRAM) alive, and hence requires some energy to uphold the suspended state. The latter, in contrast, does not require any energy while suspended and therefore allows for indefinitely long sleeps in battery-powered systems, but in turn suffers from long resume times. In our work-in-progress effort, we present a novel approach to this decades-old problem that combines the best of both worlds by leveraging upcoming CXL-based persistent memory technologies (e.g., hybrid SSDs) in an OS/firmware co-design approach. Specifically, we enable cooperation between firmware and the OS to shortcut the startup process when resuming from a powered-off state and minimize copying effort from non-volatile storage by mapping memory pages via CXL. Preliminary results show a 5.8× faster wake-up from a powered-off state vs. S4.

1 Introduction

Common consumer use cases involve computers that are used only for certain periods during the day, e.g., a laptop that is only in use during work hours. Keeping the system in full operation during the remaining time would be a waste of energy, however, users expect their computer to be ready to use with only minimal delay when returning. In turn, several designs for *system suspend* have been developed in the past that provide various trade-offs regarding resume time, energy consumption, and crash resiliency.

Traditionally, ACPI S3 and S4 have been widespread standardized design concepts for system suspend [19]. S3 is a *suspend-to-memory* approach that turns the system off except for system memory. Given that all system state is still present in memory when resuming, the system is ready again very soon when desired by the user. However, keeping memory contents alive requires (in the case of typical DRAM technology) continuous energy supply, which in turn creates a risk of data loss for battery-powered systems. S4, in contrast, realizes the *suspend-to-disk* idea where all memory contents are copied to non-volatile storage before powering off the system. Although this allows resuming the system into the previous state from a powered-off state, it requires completing most of a typical cold boot process before the operating system can restore memory contents from mass storage.

In recent years, *suspend-to-idle* (also known as S0ix, *modern standby*, or *low-power idle*) has evolved as a new approach that can be considered a subtype of *suspend-to-memory* [11, 16, 19]. While ACPI S3 requires the help of system firmware, S0ix is a pure operating system-level design that expects the CPU to enter a low-power state as soon as all cores are idle and execution is fully halted. In turn, the operating system can decide to leave peripheral hardware (e.g., NICs) powered. S0ix therefore aims to be able to wake the system for notifications or incoming calls while also promising shorter resume times due to removing firmware from the equation.

Researchers have brought forward several ideas around system suspend, e.g., page priority analysis to improve S4 resume times [9, 15], or suspending to PCRAM [22]. Instant-on/off approaches that are typically found in consumer appliances such as smart TVs are related to system suspend, but also different in central aspects. Typically, they focus on resuming to a predefined system state (i.e., not an arbitrary previous one) [13] or degrade runtime performance by fully exchanging DRAM for a persistent memory technology [14].

In this work, we present a novel design that aims to provide wake-up times similar to *suspend-to-memory*, while also being as energy-efficient as *suspend-to-disk*. For this goal, we employ CXL-based hybrid SSD technology that we embed into a co-design approach that includes both system firmware as well as the operating system. We thereby significantly reduce the amount of work that needs to be done when resuming from a powered-off state in a manner that is agnostic of specific firmware or OS implementations.

2 Background

CXL is a recent CPU-to-device interconnect standard that offers, among other new features, byte-granular access to device-attached memory [7]. Despite being a data center-focused standard, several stakeholders have voiced an interest for bringing CXL into the consumer space [4, 6, 8]. Based upon CXL, various vendors have announced *hybrid SSD* products, e.g., Samsung *CMM-H* [17] and Wolley *NVMe-over-CXL* [18]. Unlike traditional SSDs, which solely expose a block-granular and asynchronous access interface via NVMe, hybrid SSDs additionally offer a byte-granular synchronous interface via the *CXL.mem* protocol. Various studies have explored the design space and possible use cases for hybrid SSDs [3, 5, 12, 21].

*PhD student, † Undergraduate student

3 Design

Our approach encompasses several novel design aspects that form a suspend mode that aims to bring together three, previously contradicting, design goals: fast resume times, zero power consumption in suspended state, and near-zero runtime performance impact. We will describe our system suspend design by iterating over the high-level steps performed during suspend and resume. Generally, we assume the presence of a hybrid SSD as described in the previous section.

First, the suspend process in our design starts off in a similar fashion to an S3 suspend. The operating system freezes all processes and stores important peripheral hardware state into system memory. However, unlike during S3, we then migrate all user space memory pages to the hybrid SSD. *Migrating* in this context encompasses not only the process of copying page contents, but also incurs modifying the page tables to make all pages point directly to the hybrid SSD. After completing OS-level suspend preparations, control is handed over to the firmware (i.e., on x86, a *System Management Interrupt* (SMI) is triggered to enter *System Management Mode* (SMM) [10]). Firmware then copies the OS kernel's memory to the hybrid SSD. Doing this in firmware ensures that the kernel memory is precisely in the same state it would be in during an S3 suspend process at the moment of giving up control. Now that all memory contents are preserved on non-volatile storage, firmware powers off the system.

