The Non-Volatile Systems Laboratory Research Overview

Dr. Steven Swanson, Director
swanson@cs.ucsd.edu
http://nvsl.ucsd.edu

January 16, 2015

The Non-Volatile Systems Laboratory (NVSL) works to improve computer system performance, reliability, efficiency, and security by understanding emerging non-volatile memories (NVMs) and integrating them into computer systems. The NVSL's research agenda extends from hardware through the operating system and up through applications. Its work includes projects to build prototype solid-state storage devices [5, 4, 1, 8, 9, 20, 21, 2], develop programming models for NVMs [11, 25, 13], adapt and re-engineer applications to make better use of NVMs [10, 12, 3, 7, 6], characterize NVMs [14, 17, 23, 22, 16, 15], and explore the security implication of NVMs [24, 19, 18]. In addition, the NVSL helps develop the NVM research community by hosting the annual Non-Volatile Memories Workshop.

The sections below summarize our work in different areas. Then, we provide abstracts of selected publications and a full bibliography of our work. All of our papers are available at http://nvsl.ucsd.edu/index.php?path=pubs.

Programming and Using Non-Volatile Main Memory

NVMs on the processor's memory bus present many new challenges to programmers and system designers. We developed NV-Heaps [11], a persistent object system built for fast non-volatile main memories (NVMMs). NV-Heaps provides ACID transactions over non-volatile, pointer-based data structures and prevents several new types of bugs that can arise in NVMMs (e.g., pointers from volatile memory into non-volatile memory).

Deploying NVMMs in an enterprise environment presents additional challenges, since persistent data must be replicated to prevent data loss. Since network latencies can exceed write latencies for fast NVMMs, the cost of replication might squander NVMM performance. Our Mojim system [25] nearly eliminates the impact of network latency using an optimized RDMA-based protocol to replicate the contents of NVMM at one machine to another.

Building Next-Generation SSDs

We have constructed a series of SSD prototypes that explore the challenges and opportunities that NVMs present. The first of these prototypes was Moneta [4, 5, 8], an SSD built for next-generation NVMs such as PCM and STTM. Moneta combined streamlined hardware with a highly-optimized kernel subsystem that eliminate several bottlenecks in conventional storage software stacks. Moneta also allowed applications to access the SSD directly, without interacting with the file system, while still enforcing file system permissions [8]. As a result, Moneta approaches the minimum software overhead possible for SSD access. An invited paper in IEEE Computer [21] summarizes the insights of the Moneta prototypes.

Moneta's optimized architecture exposed application inefficiencies as the next target for optimization. We extended Moneta with database transaction support [10], caching operations [3], and key-value storage [12]. In each of these cases, we realized large application-level gains by customizing the SSD.

To provide more general extensibility, we built Willow [20], an easy-to-program solid-state drive. Willow allows the execution of untrusted code on the SSD, allows multiple extensions to be active simultaneously, and enforces file system protection. The Willow programming model allows developers to co-design applications with SSD semantics, leading to large application-level performance gains.

Other Moneta follow-on projects include QuickSAN [9] and Onyx [1]. QuickSAN added a high-speed ethernet connection so that one Moneta SSD could retrieve information directly from another SSD without needing to traverse a conventional network stack. The result was a distributed SAN architecture with minimal software overheads.

Onyx [1] was the first publicly demonstrated PCM SSD. The original Moneta prototype used DDR2 DRAM and a custom memory controller to emulate NVM bandwidth and latency. Onyx improved upon this by incorporating real PCM memory into Moneta by means of custom-built PCM memory card.

Security and Trust in Solid State Storage

Reliably and verifiably removing data from computer storage systems (i.e., *sanitizing* the storage), is critical to information security. NVMs present new challenges in this area since existing sanitization techniques and protocols served conventional disk drives.

The NVSL's work on solid-state sanitization [24] has had having a significant impact on data security policies in industry, government, and defense. We consult with the US Federal Government, the Navy, and the Coast Guard to verify the reliability of secure erasure commands. We also work with the National Associate for Information Destruction (an industry trade group) to develop standards for secure erasure.

