Using Active NVRAM for I/O Staging

Sudarsun Kannan, Dejan Milojicic, Ada Gavrilovska, Karsten Schwan

HP Laboratories
HPL-2011-131

Keyword(s):

NVRAM; Cloud; In-situ processing

Abstract:

A well-known problem for large scale cloud applications is how to scale their I/O performance. While next generation storage class memories like phase change memory and Memristors offer potential for high I/O bandwidths, if left unchecked, the raw volumes and rates of I/O already present in cloud can quickly overwhelm future I/O infrastructures. This fact is motivating research on 'data staging' in which I/O and data movement are partly replaced with and/or supplemented by computations that process data before moving it across I/O channels – in situ – to filter or reduce it, to better organize it for subsequent access (e.g., by other applications as in coupled codes), or to analyze it to quickly to derive important insights about the application producing those large data volumes. This paper proposes a technique that uses and exploits 'Active NVRAM' (non volatile memory) for staging I/O. Active NVRAMs are node-local NVRAMs that are embedded with a low power system-on-chip compute element. These active compute elements can be used to operate on output data asynchronously with the tasks performed by computational node elements, to reduce data or to perform some of the data processing required for data analytics before data is moved to longer term storage. The paper further describes the Active NVRAM design, sample ways in which it is used for I/O acceleration, and initial performance results evaluating the opportunities for and limitations of the Active NVRAM approach.

External Posting Date: August 21, 2011 [Fulltext] Approved for External Publication
Internal Posting Date: August 21, 2011 [Fulltext]

Using Active NVRAM for I/O Staging

Sudarsun Kannan, Dejan Milojicic
HP Labs
Palo Alto, CA, USA
sudarsun.kannan, dejan.milojicic@hp.com

Ada Gavrilovska, Karsten Schwan
Georgia Institute of Technology
Atlanta, GA, USA
ada, schwan@cc.gatech.edu

***Abstract*—A well-known problem for large scale cloud applications is how to scale their I/O performance. While next generation storage class memories like phase change memory and Memristors offer potential for high I/O bandwidths, if left unchecked, the raw volumes and rates of I/O already present in cloud can quickly overwhelm future I/O infrastructures. This fact is motivating research on 'data staging' in which I/O and data movement are partly replaced with and/or supplemented by computations that process data before moving it across I/O channels – in situ – to filter or reduce it, to better organize it for subsequent access (e.g., by other applications as in coupled codes), or to analyze it to quickly to derive important insights about the application producing those large data volumes. This paper proposes a technique that uses and exploits 'Active NVRAM' (non volatile memory) for staging I/O. Active NVRAMs are node-local NVRAMs that are embedded with a low power system-on-chip compute element. These active compute elements can be used to operate on output data asynchronously with the tasks performed by computational node elements, to reduce data or to perform some of the data processing required for data analytics before data is moved to longer term storage. The paper further describes the Active NVRAM design, sample ways in which it is used for I/O acceleration, and initial performance results evaluating the opportunities for and limitations of the Active NVRAM approach.**

***Keywords*-NVRAM, Cloud, In-situ processing**

I. Introduction

It is a well known fact that data size is growing faster than Moore's law. The growth has been 56 times over seven years. Not just enterprise scale applications but HPC applications have started considering cloud as an suitable cost effective infrastructure. With more services moving towards cloud based datacenters for scaling, the data explosion trend is predicted to continue. While processing power is important for applications to scale, equally important is an efficient method to store generated data (I/O), to operate on these data and absorb useful data and decimate the rest.

