Research Statement

Laura Grupp

My approach to architecture's goal of providing a fast, efficient and reliable foundation for computing systems is to leverage the unexploited potential of new technologies. This requires characterization of unpublished behavior, knowledge of technology-specific management techniques and study of effects on system-level performance. I have come to this approach through the study and characterization of NAND flash memory chips and the redesign of the management layer hosted by the controller in Solid State Drives (SSDs). However, these skills will apply more broadly to newer technologies as well as to new algorithms for existing technologies.

My work on characterization and application- and device-specific enhancements to the management of flash memory is helping the entire flash community navigate an interesting and challenging point in the development of flash memory. Collectively, the community has established the basics of working with the technology but will need creative, well informed designs to make further significant advances in the face of negative scaling trends. The primary driving force in the manufacture of new generations of flash technology is to increase its density through transistor scaling and by adding bits to each memory element. However, with increasing density we have observed consistent and significant decreases in reliability, performance, energy efficiency and lifetime [1]. To help mitigate these trends, companies who design SSDs are all pushing past the vetted, published capabilities of flash memory, but keeping nearly all of their advancements as trade secrets. Published manufacturers specifications say very little about what the technology can do, but they are all that is available to academics. My characterization work is the vehicle by which I am cultivating the academic community's understanding of flash's abilities and improving our algorithms for managing the technology.

The information manufacturers do publish show a very idiosyncratic interface, which conveys complexity to how we manage the technology. In addition to the read and write operations necessary in memory and storage, flash exposes an erase operation whereby no page can be written multiple times between erases. Furthermore, reads and writes operate on flash pages, while erases function at the granularity of a flash block – a unit consisting of dozens to hundreds of pages. To manage these two constraints while allowing page-sized accesses, SSDs employ a thick layer of management algorithms called the Flash Translation Layer (FTL) in the SSD's controller. The primary purpose of this layer is to maintain a map between logical and physical addresses and provide all other algorithms to operate on the flash in a log-structured fashion while exposing a linear logical address space.

In the following paragraphs I will describe my work on flash memory in the areas of characterization, performance and security, and describe the work I intend to pursue next.

Gaining Device Level Insight

I have served as the lead student on our lab's flash characterization effort, helping our lab and the broader research community better understand how flash chips work, and how we can leverage these behaviors for more efficient flash-based storage. This has involved studying literature regarding decisions made in the design of chips, characterizing commercial products to reverse engineer which ideas are used in practice, discussing my discoveries with chip and SSD manufacturers, mentoring less experienced students as they complete new research and collaborating with experts in other fields to perform research that otherwise would not have been possible. Enabling all this work is our in-house characterization platform called Ming the Merciless and Mongo FTL simulator [2, 3]. With these tools, we systematically collect performance, power and reliability information under a variety of scenarios and a mix of operations on a reasonable cross sections of chips from different manufacturers to ultimately propose and prototype changes to the FTL.

We have found a great deal of unexploited potential and hidden behaviors which provide far more flexibility than the academic community originally realized. With my proficiency in flash and my colleagues' expertise in coding algorithms we extend the limits on the reliable use of flash, first into longer lifetime through the use of graduated ECC [4] and then into higher density parts [5, 6, 7]. We also explored the effects of partial and full power loss and detected several non-intuitive behaviors which could lead to unexpected data corruption [8]. Finally, we discovered a way to extract a unique signature from each chip using the variations in the silicon of the flash chip, which we can use to secure devices that include flash dies, and to generate random numbers [9]. Through these mentorships and collaborations, I enjoyed enabling work that I would not have been able to complete independently.

Designing High Performance FTLs

Most of my work focuses primarily on improving the performance of flash-based systems through incorporating application- and device-level insight into the management layer. The first of these proposals relieved the bottleneck in the FTL for big data applications running on a system called Gordon [10, 11]. Gordon is a data-centric cluster architecture which achieves low power and high performance through the use of Atom processors and NAND flash memory. We characterized our map-reduce workloads looking for how to improve flash performance and found they produced large access sizes and that the parallelism in our flash array was not fully utilized. I presented proposals in the design of Gordon's FTL which allow a direct path to increasing the array's bandwidth by using all of the chips within the array both for servicing page-sized segments of a large access in parallel and for performing multiple inflight operations simultaneously.

