Measuring I/O Behavior on Upcoming Systems with NVRAM

Christian Herold
Technische Universität Dresden
christian.herold@tu-dresden.de

Ronny Tschüter
Technische Universität Dresden
ronny.tschueter@tu-dresden.de

ABSTRACT

Upcoming HPC systems will use NVRAM to address existing I/O bottlenecks. The I/O performance thereby is one of the keys for the exascale challenges. NVRAM introduces a new level in the memory hierarchy and can be utilized by different technologies e.g. memory mapped files or block transfer operations. Using NVRAM without considering the complete hierarchy may lead to an inefficient usage and bad I/O performance. Therefore in this work, we evaluate techniques for measuring the I/O behavior of applications that utilize NVRAM in various use cases. We extended the application tracing of our tool Score-P [2] and introduced several metrics as well as events for different use cases of NVRAM.

KEYWORDS

performance analysis, I/O tracing, nvram, persistent memory

ACM Reference format:
Christian Herold and Ronny Tschüter. 2017. Measuring I/O Behavior on Upcoming Systems with NVRAM. In *Proceedings of The International Conference for High Performance Computing, Networking, Storage, and Analysis - Regular Poster, Denver, Colorado USA, November, 2017 (SC2017),* 2 pages.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

I/O performance is highly dependent on how activities utilize different components. For instance, data movements are costly events on existing I/O hierarchies with SSDs, spinning devices and tapes. Exascale systems are expected to have an NVRAM layer in order to mitigate I/O related issues. This layer is expected to reside between the DRAM and the traditional I/O system hierarchy, see figure 1. Introducing NVRAM as an new memory level offers various use cases. The most significant usage models are:

- *Application direct usage model:* NVRAM is mapped into the address space and can be accessed by CPU load/store instructions.
- *Storage usage model:* NVRAM is available as persistent block device like a very fast SSD.
- *Memory usage model:* NVRAM is used as an extension of the main memory and it behaves like volatile DRAM.

NVRAM supports various application types and offers lower access latencies than current mass storage technologies. Nevertheless, it introduces a new layer in the memory hierarchy which the developer needs to consider when designing and tuning an application. The various usage models require diverse experiences in order to efficiently utilize NVRAM. At this point, performance analysis tools are needed to support the application development by measuring the I/O behavior for all usage models.

For this purpose, we are extending the tracing tool Score-P to provide I/O measurements for each use case. Sections 3-5 explain how we support each usage model.

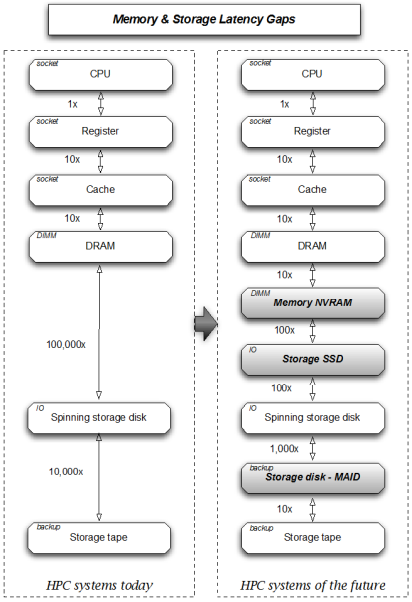

Figure 1: Memory hierarchy transition of HPC systems

2 RELATED WORK

State of the art I/O monitoring tools mostly focus on the *storage usage model* and record the behavior of block transfer operations.

Darshan [1] can be used to measure and characterize the I/O behavior of a HPC application. Therefore, it intercepts operation calls to the I/O libraries and extracts I/O statistics and timing information. At the end of the execution, Darshan compresses and stores the I/O behavior in log files. Darshan supports only the *storage usage model* by recording I/O statistics and it does not provide an in-depth I/O analysis.

The SIOX [3] framework also provides I/O tracing and, based on that, tools for analyzing, reporting and optimizing the I/O behavior of an HPC application. Similar to Darshan, SIOX only supports the *storage usage model.*

VampirTrace [4] also provides I/O tracing but reached the end of its life. Score-P [2] is its successor and currently does not support

SC2017, November, 2017, Denver, Colorado USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

I/O tracing. However, this work contributes to the on going efforts to support the *storage usage model*.

3 THE APPLICATION-DIRECT USAGE MODEL

The Non-Volatile Memory Library (NVML) [5] is currently the most common framework to directly access the NVRAM. We have implemented library wrappers for the NVML in our tracing tool Score-P. As a result, each function call to the NVML is intercepted by these wrappers in order to generate events like entering a function and/or record metrics, e.g. memory consumption. The resulting trace can be analyzed in Vampir. The NVML consists of several libraries. We currently support:

- **libpmem** - a low level interface to access the persistent memory and support the high level libraries of the NVML
- **libpmemobj** - provides a transactional object store to allocate objects and transfer them to persistent memory
- **libpmemblk** - responsible for managing a persistent array of blocks using atomic updates

For these libraries, we generate events for entering and leaving API functions in order to measure the duration of a call. We record the memory consumption of NVRAM by wrapping allocation and de-allocation operations of libpmem. This metric is updated during the application runtime. It helps the user to find time intervals with significant memory consumption.