While most of the recent research have focused to improve the data storage efficiency by improving the I/O bandwidth using hardware and software techniques such as using new storage class memories (e.g. NVRAM), very few research efforts have focused on efficient mechanisms for I/O data processing (post processing). The I/O data processing which we refer here are data operations which can be either done during computation or offline in a local or remote node. While post processing is common in HPC systems, it is becoming equally important in Cloud too with increasing data. For example data analytics and monitoring are common operations in data center environment, for example using dedicated Hadoop cluster for analytics and monitoring [1]. Today services in cloud generate petabytes of data which are either stored to a shared persistent storage disks or moved over network for analytics processing. Either fetching data again from shared disks or over the network to perform post processing is not efficient and such data movement cause considerable data traffic. C.Wang et al. [2] argue that one way to attain scalability in analytics and monitoring is to immediately analyze data locally and filter monitoring information to reduce total data volumes passed across monitoring topologies or to disk. D. Logothetis et al. [1] throw a light on in-efficiency in moving terabytes of data for log processing to centralized locations and stress on the need for fast In-situ post processing. Post processing serve two purposes.

1) Post processing such as data reduction reduces the magnitude of data moved across I/O channel and also makes the data management easier. This directly benefits the I/O performance.
2) Post processing methods such as sorting, indexing, layout reorganization is important from the point of data analytics which require data assembled in a specific layout.

In this paper we specifically focus on improving performance of post processing for large scale applications in cloud. We propose a novel mechanism to use Active NVRAM for I/O staging to improve I/O throughput and more importantly post processing performance. Active NVRAM's are non-volatile memory attached with a low power compute element (Figure 1). Active NVRAM [3] is based on system-on-a-chip architecture providing high memory bandwidth to the compute elements. Each active NVRAM has its own operating system. NVRAM technologies such as Memristor and PCM offer 100X faster read-write performance and have endurance of approximately a million writes. They are scalable in terms of storage density with ability to store multiple bits per cell and require low power(also no refresh required) when compared to DRAM

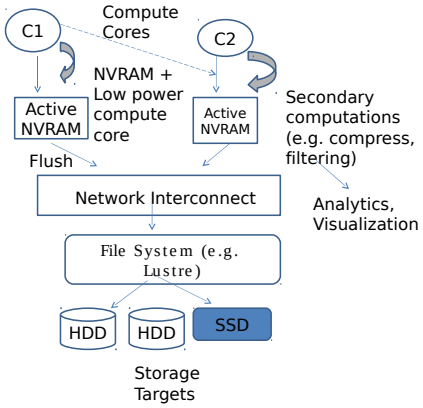

Figure 1. Architecture

which makes them a suitable replacement for disks. Our preliminary evaluations show high I/O performance gain using active NVRAM and we also analyze the effectiveness of Active NVRAM approach for post processing using HPC benchmarks. We argue that, by using Active NVRAM significant data post processing operations can be accomplished efficiently.

Technical contributions of the paper include

1) a framework that provides applications to use NVRAM for high performance I/O. The framework also supports asynchronous I/O data post processing using active NVRAM with minimal changes required to application
2) results of a common I/O post processing application-distributed sorting and the performance benefits of using Active NVRAM
3) Overheads of Active NVRAM based staging approach

II. Related Work

C.Wang et al. [2] motivate the need for processing data closer to computation in cloud. D. Logothetis et al. [1] discuss the time sensitivity aspects of log data post processing in cloud and the need for an efficient in-situ processing. However such in-situ processing on compute core do add overheads to application runtime which we aim to address using Active NVRAM. Moneta [4] provides an architectural overview of using distributed non volatile memories for I/O storage and provides insight on software optimization required for using non volatile memories such as SSD for distributed high performance storage. Our work relies on node local data storage and uses NVRAM like Memristor and PCM to improve I/O and also differs from others by not only improving I/O throughput using NVRAM but also improve post processing performance of I/O data. F.Zheng et al. [5] use dedicated I/O staging nodes specifically for post processing by moving data through high bandwidth interconnect for HPC systems. In cloud environment with restricted memory and network bandwidth for commodity VM's and with increasing I/O data, time taken to move data from compute core to staging nodes can add application overhead which we plan to investigate in future.

III. Post processing examples

While current approaches perform 'on-compute-core' processing or in dedicated I/O nodes, we propose using Active NVRAM. We implemented a simple data reduction technique by compression of I/O data and distributed merge sort as driving examples. While compression reduces the data and improves I/O performance, sorting is a commonly used analytics application [6], for example in map reduce based analytics to sort input files, and in monitoring too. We believe these two examples provide a good representation of computation intensive (compression) and communication intensive (sorting) post processing operations. The experimental evaluation analyzes the performance tradeoffs of using Active NVRAM for post processing.