With Gordon, we learned to exploit the parallelism in large SSDs, but I have also developed techniques to exploit the characteristics of individual chips to provide lower latency. Flash is a technology which supports storing multiple bits in each memory element, considerably decreasing the memory's cost per bit. Unfortunately, all other metrics experience significant decline with each added bit. However, one of the most exciting discoveries of my characterization work is that these declines are confined to the new capacity. For example, in two-bit flash the half of the pages comprised of the first bit complete a write nearly as fast as single-bit technology, while the other half (comprised of the second bit) take over four times that time [2]. Manufacturer specifications for two-bit technology imply a moderate latency for all pages, but our observations show a systematic, predictable duality in performance. Furthermore, the pages which are fast to write are also fast to read and use significantly less energy to write. Finally, I have observed these trends scaling up to TLC and careful study of literature on device level physics shows the trend is extremely likely to continue. For these reasons I found great utility in proposing a small modification to FTL designs to expose these properties. This opens up a wealth of opportunity for significantly improving the latency of high density flash-based systems.

My first proposal at harnessing low-latency pages exposes a priority bit to applications so the FTL can simply match up the high priority accesses to the well performing areas. This mechanism could be applied to improve performance for a selection of accesses – when we applied it to speed up swap accesses, the performance of our benchmarks improved by an average of 44%. It could also apply to mobile devices – while they are on battery power all accesses are high priority so the FTL sends them to the energy efficient pages.

While locating these fast and slow pages is very straightforward and consistent across all manufacturers, accessing them efficiently under the constraints imposed by the chips themselves is far more challenging. The chip requires that we access pages in order in each block but when accessed in order the fast and slow page pages alternate. This means that when we need to access a fast page the FTL may skip over a slow page, which it cannot use until the log-structured algorithm used to access the flash returns to this area to clean and reuse it. In the worst case this reduces the capacity of the chip from two-bit technology to single-bit technology. Because of the popularity of two-bit flash, it is less than half the cost per bit of single-bit flash – making such a design a smart financial choice, but we can do better. My latest work focuses on finding the correct FTL design, in order to provide the high performance of single-bit technology with two-bit technology and while minimizing loss of performance or capacity.

My approach takes advantage of the "burstiness" present in many workloads which we are able to service completely with fast pages. The FTL design tracks many write points within the flash array, which move through each block in order alternately pointing to fast and slow pages. During periods of inactivity, the FTL performs garbage collection and places data from internal moves on slow pages whenever they are available, and continues the garbage collection until there is enough free space in the array and all of the write points point to fast pages. In this way, we soak up all slow areas and absorb the following burst of traffic with fast pages only. Simulation results show no added wear, and up to 100% of optimal performance.

With bursty traffic accommodated, I next have turned my attention to accommodating sustained traffic with high performance whose degradation is graceful rather than the erratic behavior often observed in current designs. This is a challenge, because FTLs and workloads combine in many ways to create unpredictable behavior which can rapidly get out of hand. For example, if the FTL provides high performance by neglecting to clean up half-used blocks it may find itself in the position of having to suddenly block all incoming accesses. A blocked access may be a rare event but is terribly destructive, creating a long tail in the distribution of performance provided by the SSD and eclipsing all the benefits of flash technology. In order to create a design which avoids this long tail, I have developed a linear model of FTL operation. I will allow me to rapidly explore the relationships between the key settings in the FTL and their interactions with a variety of workloads.

Securing the Data in Flash Devices

In addition to improving the performance of flash-based SSDs, my characterization work has enabled far more efficient means of securing the data we store in them. SSDs have been designed as a drop-in replacement for hard disk drives (HDDs), but not all existing techniques for HDDs transfer directly to SSDs. Guaranteed sanitization of the data stored on a drive is one tool which must be designed differently, because SSDs' internal structure is very different from that of an HDD. In SSDs, each write moves the logical address to a new physical location leaving old copies of the data in the previous physical position. This old data remains available to anyone with physical access to the raw flash chips. In order to conclusively remove the page of stale data, the page must be erased, typically when the FTL moves data from partially-used blocks and erases the remaining stale data. But before the erase can proceed, all the valid data from that page's block must be copied to another part of the array. In the worst case, the algorithm will need to perform such an erase with each write to maintain a "clean" drive to, for example, support rapid and/or unexpected cleansing of individual files on the drive.