Other work in the area of trust and security includes a technique for extracting unique fingerprints from flash devices [19] and extracting truly random data from DRAM in embedded systems [18].

Characterizing Non-Volatile Memories

NVMs exhibit wide variation in performance, power-efficiency, and reliability. Understanding that variability is necessary for researchers and engineers to design NVM-based storage devices that are as fast, as efficient, and as reliable as possible.

We published an early paper characterizing flash memories [14] and two papers examining the impact of power failure on data integrity in flash memory [23, 22]. We also developed an FTL that leverages flash memory's idiosyncrasies to improve performance and reliability [16].

The Non-Volatile Memories Workshop

Since 2010 we have helped organized the Non-Volatile Memories Workshop (NVMW: http://nvmw.ucsd.edu), an annual research meeting that builds and strengthens an NVM research community that spans the many areas ranging from semiconductor devices to system organization to software applications. The NVMW's technical program spans all these areas and provides a unique venue for networking, discussion, and collaboration. It attracts between 200-300 researchers and students. The workshop is a collaboration with the Center for Magnetic Recording Research (CMRR; http://cmrr.ucsd.edu).

Programming and Using Non-Volatile Main Memory

Mojim: A Reliable and Highly-Available Non-Volatile Memory System [25]

Abstract: Next-generation non-volatile memories (NVMs) promise DRAM-like performance, persistence, and high density. They can attach directly to processors to form non-volatile main memory (NVMM) and offer the opportunity to build very low-latency storage systems. These high-performance storage systems would be especially useful in large-scale data center environments where reliability and availability are critical. However, providing reliability and availability to NVMM is challenging, since the latency of data replication can dominate the low latency that NVMM should provide.

We propose *Mojim*, a system that provides the reliability and availability that large-scale storage systems require, while preserving the performance of NVMM. Mojim achieves these goals by using a two-tier architecture in which the primary tier contains a mirrored pair of nodes and the secondary tier contains one or more secondary backup nodes with weakly consistent copies of data. Mojim uses highly-optimized replication protocols, software, and networking stacks to minimize replication costs and expose as much of the NVMM's performance as possible. We evaluate Mojim using raw DRAM a proxy for NVMM and using commercial NVMM emulation system and find that Mojim provides replicated NVMM with similar or even better performance than unreplicated NVMM (reducing latency by 29% to 80% and delivering between 0.4 to 5× the throughput). We demonstrate that replacing MongoDB's built-in replication system with Mojim improves MongoDB's performance by between is 3.1 to 4×.

NV-Heaps: Making Persistent Objects Fast and Safe With Next-Generation, Non-Volatile Memories [11]

Abstract: Persistent, user-defined objects present an attractive abstraction for working with non-volatile program state. However, the slow speed of persistent storage (i.e., disk) has restricted their design and limited their performance. Fast, byte-addressable, non-volatile technologies, such as phase change memory, will remove this constraint and allow programmers to build high-performance, persistent data structures in non-volatile storage that is almost as fast as DRAM. Creating these data structures requires a system that is lightweight enough to expose the performance of the underlying memories but also ensures safety in the presence of application and system failures by avoiding familiar bugs such as dangling pointers, multiple free()s, and locking errors. In addition, the system must prevent new types of hard-to-find pointer safety bugs that only arise with persistent objects. These bugs are especially dangerous since any corruption they cause will be permanent.

We have implemented a lightweight, high-performance persistent object system called NV-heaps that provides transactional semantics while preventing these errors and providing a model for persistence that is easy to use and reason about. We implement search trees, hash tables, sparse graphs, and arrays using NV-heaps, BerkeleyDB, and Stasis. Our results show that NV-heap performance scales with thread count and that data structures implemented using NV-heaps out-perform BerkeleyDB and Stasis implementations by 32× and 244×, respectively, by avoiding the operating system and minimizing other software overheads. We also quantify the cost of enforcing the safety guarantees that NV-heaps provide and measure the costs of NV-heap primitive operations.