IV. Other Active NVRAM applications

Though in this paper we specifically use active NVRAM for data post processing as an example, we believe our mechanism can have huge impact on other large scale applications too where I/O operations can be offloaded to active NVRAM. For example, current database applications have been divided into two data components. One component is persistent disk based database files stored in set of nodes, the other component is an in-memory key value store using dedicated nodes loaded with database values. To address power and cost issues of in-memory, FAWN [7] proposes a system architecture for using wimpy nodes (low power cores) with flash based storage devices. We believe node active NVRAM can be added as additional component(wimpy node) to physical machines for key-value store operations with database engine using powerful cores. This has potential not only in terms of performance but power benefits too. H.Amur et al. [8] propose a mechanism for alternating between high performance nodes and wimpy nodes based on data workloads for power benefits. We plan to investigate on these applications in our future work.

V. Importance of Active NVRAM in cloud

Poor I/O bandwidth for large scale applications in virtualized cloud environment is a well know problem. Inspite of different I/O optimizations, a major cause of poor I/O bandwidth performance is because of virtualization technology. Sharing physical I/O channels such as Gigabit Ethernet between virtual machines [9] results in poor bandwidth of around 87 MB for large scale instances in EC2. For large scale applications, the inter VM contention for I/O bandwidth adds substantial overheads. We believe, using a node local Active NVRAM would provide significant performance benefit in cloud environment. In this paper we do not focus on I/O virtualization issues. For evaluation we

use a non virtualized cluster at HP labs with NFS file system. In our future work we would like to evaluate our approach in a virtualized environment [10].

VI. Design

For applications to use NVRAM's for I/O, we propose an Active NVRAM framework. Active NVRAM framework (AMF) provides application with simple memory based interfaces for accessing NVRAM. Generally large scale applications have multiple processes or threads in each node and each process writes I/O data in chunks to active NVRAM. A chunk can be defined as sequential stream of data. Each chunk is a structure composed of metadata and data. The metadata contains information about physical address of chunk in active NVRAM, length of chunk, data type and a set of flags describing state of chunk. The metadata also contains information about set of operations that needs to perform as a part of post processing on each chunk.

A. NVRAM Allocation and commit

To write data to active NVRAM from a source data structure (e.g. variable, arrays, structure) applications make *nvmalloc()* call. The call results in allocation of memory in DRAM and also equivalent allocation of memory in active NVRAM. All updates to the allocated memory before commit operations are buffered in DRAM. This avoids the write latency to non volatile memory and indirectly addresses the wear leveling issues of NVRAM. Active NVRAM is persistent and applications know when data needs to be committed and saved. Hence it is responsibility of applications to explicitly commit *nvcommit()* data to active NVRAM. A commit guarantees an atomic failsafe update of data to active NVRAM. Since currently NVRAM's such as Memristor, Phase Change Memories are not available, the AMF uses memory mapped files to emulate persistence. AMF is flexible to work with any memory mapping compatible hardware.

B. Active NVRAM

Active NVRAM framework aims to decouple the general compute cores and the active NVRAM element. The compute cores and active element are independent and do not communicate. They behave in a producer consumer model. The compute cores execute applications, produce data in form of chunks, write them to their respective process compartment in NVRAM and continue with next iteration step. A compartment is similar to process address space. The compartment avoids memory conflicts between processes. The chunk metadata describes how the data needs to treated by the active element and what operations should be applied over the data chunk. To handle multi-threaded environments, AMF uses a global active NVRAM work queue. All processes that commit data to NVRAM, update their chunk metadata to the global work queue after the data

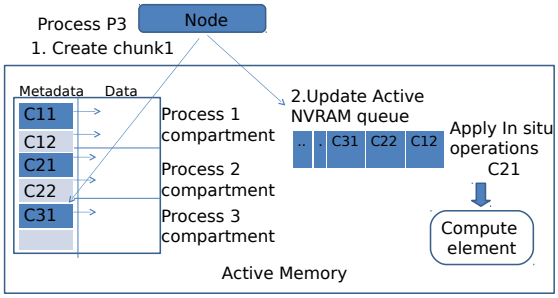

Figure 2. Design

is committed to NVRAM. The active element de-queues entry one by one applies processing to chunk.

