The Role of Non-Volatile Memory from an Application Perspective

Brett M. Kettering and James A. Nunez
Department of Defense

***Abstract* - Current, emerging, and future NVM (non-volatile memory) technologies give us hope that we will be able to architect HPC (high performance computing) systems that initially use them in a memory and storage hierarchy, and eventually use them as the memory and storage for the system, complete with ownership and protections as a HDD-based (hard-disk-drive-based) file system provides today.**

I. INTRODUCTION

HPC (high performance computing) applications spend a great deal of time waiting for data. HDD (hard disk drive) technology advances have not kept pace with those of other HPC system components, namely the CPU. Consequently, in order to reduce the amount of time that an application waits, I/O engineers have developed subsystems that implement technological stop gaps. However, these stop gaps involve using more hardware, which consumes more power and, of course, increases cost. This approach is not sustainable in any of these parameters.

More hardware means more money is required to procure, house, and run the I/O subsystem. We all live within a financial budget and HPC data centers are no exception. We must find a way to control costs. Furthermore, having more hardware components presents a serious resilience problem where there is an increasing probability that some component in the I/O subsystem will fail forcing even more stop-gap solutions to accommodate the ever-more-certain failure. Applications that run for longer than the MTTF (mean time to failure) of the HPC system must save state in the form of file system I/O to make progress. Some researchers have predicted that on future systems, HPC applications will spend all their time saving state and have no time for computation. The mathematics was presented in [1]. John T. Daly, the author, and others have since used this mathematical foundation to make such predictions. It is also possible that system failures will not be recoverable and we won't be able to complete the I/O in order to save state.

There is limited power available to run HPC data centers and they must live within a power budget. In turn, HPC storage systems must also live within a share of that budget. Adding more and more hardware, in particular storage components that constantly draw power, like HDDs, that uses more and more power is not a sustainable solution.

Thus, we must find a solution that controls costs while it reduces the hardware resilience problem, uses less power, and significantly improves performance. This has been the focus of our research. The innovative use of NVM (non-volatile memory) in HPC system architectures will achieve this goal. In this paper we will discuss the nature of how our applications currently do I/O, how we believe NVM can be evolutionarily integrated into HPC systems so that we see I/O performance improvements without changing how our applications do I/O, and finally how we envision NVM being revolutionarily integrated into HPC systems that will change how HPC applications

do I/O, but provide such great advantages that HPC application developers will be willing to make the necessary changes.

II. HPC APPLICATION I/O TYPES

To highest order, the large-scale compute intensive applications that characterize many of the workloads found in simulation and predictive science communities are characterized by data expansion. The platforms designed to build and run such applications can be thought of as experimental facilities for running very large numerical experiments that generate copious data. By way of contrast, the large-scale data intensive applications that characterize workloads for various types of data analytics are characterized by data reduction. The platforms designed to compute in this application space must comb through large volumes of data looking for trends, correlations, and anomalies.

Some HPC applications frequently perform defensive I/O; the state of DRAM is written to persistent storage in case the application or system crashes and we need to restart from the last dump. This is usually a restart dump for each processor core (2 - 4 GB per core, commonly 100's, occasionally 10's of thousands and Exacale will use 100's of thousands), generally with two dumps in the file system at any given point. The application will also do dumps (2 - 4 GB) to show state as the simulation progresses that can be used to create visualizations at a later date. In general, application users are willing to give a maximum of 10% of run-time to doing this I/O. They'd prefer it be less. Increasing the I/O performance of the storage devices enables an application to use the 10% run-time budget for I/O to do more frequent dumps so that the steps of a simulation can be examined at a finer level. More data points allow higher-confidence interpolation for what happened between the dumps. There is a claim that 3D PCM (phase change memory) mechanisms can make defensive I/O overhead for an Exascale system be <4%, whereas with HDD a Petascale system is 25% or more [2].

An increasingly important function of a system is to reason about its input and provide an output that is more insightful. Such systems transform data into information; information into knowledge; and knowledge into domain-specific intelligence. A series of applications will be run on an initial input of data. Each application does some sort of transformation to the input and produces an output that the next application uses. Sometimes the intermediate transformations are retained and sometimes they are not. The volumes of data are going to be similar to those detailed in the discussion of defensive I/O. Even today, such data-producing applications can produce O(10PB) per year [3].