Building Next-Generation SSDs

Willow: A User-Programmable SSD [20]

Abstract: We explore the potential of making programmability a central feature of the SSD interface. Our prototype system, called Willow, allows programmers to augment and extend the semantics of an SSD with application-specific features without compromising file system protections. The SSD Apps running on Willow give applications low-latency, high-bandwidth access to the SSD's contents while reducing the load that IO processing places on the host processor. The programming model for SSD Apps provides great flexibility, supports the concurrent execution of multiple SSD Apps in Willow, and supports the execution of trusted code in Willow.

We demonstrate the effectiveness and flexibility of Willow by implementing six SSD Apps and measuring their performance. We find that defining SSD semantics in software is easy and beneficial, and that Willow makes it feasible for a wide range of IO-intensive applications to benefit from a customized SSD interface.

Moneta: A High-Performance Storage Array Architecture for Next-Generation, Non-volatile Memories [5]

Abstract: Emerging non-volatile memory technologies such as phase change memory (PCM) promise to increase storage system performance by a wide margin relative to both conventional disks and flash-based SSDs. Realizing this potential will require significant changes to the way systems interact with storage devices as well as a rethinking of the storage devices themselves. This paper describes the architecture of a prototype PCIe-attached storage array built from emulated PCM storage called *Moneta*. Moneta provides a carefully designed hardware/software interface that makes issuing and completing accesses atomic. The atomic management interface, combined with hardware scheduling optimizations, and an optimized storage stack increases performance for small, random accesses by 18x and reduces software overheads by 60%. Moneta array sustain 2.8 GB/s for sequential transfers and 541K random 4 KB IO operations per second (8× higher than a state-of-the-art flash-based SSD). Moneta can perform a 512-byte write in 9 μs (5.6× faster than the SSD). Moneta provides a harmonic mean speedup of 2.1× and a maximum speed up of 9× across a range of file system, paging, and database workloads. We also explore trade-offs in Moneta's architecture between performance, power, memory organization, and memory latency.

Providing Safe, User Space Access to Fast, Solid State Disks [8]

Abstract: Emerging fast, non-volatile memories (e.g., phase change memories, spin-torque MRAMs, and the memristor) reduce storage access latencies by an order of magnitude compared to state-of-the-art flash-based SSDs. This improved performance means that software overheads that had little impact on the performance of flash-based systems can present serious bottlenecks in systems that incorporate these new technologies. We describe a novel storage hardware and software architecture that nearly eliminates two sources of this overhead: Entering the kernel and performing file system permission checks. The new architecture provides a private, virtualized interface for each process and moves file system protection checks into hardware. As a result, applications can access file data without operating system intervention, eliminating OS and file system costs entirely for most accesses. We describe the support the system provides for fast permission checks in hardware, our approach to notifying applications when requests complete, and the small, easily portable changes required in the file system to support the new access model. Existing applications require no modification to use the new interface. We evaluate the performance of the system using a suite of microbenchmarks and database workloads and show that the new interface improves latency and bandwidth for 4 KB writes by 60% and 7.2×, respectively, OLTP database transaction throughput by up to 2.0×, and Berkeley-DB throughput by up to 5.7×. A streamlined asynchronous file IO interface built to fully utilize the new interface enables an additional 5.5× increase in throughput with 1 thread and 2.8× increase in efficiency for 512 B transfers.

From ARIES to MARS:Transaction Support for Next-Generation Solid-State Drives [10]

Abstract: Transaction-based systems often rely on write-ahead logging (WAL) algorithms designed to maximize performance on disk-based storage. However, emerging fast, byte-addressable, non-volatile memory (NVM) technologies (e.g., phase-change memories, spin-transfer torque MRAMs, and the memristor) present very different performance characteristics, so blithely applying existing algorithms can lead to disappointing performance.

This paper presents a novel storage primitive, called Editable Atomic Writes (EAWs), that enables sophisticated, highly-optimized WAL schemes in fast NVM-based storage systems. EAWs allow applications to safely access and modify log contents rather than treating the log as an append-only, write-only data structure, and we demonstrate that this can make implementing complex transactions simpler and more efficient. We use EAWSs to build MARS, a WAL scheme that provides the same as features ARIES (a widely-used WAL system for databases) but avoids making disk-centric implementation decisions.