VII. Experimental Evaluation

To evaluate our approach of using Active NVRAM for I/O staging, we used infiniband-based cluster at HP. The cluster nodes run non-virtualized Linux OS. Each node has Quad core Intel Xeon processors with node local hard disks for storage. To emulate active NVRAM, each node uses three cores for running actual simulation and one core is dedicated for active NVRAM compute element. The home directories of applications are mounted over NFS. Experimental results

1) validates the benefits of using NVRAM for I/O in terms of throughput
2) provides a comparison of performance tradeoffs of using Active NVRAM with respect to compute core in-situ processing
3) analyzes application overheads due to software and hardware limitations in using Active NVRAM

A. I/O throughput

For evaluation we primarily use NAS-NPB benchmarks [11] NAS-NPB benchmarks provide a good mix of applications with different characteristics such as communication intensive, I/O intensive and embarrassingly parallel applications. We primarily use the I/O intensive BTIO benchmark for most of our evaluation. Figure 3 compares I/O throughput of applications using disk storage in NFS mode, Ramdisk and Active NVRAM. We use the application check pointing as source of I/O data. Ramdisks are memory mounted file system.

1) Observations: As expected, using NFS results in very poor throughput. Ramdisk and NVRAM provide huge throughput gains compared to disk. For the largest data size in out experiment, we observed around 385% gain over disk as expected. An interesting thing to note is the performance gains with NVRAM approach when compared to using ramdisk. Though ramdisk creates a file system on system memory, applications use the file system semantics such as seek(), read(), write() in block sizes which results in additional overheads when compared to NVRAM. With increasing computation, the I/O data size also increases.

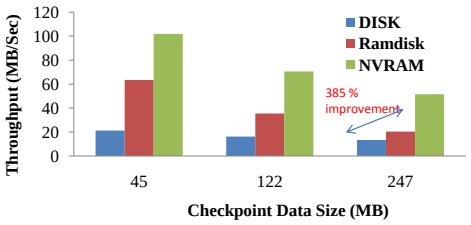

Figure 3. Throuput comparison - Disk, Ramdisk, NVRAM

The throughput of disk based I/O further decreases because of increased NFS contention. In case of active NVRAM, we use DRAM to emulate NVRAM by memory mapping file. With increasing data size, memory used by application and active NVRAM increases, resulting in increase disk swapping which is why the effective througput decreases. We believe such overheads should be absent when using real hardware.

B. Performance impact in using NVRAM

The next experiment evaluates implication of using NVRAM for data post processing with distributed I/O data sorting as an example using the I/O intensive BTIO benchmark for experiments. Figure 4 compares application runtime for on-compute core post processing indicated by green points with active NVRAM approach indicated by red points. The experiments relate application performance with varying the number of cores (x-axis) and the data size (y-axis).

1) Observations: When the *data size(MB) to no. of compute cores* ratio is small; application performance is almost similar for both approaches. But as the ratio increases, there is a considerable performance gain in case of active NVRAM approach. The reason for this behavior can be attributed towards communication intensive nature of sorting. For small data sizes, post processing after each checkpoint step does not add much overhead to application run time. But with increasing data size, even though 'on compute core' approach uses powerful compute cores, communication latencies adds substantial impact to application runtime whereas asynchronous processing approach in Active NVRAM hides such overheads.

The Figure 6 provides insights on using a computation intensive compression for post processing without any communication. The algorithm is a loosy compression and provides a 4:1 data reduction. The graph indicates almost no performance difference when using the 'on-compute core' for post processing and compared to active NVRAM for different *data size to no. of compute cores* ratio.