On resume (i.e., when powering on the system), the specific preparations done while entering suspend allow taking several shortcuts to bring the system into a ready state. Notably, within firmware, our design can take an S3-like initialization path where peripheral hardware initialization is skipped and only CPU silicon initialization needs to be performed. To this end, firmware establishes a minimal CXL link and then copies back kernel memory from the hybrid SSD device into system memory. At this point, the system is already in the same state it was in before suspending from an OS perspective. An S4 resume would, in contrast, require to fully initialize peripheral hardware as well as the respective EFI drivers before control can be handed over to a bootloader and then, the operating system. Our design then jumps back to the operating system's S3 resume vector and the kernel can perform its typical resume duties. After that, the system is directly usable from an end user's point of view as user space processes can operate immediately as all their virtual memory pages point to the hybrid SSD. If we were to use NVMe instead of CXL, a substantial amount of page faults along with asynchronous page copies would be incurred, which would arguably slow down the system to the point of being unusable from an end user's perspective. Operating processes from the hybrid SSD, however, has a performance impact compared to operation in main memory. We therefore propose migrating pages back to system memory in an asynchronous fashion in a background kernel worker.

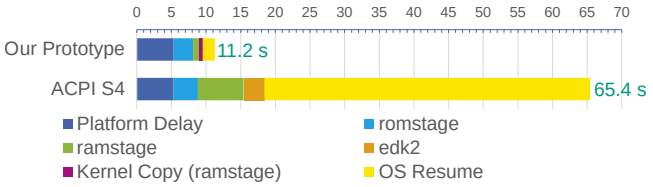

Figure 1. Resume time from power button press to Linux HDMI video output (seconds).

This migration must be performed carefully to ensure that no user-perceivable system lag is created by competing for bandwidth with actively running processes, while also minimizing the time taken until all memory is restored.

We add a 1 MiB shared memory buffer for implementation-agnostic communication between the operating system and firmware to enable their cooperation. Based on this close cooperation scheme, we expect our design to allow an interactive system to resume from non-volatile storage to a usable state with only minimally larger delay compared to an S3 implementation.

4 Evaluation

We implement a prototype of our design in Linux 6.9 and coreboot 24.05 [1] on a mainboard equipped with a CXL-capable Intel Xeon Silver 4410Y and 16 GB DDR5 memory. As hybrid SSDs are not yet commercially available, we resort to an FPGA-based CXL memory expander for our prototype.

Figure 1 shows preliminary results. We evaluate the time it takes the system to resume from the moment the power button is pressed until there is video output via HDMI from a GUI environment running on Linux, i.e., the user-perceived resume delay. Our prototype is able to resume the system within 11.2 s, whereas an S4 resume takes 65.4 s. In the plot, *platform delay* denotes the time between power-on and first x86 code execution within firmware, *romstage* and *ramstage* form the time spent in coreboot, and *edk2* [2] is the EFI DXE [20] firmware component. As we are currently working on firmware-level support for S3 and suspend-to-idle on this system, a direct comparison is unfortunately not yet possible at the time of writing. However, we expect a S3 resume to be merely 4.2 % faster than our prototype as our only additional overhead is the kernel copy process (472 ms, 1.5 GiB).

We plan to evaluate energy efficiency, runtime performance and interactivity impact, as well as potential data center use cases in the future.

5 Conclusion

We have presented the first design approach for system suspend that achieves the combination of zero power consumption while suspended, short wake-up times, and nearly unaffected runtime performance. An early prototype implementation of our design resumes 5.8× faster than S4.

References

[1] 2024. *coreboot: Fast, secure and flexible Open Source firmware.* https://www.coreboot.org/

[2] 2024. *What is TianoCore?* https://www.tianocore.org/

[3] Ahmed Abulila, Vikram Sharma Mailthody, Zaid Qureshi, Jian Huang, Nam Sung Kim, Jinjun Xiong, and Wen-mei Hwu. 2019. FlatFlash: Exploiting the Byte-Accessibility of SSDs within a Unified Memory-Storage Hierarchy. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems* (Providence, RI, USA) *(ASPLOS '19)*. Association for Computing Machinery, New York, NY, USA, 971–985. https://doi.org/10.1145/3297858.3304061

[4] Paul Alcorn. 2022. *AMD Working to Bring CXL Memory Tech to Future Consumer CPUs.* https://www.tomshardware.com/news/amd-working-to-bring-cxl-technology-to-consumer-cpus

[5] Duck-Ho Bae, Insoon Jo, Youra Adel Choi, Joo-Young Hwang, Sangyeun Cho, Dong-Gi Lee, and Jaeheon Jeong. 2018. 2B-SSD: The Case for Dual, Byte- and Block-Addressable Solid-State Drives. In *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, Los Angeles, CA, 425–438. https://doi.org/10.1109/ISCA.2018.00043