We have implemented EAWs and MARS in a next-generation SSD to demonstrate that the overhead of EAWs is minimal compared to normal writes, and that they provide large speedups for transactional updates to hash tables, B+trees, and large graphs. In addition, MARS outperforms ARIES by up to 3.7$\times$ while reducing software complexity.

Onyx: A Prototype Phase-Change Memory Storage Array [1]

Abstract: We describe a prototype high-performance solid-state drive based on first-generation phase-change memory (PCM) devices called *Onyx*. Onyx has a capacity of 10 GB and connects to the host system via PCIe. We describe the internal architecture of Onyx including the PCM memory modules we constructed and the FPGA-based controller that manages them. Onyx can perform a 4 KB random read in 38 μs and sustain 191K 4 KB read IO operations per second. A 4 KB write requires 179 μs. We describe our experience tuning the Onyx system to reduce the cost of wear-leveling and increase performance. We find that Onyx out-performs a state-of-the-art flash-based SSD for small writes ($<$ 2 KB) by between 72 and 120% and for reads of all sizes. In addition, Onyx incurs 20-51% less CPU overhead per IOP for small requests. Combined, our results demonstrate that even first-generation PCM SSDs can out-perform flash-based arrays for the irregular (and frequently read-dominated) access patterns that define many of today's "killer" storage applications. Next generation PCM devices will widen the performance gap further and set the stage for PCM becoming a serious flash competitor in many applications.

Security and Trust in Solid State Storage

Reliably Erasing Data From Flash-based Solid State Drives [24]

Abstract: Reliably erasing data from storage media (*sanitizing* the media) is a critical component of secure data management. While sanitizing entire disks and individual files is well-understood for hard drives, flash-based solid state disks have a very different internal architecture, so it is unclear whether hard drive techniques will work for SSDs as well.

We empirically evaluate the effectiveness of hard drive-oriented techniques and of the SSDs' built-in sanitization commands by extracting raw data from the SSD's flash chips after applying these techniques and commands. Our results lead to three conclusions: First, built-in commands are effective, but manufacturers sometimes implement them incorrectly. Second, overwriting the entire visible address space of an SSD twice is usually, but not always, sufficient to sanitize the drive. Third, none of the existing hard drive-oriented techniques for individual file sanitization are effective on SSDs.

This third conclusion leads us to develop flash translation layer extensions that exploit the details of flash memory's behavior to efficiently support file sanitization. Overall, we find that reliable SSD sanitization requires built-in, verifiable sanitize operations.

Extracting Device Fingerprints from Flash Memory by Exploiting Physical Variations [19]

Abstract: We evaluate seven techniques for extracting unique signatures from NAND flash devices based on observable effects of process variation. Four of the techniques yield usable signatures that represent different trade-offs between speed, robustness, randomness, and wear imposed on the flash device. We describe how to use the signatures to prevent counterfeiting and uniquely identify and/or authenticate electronic devices.

Characterizing Non-Volatile Memories

Characterizing Flash Memory: Anomalies, Observations, and Applications [14]

Abstract: Despite flash memory's promise, it suffers from many idiosyncrasies such as limited durability, data integrity problems, and asymmetry in operation granularity. As architects, we aim to find ways to overcome these idiosyncrasies while exploiting flash memory's useful characteristics. To be successful, we must understand the tradeoffs between the performance, cost (in both power and dollars), and reliability of flash memory. In addition, we must understand how different usage patterns affect these characteristics. Flash manufacturers provide conservative guidelines about these metrics, and this lack of detail makes it difficult to design systems that fully exploit flash memory's capabilities. We have empirically characterized flash memory technology from five manufacturers by directly measuring the performance, power, and reliability. We demonstrate that performance varies significantly between vendors, devices, and from publicly available datasheets. We also demonstrate and quantify some unexpected device characteristics and show how we can use them to improve responsiveness and energy consumption of solid state disks by 44% and 13%, respectively, as well as increase flash device lifetime by 5.2x.