2) Observations: The reason for this behavior can be attributed as follows

1) In case of on compute core approach, though compression is performed synchronously with computation, the ratio of compute cores to active element is 3:1 which explains why this approach adds less than 2 percent overhead to application run time.
2) The time spent on compression is around 4 percent of application run time and so there is not much significant gain in doing asynchronously in active NVRAM. In our future work we plan to use different compression techniques to understand the performance tradeoffs.

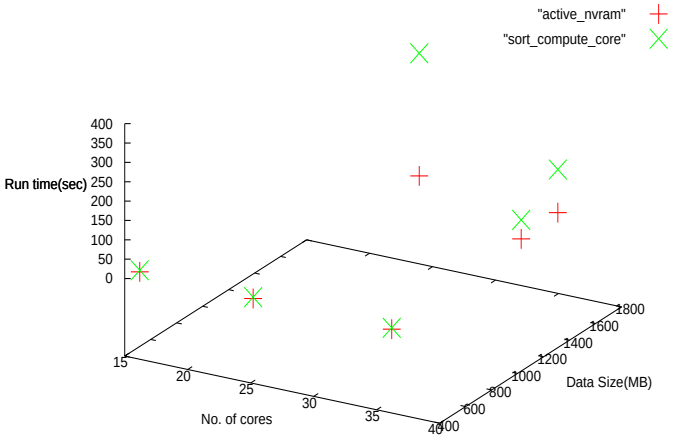

Figure 4. Post processing- Sorting

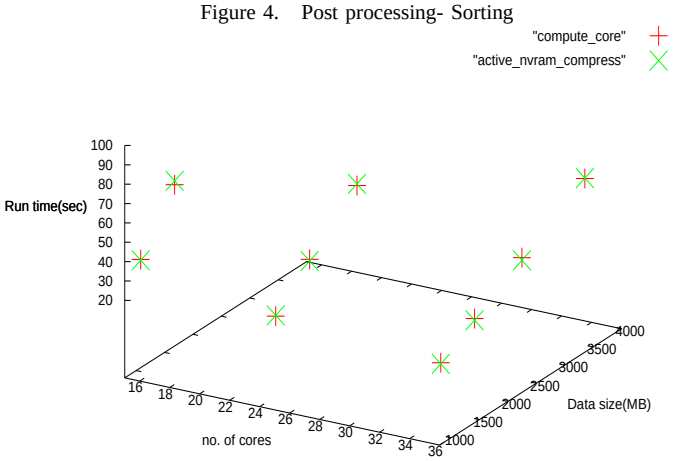

Figure 5. Post processing- Compression

C. Active NVRAM overheads

The compute cores and compute element in active NVRAM share the communication channel, and NVRAM. The next experiment Figure 6 analyzes the overheads of post-processing using active NVRAM due to resource sharing. We use different applications in NPB benchmark by modifying the benchmark to write I/O using our NVRAM framework. We use distributed sort as it as good example of communication intensive in situ processing.

1) Observations: In case of communication intensive benchmarks such as MG, CG, if active NVRAM processing is also communication intensive, application experiences

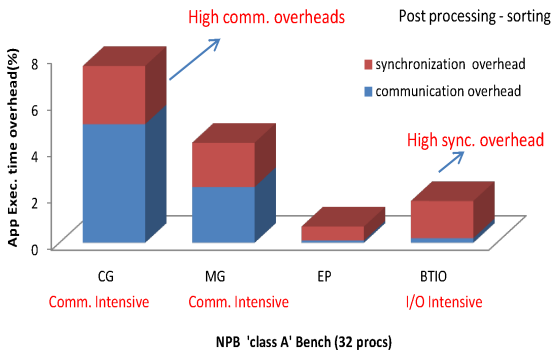

Figure 6. Active NVRAM - Overheads

around 8% overhead. This is because of communication channel contention between application processes and Active NVRAM process. For embracingly parallel application benchmarks (EP) with minimum communication and small I/O size, there is less communication overhead. Compute cores and active NVRAM function in producer-consumer model and hence during global queue updates, the queue requires locking. Synchronization overhead is slightly higher in case of I/O sensitive BTIO benchmark. Our future work would explore methods to reduce such synchronization overheads and communication overheads by efficient scheduling.