My characterization work allows us to push beyond the bounds defined in the chips' specifications to expose a page-size erase command called *Scrub* [12], which enables far more efficient single-file sanitization techniques. It is essentially a second write to a page, which is disallowed in the chips' specifications. However, my characterization work established the bounds of acceptable use for Scrub which performs a very well specified second write to the given page. Scrub provides a tool for SSD designers to implement simple and effective secure erase commands which execute 125x faster than the same commands without scrub. Such algorithms would otherwise be cripplingly slow, and a more complicated or less immediately effective solution would be necessary.

Future Directions

The contributions of my completed and current work focus on enhancing the management techniques the storage community has put in place for flash systems. For my future work, I intend to concentrate on creating more flexibility in SSDs' interfaces in order to facilitate more productive use of these and other flash management algorithms. To date, the community has focused its efforts on making flash useable in the larger computing structure by developing complex algorithms to accommodate flash's unusual access restrictions while making the drive appear like an HDD. While this interface has promoted the introduction of flash-based devices, it leads to duplicated and contradictory algorithms on either side of the interface and hinders the synchronization of application and device characteristics. Furthermore, we

have found that providing a management layer design with a particular application in mind improves the efficiency of the drive by removing these costly overheads.

Moving forward, I intend to redefine the interface to SSDs in order to support and promote the creation of application-specific algorithms using one or more of the following approaches.

Evolution of the Storage Interface

Through my characterization work, we have developed a large set of mechanisms by which the management layer can trade-off between latency, bandwidth, reliability, density, lifetime, retention time and energy. Thus far, most FTLs exploit these trade-offs through, for example, observing the limited information from access streams coming in to the SSD and working based on predictions. However, with many of these mechanisms now in place, the time is right for us to bridge the divide between the mechanisms and applications' access to them.

Beginning with a taxonomy of the fundamental set of trade-offs in flash and the set of access types which require unique combinations of characteristics from the storage array, we can propose minimal but impactful modifications to SSDs' interface without requiring radical system-wide changes. Our proposed priority bit (described above) is one such modification, where we give applications access to fast and energy efficient areas in the array by adding a single, optional bit to each access command. Such modifications provide access for the applications to tap into the trade-off mechanisms in the flash through evolution of the storage interface, rather than a complete re-build which could break backward compatibility.

I propose adding a rich collection of metadata to each access, which would indicate the access's unique combinations of choices for the trade-offs available from the flash array. The metadata may give different weights to different properties, specify only a subset of those available and tap in to each as either distinct options (fast or slow) or as a gradient (a range in error rate). The two keys are to design the right combination of trade-offs in the flash for each type of access and to create the correct structures and algorithms to enable all of these trade-offs simultaneously within the FTL.

NAND Flash Core Management Layer

Given the idiosyncrasies of interfacing with flash chips, and the complex algorithms developed to run on top of a flash array, another possible avenue is to separate the most basic management functionality into a minimal core. The goal is to give all the flexibility to the algorithms which sit on top of the flash while hiding the unpleasantness of the technology. One of the primary questions is *what is the basic set of functionality that a flash-based storage system must have?* This minimal set may include algorithms to perform wear leveling and ECC but exclude the heavy and many-varied approaches to logical to physical address mapping. The primary challenges will be in finding the universal set of algorithms to factor into the core management layer and designing an efficient interface to them.

The core could give management algorithms a cleaner, better organized interface to the properties and tradeoffs we are currently discovering. It would expose properties such as fast and slow pages, pages with higher and lower retention time and a scrub-like command which accommodates its limitations automatically. With the appropriate interface definition, a different core could be defined for different storage technologies such as PCM based SSDs or hybrid drives.

The core management layer enables drives which are much more versatile and, with the correct coordination with higher-level management layers, more light weight. Logical, application-specific SSDs could plug in to the core as an interface to the storage. It may even be possible to attach multiple virtual SSDs to the same pool of storage.