Understanding the Impact of Power Loss on Flash Memory [22]

Abstract: Flash memory is quickly becoming a common component in computer systems ranging from music players to mission-critical server systems. As flash plays a more important role, data integrity in flash memories becomes a critical question. This paper examines one aspect of that data integrity by measuring the types of errors that occur when power fails during a flash memory operation. Our findings demonstrate that power failure can lead to several non-intuitive behaviors. We find that increasing the time before power failure does not always reduce error rates and that a power failure during a program operation can corrupt data that a previous, successful program operation wrote to the device. Our data also show that interrupted program operations leave data more susceptible to read disturb and increase the probability that the programmed data will decay over time. Finally, we show that incomplete erase operations make future program operations to the same block unreliable.

Bibliography

[1] Ameen Akel, Adrian M. Caulfield, Todor I. Mollov, Rajesh K. Gupta, and Steven Swanson. Onyx: A prototype phase-change memory storage array. In *Proceedings of the 3rd USENIX conference on Hot topics in storage and file systems*, HotStorage'11, pages 1–5. USENIX Association, 2011.

[2] Rajeev Balasubramonian, Jichuan Chang, Troy Manning, Jaime H. Moreno, Richard Murphy, Ravi Nair, and Steven Swanson. Near-data processing: Insights from a micro-46 workshop. *Micro, IEEE*, 34(4):36–42, July 2014.

[3] Meenakshi Sundaram Bhaskaran, Jian Xu, and Steven Swanson. Bankshot: Caching slow storage in fast non-volatile memory. In *1st Workshop on Interactions of NVM/Flash with Operating Systems and Workloads*, INFLOW'13, 2013.

[4] Adrian M. Caulfield, Joel Coburn, Toder I. Mollov, Arup De, Ameen Akel, Jiahua He, Arun Jagatheesan, Rajesh K. Gupta, Allan Snavely, and Steven Swanson. Understanding the Impact of Emerging Non-Volatile Memories on High-Performance, IO-Intensive Computing. In *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '10, pages 1–11, Washington, DC, USA, 2010. IEEE Computer Society. Nominated for Best Technical Student Paper.

[5] Adrian M. Caulfield, Arup De, Joel Coburn, Todor I. Mollov, Rajesh K. Gupta, and Steven Swanson. Moneta: A high-performance storage array architecture for next-generation, non-volatile memories. In *Proceedings of the 2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO '43, pages 385–395, Washington, DC, USA, 2010. IEEE Computer Society.

[6] Adrian M. Caulfield, Laura M. Grupp, and Steven Swanson. Gordon: using flash memory to build fast, power-efficient clusters for data-intensive applications. In *ASPLOS '09: Proceeding of the 14th international conference on Architectural support for programming languages and operating systems*, pages 217–228, New York, NY, USA, 2009. ACM. Selected as an IEEE Micro TopPick.

[7] Adrian M. Caulfield, Laura M. Grupp, and Steven Swanson. Gordon: An improved architecture for data-intensive applications. *IEEE Micro*, 30:121–130, 2010. IEEE Micro Top Picks.

[8] Adrian M. Caulfield, Todor I. Mollov, Louis Eisner, Arup De, Joel Coburn, and Steven Swanson. Providing Safe, User Space Access to Fast, Solid State Disks. In *Proceeding of the 17th international conference on Architectural support for programming languages and operating systems*, New York, NY, USA, March 2012. ACM.

[9] Adrian M. Caulfield and Steven Swanson. QuickSAN: A Storage Area Network for Fast, Distributed, Solid State Disks. In *ISCA '13: Proceeding of the 40th Annual International Symposium on Computer Architecture*, pages 1–11, New York, NY, USA, June 2013. ACM.

[10] Joel Coburn, Trevor Bunker, Meir Shwarz, Rajesh K. Gupta, and Steven Swanson. From ARIES to MARS:transaction support for next-generation solid-state drives. In *Proceedings of the 24th International Symposium on Operating Systems Principles (SOSP)*, 2013.