DISC (data-intensive supercomputing) systems are characterized by processing large amounts of data that is produced by a variety of sensors. This data is streamed-in and must be mined for important elements. Those mined elements must be reported to a responder, categorized and saved for later analysis, or archived to storage. These systems are dominated by random IOPS (I/O operations per second). The general model of this type of application is data reduction. However, depending on the data retention requirements imposed on the organization, the volume of data can grow to significant sizes. One can easily imagine applications that consume O(PB) or even O(EB) per year and reduce that to a manageable amount.

It is easy to envision that many of these I/O models will require the use of databases. Each of

these databases must provide for queries that are used to make these transformations and to provide the lower-level information for further examination by the querying entity.

III. NVM ADVANTAGES OVER FLASH & DRAM

Future NVMs will not have the read/write imbalance of PCM and will have better scaling, endurance, and performance properties. Consequently, they will be even more advantageous than PCM.

PCM is bit-alterable. A huge amount of system overhead relative to flash is eliminated because it is not necessary to first erase and then rewrite an entire block when only a bit changes value [4].

The write throughput of PCM is faster than NOR and NAND flash, perhaps because the phase change is fast and self-limiting. Flash programming is not self-limiting, and must be done slowly and monitored [4].

PCM can be used as code execution memory because it is non-volatile, its read latency is that of NOR flash, and its read bandwidth that of DRAM [4].

PCM write cycle endurance is better than flash at O(10^6), with failure usually resulting from the heating element separating from the material. Other NVMs are projected to have even higher write cycle endurance. As an added bonus, it is radiation hard; whereas ionizing radiation can disturb the charge in DRAM and SRAM [4].

PCM scaling doesn't seem to be limited by the chalcogenide/heater system itself. Achieving useful densities in the multi-gigabit range will required changes to the access element. Ovonyx has already provided a possible solution in the ovonic threshold switch, which was used to demonstrate a stackable 3-D PCM array [4].

It is not necessary to short-stroke PCM for performance like for HDD. Hitachi recommends using the first 3 outer zones of the HDD, which accounts for 10% of its capacity and leaves 90% unused [5].

Vibration does not affect PCM [6].

Vibration increases HDD power consumption. A Sun study showed that it can be by as much as 2.65x (from 1.7 KWh to 4.5 KWh for a 10 terabyte write).

Vibration reduces HDD performance. A Sun study showed that it can be reduced by as much as 2.67x (40MB/s down to 15MB/s) for sequential I/O using SATA.

A Q-Associates study showed that by using a vibration-dampening rack by Green Platform, random read IOPS increased by 56% - 246% and random write IOPS increased by 34% - 88%. Overall results showed throughput increased by a maximum of 250%. The Green Platform rack cost 4x - 5x a normal rack, so the additional cost per rack was from $6,000 - $8,000. Green Platform claims this is easily justified by increased performance, reduced power usage, and fewer HDDs needed to meet performance goals.

IV. HPC ARCHITECTURES USING NVM

We envision that the initial use of NVM in systems will be evolutionary in nature. Over time, application developers will realize that their applications will be able to accomplish much more if they change their I/O paradigm. At that point the applications will make the

revolutionary changes to enable them to fully benefit from NVM capabilities.

With the initial introduction of NVM into the stack, the I/O model will not change. So, the injection of NVM into the architecture will have to support the current way applications do I/O to a persistent storage system. As with all technologies, NVM will be expensive before wide adoption. Thus, before the TCO (total cost of ownership) in dollars per gigabyte becomes viable, we'll:

a) Dump quickly to local or global NVM. While computing is going on, we'll trickle that information off to a larger, slower persistent storage system [7], [8].
b) Use NVM as a component of a more affordable persistent storage system where a small amount of NVM can increase the performance; e.g. as the metadata server for a PFS or a special-purpose database for primarily random read transactions.

After NVM TCO becomes viable, we'll convert the entire storage system to use NVM because it will be as dense as current storage systems, much faster, and use less energy.