Monitoring **libpmemobj** provides similar metrics as libpmem but with focus on objects. Thereby, we are going to provide the following metrics for the object store:

- Number of allocated objects per pool
- Consumption per memory pool

As a memory pool is allocated with a number of objects, the usage of this memory pool increases and decreases as the application execution proceeds over time. We will also provide a High-Water-Mark metric for the utilization of the memory pool over its entire lifetime. With this metric, the user will be able to identify peak times where a significant amount of objects in the pool have been consumed.

The **libpmemblk** library is responsible for managing a persistent array of blocks where all elements in a block are of the same size. Furthermore, the interface provides operations to atomically write/read blocks of an array. An uncareful usage can severely impact application performance. For this purpose, we create events for corresponding data transfers. The following information is stored in the events:

- Type of operation, e.g. read or write
- Transfer size
- Begin and end timestamp

This information can assist in finding the bottlenecks that might have occurred during data transfer operations from/to NVRAM. Additionally, the I/O bandwidth can be determined by using the timestamps and transfer sizes.

4 THE STORAGE USAGE MODEL

In this model, NVRAM is used as a block device like a SSD and data is transferred by file I/O operations, e.g. `pread` (Posix). Tracing file I/O accesses is an upcoming feature of Score-P we are currently working on. Hence, we use a development version of Score-P to analyze the I/O behavior. We also contribute in this development to provide additional information about storage for upcoming systems with NVRAM. In order to distinguish between different types of devices and filesystems, we add mount and device information to the trace for each file which was opened/created.

5 THE MEMORY USAGE MODEL

In this use case, NVRAM is volatile like DRAM and both memory types can be accessed by addresses and without any indirection. The access latencies are different between these memory types where NVRAM is slower than DRAM by a certain factor. As a consequence, it is important to know where memory accesses have been performed in the application and which memory type was accessed. This will help to find hotspots for using DRAM and NVRAM.

Supporting I/O analysis for the *memory usage model* is one of our future tasks. We are going to provide memory statistics for DRAM and NVRAM accesses. Furthermore, we will also distinguish between remote and local memory for multi-socket systems. These statistics will guide the user to improve the data locality and access patterns of the application.

6 CONCLUSION

This work enables application developers to measure the I/O behavior of programs that utilize NVRAM. We extended the Score-P measurement system to record I/O activities for all use cases except the *memory usage model*. Supporting the I/O recording for the *memory usage model* is a future task within the NEXTGenIO project. New metrics and events were developed in order to depict the I/O access behavior of NVRAM as accurately as possible. All these features will be released in a future version of Score-P and can be used to understand and optimize the NVRAM usage of HPC applications.

7 ACKNOWLEDGEMENT

This research was undertaken as part of the NEXTGenIO project, which is funded through the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement no. 671951.

REFERENCES

[1] Philip Carns, Kevin Harms, William Allcock, Charles Bacon, Samuel Lang, Robert Latham, and Robert Ross. 2011. Understanding and Improving Computational Science Storage Access Through Continuous Characterization. *Trans. Storage* 7, 3, Article 8 (Oct. 2011), 26 pages. https://doi.org/10.1145/2027066.2027068

[2] Andreas Knüpfer, Christian Rössel, Dieter an Mey, Scott Biersdorff, Kai Diethelm, Dominic Eschweiler, Markus Geimer, Michael Gerndt, Daniel Lorenz, Allen Malony, et al. 2012. Score-P: A joint performance measurement run-time infrastructure for Periscope, Scalasca, TAU, and Vampir. In *Tools for High Performance Computing 2011*. Springer, 79–91.

[3] Julian M. Kunkel, Michaela Zimmer, Nathanael Hübbe, Alvaro Aguilera, Holger Mickler, Xuan Wang, Andriy Chut, Thomas Bönisch, Jakob Lüttgau, Roman Michel, and Johann Weging. 2014. *The SIOX Architecture – Coupling Automatic Monitoring and Optimization of Parallel I/O.* Springer International Publishing, Cham, 245–260. https://doi.org/10.1007/978-3-319-07518-1_16

[4] Matthias S Müller, Andreas Knüpfer, Matthias Jurenz, Matthias Lieber, Holger Brunst, Hartmut Mix, and Wolfgang E Nagel. 2007. Developing Scalable Applications with Vampir, VampirServer and VampirTrace.. In *PARCO*, Vol. 15. 637–644.

[5] nvml 2017. NVM Library – NVM Library for using memory-mapped persistence. (2017). Retrieved June, 2017 from http://pmem.io/nvml/