[11] Joel Coburn, Adrian M. Caulfield, Ameen Akel, Laura M. Grupp, Rajesh K. Gupta, Ranjit Jhala, and Steven Swanson. Nv-heaps: Making persistent objects fast and safe with next-generation, non-volatile memories. In *Proceedings of the sixteenth international conference on Architectural support for programming languages and operating systems*, ASPLOS '11, pages 105–118. ACM, 2011.

[12] Arup De, Maya Gokhale, Rajesh Gupta, and Steven Swanson. Minerva: Accelerating data analysis in next-generation ssds. In *Proceedings of The 21st IEEE International Symposium on Field-Programmable Custom Computing Machines*, pages 1–8, 2013.

[13] Louis Alex Eisner, Todor Mollov, and Steven Swanson. Quill: Exploiting fast non-volatile memory by transparently bypassing the file system. Technical Report CS2013-0991, Department of Computer Science & Engineering, University of California, San Diego, Jan 2013.

[14] Laura M. Grupp, Adrian M. Caulfield, Joel Coburn, Steven Swanson, Eitan Yaakobi, Paul H. Siegel, and Jack K. Wolf. Characterizing flash memory: anomalies, observations, and applications. In *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 42, pages 24–33, New York, NY, USA, 2009. ACM.

[15] Laura M. Grupp, John D. Davis, and Steven Swanson. The bleak future of nand flash memory. In *Proceedings of the 10th USENIX conference on file and storage technologies*, FAST'12, pages 1–8. USENIX Association, 2012.

[16] Laura M. Grupp, John D. Davis, and Steven Swanson. The Harey Tortoise: Managing Heterogeneous Write Performance in SSDs. In *Proceedings of the 2013 USENIX Annual Technical Conference*, USENIX ATC'13,

pages 1–12, Berkeley, CA, USA, 2013. USENIX Association.

[17] V. Mohan, T. Bunker, L. Grupp, S. Gurumurthi, M.R. Stan, and S. Swanson. Modeling power consumption of nand flash memories using flashpower. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 32(7):1031–1044, 2013.

[18] Keaton Mowery, Michael Wei, David Kohlbrenner, Hovav Shacham, and Steven Swanson. Welcome to the entropics: Boot-time entropy in embedded devices. In *IEEE Symposium on Security and Privacy (Oakland 2013)*, pages 1–15.

[19] Pravin Prabhu, Ameen Akel, Laura Grupp, Wing-Key Yu, G. Edward Suh, Edwin Kan, and Steven Swanson. Extracting device fingerprints from flash memory by exploiting physical variations. In *Proceedings of the 4th International Conference on Trust and Trustworthy Computing*, pages 1–17, 2011.

[20] Sudharsan Seshadri, Mark Gahagan, Sundaram Bhaskaran, Trevor Bunker, Arup De, Yanqin Jin, Yang Liu, and Steven Swanson. Willow: A user-programmable ssd. In *Proceedings of the 11th USENIX Symposium on Operating Systems Design and Implementation (OSDI '14)*, 2014.

[21] Steven Swanson and Adrian M. Caulfield. Refactor, reduce, recycle: Restructuring the i/o stack for the future of storage. *Computer*, 46(8):52–59, 2013.

[22] Hung-Wei Tseng, Laura M. Grupp, and Steven Swanson. Understanding the impact of power loss on flash memory. In *48th Design Automation Conference (DAC 2011)*, pages 1–6, June 2011.

[23] Hung-Wei Tseng, Laura M. Grupp, and Steven Swanson. Underpowering nand flash: Profits and perils. In *50th Design Automation Conference (DAC 2013)*, pages 1–6, June 2013.

[24] Michael Wei, Laura M. Grupp, Frederick E. Spada, and Steven Swanson. Reliably erasing data from flash-based solid state drives. In *Proceedings of the 9th USENIX conference on File and storage technologies*, FAST'11, pages 1–13, Berkeley, CA, USA, 2011. USENIX Association.

[25] Yiying Zhang, Jian Yang, Amirsaman Memaripour, and Steven Swanson. Mojim: A reliable and highly-available non-volatile memory system. In *Proceedings of the 20th International Conference on Architectural Support for Programming Languages and Operating Systems*, 2015.