[6] San Chang. 2023. *Making the Case for CXL Native Memory.* https://conferenceconcepts.app.box.com/s/k5awbwpyxteq9u4r8a3ey75yfx16aue1

[7] CXL Consortium. 2023. *Compute Express Link Specification Revision 3.1.*

[8] Bill Gervasi. 2024. *FleX: Bringing CXL to the Motherboard.* https://files.futurememorystorage.com/proceedings/2024/20240807_CXLT-201-1_Gervasi.pdf

[9] Chien-Chung Ho, Sheng-Wei Cheng, Yuan-Hao Chang, Yu-Ming Chang, Sheng-Yen Hong, and Che-Wei Chang. 2015. Efficient hibernation resuming with classification-based prefetching scheme for embedded computing systems. *ACM SIGAPP Applied Computing Review* 15, 1 (2015), 33–43. https://doi.org/10.1145/2753060.2753064

[10] Intel Corporation. 2024. *Intel® 64 and IA-32 Architectures Software Developer's Manual.* https://cdrdv2.intel.com/v1/dl/getContent/671200

[11] Intel Corporation. 2024. *Intel® Core™ Ultra 200V Series Processors Datasheet, Volume 1 of 2.* https://www.intel.com/content/www/us/en/content-details/829568/intel-core-ultra-200v-series-processors-datasheet-volume-1-of-2.html

[12] Myoungsoo Jung. 2022. Hello bytes, bye blocks: PCIe storage meets compute express link for memory expansion (CXL-SSD). In *Proceedings of the 14th ACM Workshop on Hot Topics in Storage and File Systems* (Virtual Event) *(HotStorage '22)*. Association for Computing Machinery, New York, NY, USA, 45–51. https://doi.org/10.1145/3538643.3539745

[13] Hiroki Kaminaga. 2006. Improving linux startup time using software resume (and other techniques). In *Linux symposium*, Vol. 17. https://www.kernel.org/doc/mirror/ols2006v2.pdf#page=25

[14] Hyojeen Kim, Eunsam Kim, Jongmoo Choi, Donghee Lee, and Sam H Noh. 2011. Building fully functional instant on/off systems by making use of non-volatile RAM. In *2011 IEEE International Conference on Consumer Electronics (ICCE)*. IEEE, 675–676. https://doi.org/10.1109/ICCE.2011.5722803

[15] Shi-wu Lo, Wei-shiuan Tsai, Jeng-gang Lin, and Guan-shiung Cheng. 2010. Swap-before-hibernate: a time efficient method to suspend an OS to a flash drive. In *Proceedings of the 2010 ACM Symposium on Applied Computing*. 201–205. https://doi.org/10.1145/1774088.1774129

[16] Microsoft Corporation. 2020. *Modern Standby vs S3.* https://learn.microsoft.com/en-us/windows-hardware/design/device-experiences/modern-standby-vs-s3

[17] Rekha Pitchumani. 2023. *CMM-H (CXL Memory Module – Hybrid): Samsung's CXL-based SSD for the Memory-centric Computing Era.* Samsung. https://semiconductor.samsung.com/us/news-events/tech-blog/webinar-memory-semantic-ssd/

[18] Bernard Shung, San Chang, and Terry Cheng. 2023. *NVMe over CXL (NVMe-oC): An Ultimate Optimization of Host-Device Data Movement.* https://sc23.supercomputing.org/proceedings/exhibitor_forum/exhibitor_forum_files/exforum118s2-file2.pdf

[19] UEFI Forum, Inc. 2022. *Advanced Configuration and Power Interface (ACPI) Specification Release 6.5.* https://uefi.org/sites/default/files/resources/ACPI_Spec_6_5_Aug29.pdf

[20] UEFI Forum, Inc. 2024. *UEFI Platform Initialization Specification Version 1.8 Errata A.* https://uefi.org/sites/default/files/resources/PI_Spec_1_8_A_final_2024.03.05.pdf

[21] Shao-Peng Yang, Minjae Kim, Sanghyun Nam, Juhyung Park, Jin yong Choi, Eyee Hyun Nam, Eunji Lee, Sungjin Lee, and Bryan S. Kim. 2023. Overcoming the Memory Wall with CXL-Enabled SSDs. In *2023 USENIX Annual Technical Conference (USENIX ATC 23)*. USENIX Association, Boston, MA, 601–617. https://www.usenix.org/conference/atc23/presentation/yang-shao-peng

[22] Chenyang Zi, Chao Zhang, Qian Lin, Zhengwei Qi, and Shang Gao. 2013. Suspend-to-PCM: a new power-aware strategy for operating system's rapid suspend and resume. In *Emerging Technologies for Information Systems, Computing, and Management*. Springer, 667–674. https://doi.org/10.1007/978-1-4614-7010-6_75