VIII. Conclusions and Future Work

In this paper we analyze the opportunities of using Active NVRAM for large scale data (I/O) intensive applications and use data post processing as a driving example. We use high performance application benchmarks first to evaluate the I/O performance gains and use the same benchmark to analyze the tradeoffs of post processing using Active NVRAM. Our preliminary results show over 385% I/O throughput benefits. While for the sort based post processing our mechanism shows 160% improvement in, but for compute intensive operations we do not see much benefit. Our preliminary results are evaluation based on limited scale and benchmark driven. In our next phase of research, we would like to improve the experiment scale and evaluate performance for different data post processing examples. Also we would like to address some of the performance overheads due to In-situ processing.

References

[1] D. Logothetis, C. Trezzo, K. C. Webb, and K. Yocum, "In-situ mapreduce for log processing," in *Proceedings of the 2011 USENIX conference on USENIX annual technical conference*, ser. USENIXATC'11. Berkeley, CA, USA: USENIX Association, 2011, pp. 9–9.

[2] C. Wang, K. Schwan, V. Talwar, G. Eisenhauer, L. Hu, and M. Wolf, "A flexible architecture integrating monitoring and analytics for managing large-scale data centers," in *Proceedings of the 8th ACM international conference on Autonomic computing*, ser. ICAC '11. New York, NY, USA: ACM, 2011, pp. 141–150.

[3] P. Ranganathan, "From microprocessors to nanostores: Rethinking data-centric systems," *Computer*, vol. 44, pp. 39–48, January 2011.

[4] A. M. Caulfield, A. De, J. Coburn, T. I. Mollow, R. K. Gupta, and S. 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*, ser. MICRO '43. Washington, DC, USA: IEEE Computer Society, 2010, pp. 385–395.

[5] F. Zheng, H. Abbasi, C. Docan, J. Lofstead, S. Klasky, Q. Liu, M. Parashar, N. Podhorszki, K. Schwan, and M. Wolf, "PreDatA - Preparatory Data Analytics on Peta-Scale Machines," in *Proceedings of the 24 th IEEE International Parallel and Distributed Processing Symposium*, Atlanta, GA, USA, Apr. 2010.

[6] R. Ananthanarayanan, K. Gupta, P. Pandey, H. Pucha, P. Sarkar, M. Shah, and R. Tewari, "Cloud analytics: do we really need to reinvent the storage stack?" in *Proceedings of the 2009 conference on Hot topics in cloud computing*, ser. HotCloud'09. Berkeley, CA, USA: USENIX Association, 2009.

[7] D. G. Andersen, J. Franklin, M. Kaminsky, A. Phanishayee, L. Tan, and V. Vasudevan, "Fawn: a fast array of wimpy nodes," in *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, ser. SOSP '09. New York, NY, USA: ACM, 2009, pp. 1–14.

[8] H. Amur, J. Cipar, V. Gupta, G. R. Ganger, M. A. Kozuch, and K. Schwan, "Robust and flexible power-proportional storage," in *Proceedings of the 1st ACM symposium on Cloud computing*, ser. SoCC '10. New York, NY, USA: ACM, 2010, pp. 217–228.

[9] S. Ostermann, A. Iosup, N. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema, "A Performance Analysis of EC2 Cloud Computing Services for Scientific Computing," in *Cloud Computing*, ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, Berlin, Heidelberg, 2010, vol. 34, pp. 115–131.

[10] R. Campbell, I. Gupta, M. Heath, S. Y. Ko, M. Kozuch, M. Kunze, T. Kwan, K. Lai, H. Y. Lee, M. Lyons, D. Milojicic, D. O'Hallaron, and Y. C. Soh, "Open cirrustmcloud computing testbed: federated data centers for open source systems and services research," in *HotCloud*, 2009.

[11] "NAS Parallel Benchmarks," http://www.nas.nasa.gov/Resources/Software/npb.html.