The time will come that applications will change their I/O model. At that time we will need a breakthrough in the high-speed interface used to connect NVM to the processing logic of the system so that a global NVM can be used for all computing resources in a data center. Some data centers currently use global parallel file systems, for example PaScalBB (Parallel Scalable Backbone) file system at LANL (Los Alamos National Laboratory) and the Spider file system at ORNL (Oak Ridge National Laboratory).

a) Before a high-speed interface to a global NVM system exists, we'll store data structures in the NVM on which we'll compute. If we want to use a different system to do more computing, we'll do a memcpy from the current system's NVM to the new compute system's NVM.
b) After a high speed interface to a global NVM exists, we'll store data structures in the global NVM. A compute system will be pointed to that global NVM location and compute directly on that data. memcpy will no longer be required to move the computation from one system to another. The global NVM will have semantics about ownership and permissions for every piece of data it stores.

Under this scenario, imagine how the application that does defensive I/O will work. In order to quickly dump the state of an application as a defense against system component failure or to dump a snapshot of the state to use for examining progress over time, an application could use NVM as a storage level that is large enough to hold the dump and can be emptied to less expensive (in a TCO sense) storage during the next compute cycle. Ultimately, it wouldn't have to do defensive I/O at all because its state would always be preserved in its NVM-resident data structures. It could use memcpy to save snapshots of its state so that application progress can be examined over time at a later date.

Recall that where the series of applications transform data from initial input to final output, the intermediate transformed data may or may not be retained. In the former case, NVM could be used as the storage location for the intermediate databases. In the latter case, NVM

would initially be used in place of an HDD/SSD-based storage system for its better performance characteristics. Later NVM would be the location where the applications leave their memory-resident data structures intact and let each application use them in turn until the process is done.

For DISC applications, NVM may be used in place of memory-resident or large specialized HDD-based lookup databases. NVM might also be used as a place to quickly dump results that are later post-processed into a database, or for the database itself.

As already noted, in each of these cases where a database is required the NVM can be used as that database's storage location.

V. CONCLUSION

Current, emerging, and future NVM (non-volatile memory) technologies give us hope that we will be able to architect HPC systems that initially use these technologies as strategic components of the memory and storage hierarchy, and eventually migrate to be the memory and storage for the system. In the end, there will be no more wasting time transforming data between its in-memory structure and its file-based structure, and no more waiting for relatively slow persistent storage devices to record or provide this transformed data. Applications will use the NVM as they do DRAM-resident data today and rely on the same NVM to provide persistence with ownership and protections as does a HDD-based file system today.

ACKNOWLEDGMENT

The authors have had many fruitful discussions with colleagues. While any errors or omissions are our own, we wish to acknowledge Greg Atwood (Numonyx), John Bent (LANL), Adrian Caulfield (UCSD), Joel Coburn (UCSD), Nathan DeBardeleben (LANL), Blake Fitch (IBM), Gary Grider (LANL), Laura Grupp (UCSD), Rajesh Gupta (UCSD), Narasimha Reddy (TAMU), Sam Siewert (Atrato), Cliff Smith (Numonyx), Steve Swanson (UCSD), and Steve Wechgelaer (NRO).

REFERENCES

[1] J. Daly, "Methodology and metrics for quantifying application throughput", Proceedings of the Nuclear Explosives Code Developers Conference, 2006.

[2] X. Dong, N. Muralimanohar, N. Jouppi, R. Kaufmann, Y. Xie, "Leveraging 3D PCRAM technologies to reduce checkpoint overhead for future exascale systems", SC09, November 2009.

[3] http://en.wikipedia.org/wiki/Terabyte, Germany's Climate Research Centre (DKRZ), 2010.

[4] A. Woodard, "The phases they are a-changin'", http://www.semiconductorblog.com/2010/05/14/the-phases-they-are-a-changin/, May 14, 2010.

[5] P. Schmid and A. Roos, "'Niagara', the short stroking took", http://www.tomshardware.com/reviews/short-stroking-hdd,2157-4.html, Mar 5, 2009.

[6] M. Feldman, "Startup takes aim at performance-killing vibration in datacenter", http://www.hpcwire.com/topic/storage/Startup-Takes-Aim-at-Performance-Killing-Vibration-in-Datacenter-82102367.html?page=2, January 19, 2010.

[7] G. Grider, HECFSIO Workshop 2010, http://institutes.lanl.gov/hec-fsio/conferences/2010/presentations/day1/Grider-HECFSIO-2010-ExascaleEconomics.pdf, Aug 2, 2010.

[8] B. Kettering, "Role of NVM for applications", NVM Workshop at UCSD, April 13, 2010.