Our awareness of the trade-offs available in flash as well as how to use this information is still in flux, and so we may find that some applications would be more efficient with greater flexibility in the storage system than can be offered by a core management layer. In this case, a toolbox which allows full access to the raw flash array in a systems-level language would be a very useful tool. Such a system would have commands which expose all of the mechanisms for controlling the trade-

offs available in the flash array in a consistent manner so the system designer could design his or her own means, for example, of organizing the physical placement of the logical addresses and of leveraging the trade-offs in the array, without an insurmountable learning curve.

At this time in the development of flash, these tools for systems-level FTL designers would be particularly effective. For HDDs, we have witnessed system-level storage algorithms adapt to the most simplistic behaviors unique to HDDs (the spin-up and seek times, for example), while more subtle behaviors were left unexploited, forgotten and inaccessible. Flash is still relatively young, with many trade-offs that have nearly equal accessibility. By creating a platform in which all of these trade-offs can be exposed to system level designers, we will allow exploration of even the most esoteric trade-offs based on their potential rather than their simplicity, facilitate straightforward introduction of new trade-offs to the FTL designers' community and enable a wide variety of SSD designs well into the future.

Bibliography

[1] L. M. Grupp, J. D. Davis and S. Swanson, "The Bleak Future of NAND Flash Memory," in *Proceedings of the 10th USENIX conference on file and storage technologies*, San Jose, CA, 2012.

[2] L. M. Grupp, A. M. Caulfield, J. Coburn, S. Swanson, E. Yaakobi, P. H. Siegel and J. K. Wolf, "Characterizing Flash Memory: Anomalies, Observations, and Applications," in *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, New York, NY, 2009.

[3] L. M. Grupp, A. M. Caulfield, J. Coburn, J. D. Davis and S. Swanson, "Beyong the Datasheet: Using Test Beds to Probe Non-Volatile Memories' Dark Secrets," in *IEEE Globecom 2010 Workshop on Application of Communication Theory to Emerging Memory Technologies (ACTEMT 2010)*, 2010.

[4] E. Yaakobi, P. H. Siegel, S. Swanson, J. K. Wolf, L. M. Grupp and J. Ma, "Error Characterization and Coding Schemes for Flash Memories," in *IEEE Globecom 2010 Workshop on Application of Communication Theory to Emerging Memory Technologies (ACTEMT 2010)*, 2010.

[5] R. Gabrys, E. Yaakobi, L. M. Grupp, S. Swanson and L. Dolecek, "Tackling Intracell Variability in TLC Flash Through Tensor Product Codes," in *International Symposium on Information Theory*, 2012.

[6] E. Yaakobi, L. M. Grupp, P. H. Siegel, S. Swanson and J. K. Wolf, "Characterization and Error-Correcting Codes for TLC Flash Memories," in *Appear in International Conference on Computing, Networking and Communications, Data Storage Technology and Applications Symposium*, 2012.

[7] R. Gabrys, L. M. Grupp, S. Swanson and L. Dolecek, "Tackling Temporal Variability in Multilevel Flash: New Error-Control Code Design and Architectural Validation," in *Invited Talk, Forty-Ninth Annual Allerton Conference*, 2011.

[8] H.-W. Tseng, L. M. Grupp and S. Swanson, "Understanding the Impact of Power Loss on Flash Memory," in *48th Design Automation Conference (DAC 2011)*, 2011.

[9] P. Prabhu, A. Akel, L. M. Grupp, W.-K. G. Yu, E. Suh, E. Kan and S. Swanson, "Extracting Device Fingerprints from Flash Memory by Exploiting Physical Variations," in *Proceedings of the 4th International Conference on Trust and Trustworthy Computing*, 2011.

[10] A. M. Caulfield, L. M. Grupp and S. 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, 2009. (Selected as an IEEE Micro TopPick).* , Washington D.C., 2009.

[11] A. M. Caulfield, L. M. Grupp and S. Swanson, "Gordon: An Improved Architecture for Data-Intensive Applications," in *IEEE Micro Top Picks*, 2010.

[12] M. Wei, L. M. Grupp, F. E. Spada and S. Swanson, "Reliably Erasing Data from Flash-Based Solid State Drives," in *Proceedings of the 9th USENIX conference on File and storage technologies*, San Jose, 2011.