Distributed Shared Persistent Memory

Yizhou Shan
Purdue University
ys@purdue.edu

Shin-Yeh Tsai
Purdue University
tsai46@purdue.edu

Yiying Zhang
Purdue University
yiying@purdue.edu

ABSTRACT

Next-generation non-volatile memories (*NVMs*) will provide byte addressability, persistence, high density, and DRAM-like performance. They have the potential to benefit many datacenter applications. However, most previous research on NVMs has focused on using them in a single machine environment. It is still unclear how to best utilize them in distributed, datacenter environments.

We introduce *Distributed Shared Persistent Memory (DSPM)*, a new framework for using persistent memories in distributed datacenter environments. DSPM provides a new abstraction that allows applications to both perform traditional memory load and store instructions and to name, share, and persist their data.

We built *Hotpot*, a kernel-level DSPM system that provides low-latency, transparent memory accesses, data persistence, data reliability, and high availability. The key ideas of Hotpot are to integrate distributed memory caching and data replication techniques and to exploit application hints. We implemented Hotpot in the Linux kernel and demonstrated its benefits by building a distributed graph engine on Hotpot and porting a NoSQL database to Hotpot. Our evaluation shows that Hotpot outperforms a recent distributed shared memory system by 1.3× to 3.2× and a recent distributed PM-based file system by 1.5× to 3.0×.

1 INTRODUCTION

Next-generation non-volatile memories (*NVMs*), such as 3DX-point [4], phase change memory (*PCM*), spin-transfer torque magnetic memories (*STTMs*), and the memristor will provide byte addressability, persistence, high density, and DRAM-like performance. These developments are poised to radically alter the landscape of memory and storage technologies and have already inspired a host of research projects [1, 12]. However, most previous research on NVMs has focused on using them in a single machine environment. Even though NVMs have the potential to greatly improve the performance and reliability of large-scale applications, it is still unclear how to best utilize them in distributed, datacenter environments.

This project takes a significant step towards the goal of using NVMs in distributed datacenter environments. We propose *Distributed Shared Persistent Memory (DSPM)*, a framework that provides a global, shared, and persistent memory space using a pool of machines with NVMs attached at the main memory bus. Applications can perform native memory load and store instructions to access both local and remote data in this global memory space and can at the same time make their data persistent and reliable. DSPM can benefit both single-node persistent-data applications that want to scale out efficiently and shared-memory applications that want to add durability to their data.

Unlike traditional systems with separate memory and storage layers [2], we propose to use just one layer that incorporates both distributed memory and distributed storage in DSPM. DSPM's one-layer approach eliminates the performance overhead of data marshaling and unmarshaling, and the space overhead of storing data twice. With this one-layer approach, DSPM can potentially provide the low-latency performance, vast persistent memory space, data reliability, and high availability that many modern datacenter applications demand.

Building a DSPM system presents its unique challenges. Adding "Persistence" to Distributed Shared Memory (DSM) is not as simple as just making in-memory data durable. Apart from data durability, DSPM needs to provide two key features that DSM does not have: persistent naming and data reliability. In addition to accessing data in PM via native memory loads and stores, applications should be able to easily name, close, and re-open their in-memory data structures. User data should also be reliably stored in NVM and sustain various types of failures; they need to be consistent both within a node and across distributed nodes after crashes. To make it more challenging, DSPM has to deliver these guarantees without sacrificing application performance in order to preserve the low-latency performance of NVMs.

We built *Hotpot*, a DSPM system in the Linux kernel. Hotpot offers low-latency, direct memory access, data persistence, reliability, and high availability to datacenter applications. It exposes a global virtual memory address space to each user application and provides a new persistent naming mechanism that is both easy-to-use and efficient. Internally, Hotpot organizes and manages data in a flexible way and uses a set of adaptive resource management techniques to improve performance and scalability.

Hotpot builds on two main ideas to efficiently provide data reliability with distributed shared memory access. Our first idea is to integrate distributed memory caching and data replication by imposing *morphable* states on persistent memory (*PM*) pages.

In DSM, when applications on a node access shared data in remote memory *on demand*, DSM caches these data copies in its local memory for fast accesses and later evicts them when reclaiming local memory space. Like DSM, Hotpot caches application-accessed data in local PM and ensures the coherence of multiple cached copies across nodes. But Hotpot also uses these cached data as *persistent replicas* and ensures their reliability and crash consistency.

On the other hand, unlike distributed storage systems, which *creates* extra data replicas to meet user-specified reliability requirements, Hotpot makes use of data copies that *already exist* in the system when they were fetched to local due to application accesses.

In essence, every local copy of data serves two simultaneous purposes. First, applications can access it locally without any network delay. Second, by placing the fetched copy in PM, it can be treated as a persistent replica for data reliability.

This seemingly-straightforward integration is not simple. Maintaining wrong or outdated versions of data can result in inconsistent data. To make it worse, these inconsistent data will be persistent in

PM. We carefully designed a set of protocols to deliver data reliability and crash consistency guarantees while integrating memory caching and data replication.

Our second idea is to exploit application behaviors and intentions in the DSPM setting. Unlike traditional memory-based applications, persistent-data-based applications, DSPM's targeted type of application, have well-defined data *commit points* where they specify what data they want to make persistent. When a process in such an application makes data persistent, it usually implies that the data can be *visible* outside the process (*e.g.*, to other processes or other nodes). Hotpot utilizes these data commit points to also push updates to cached copies on distributed nodes to avoid maintaining coherence on every PM write. Doing so greatly improves the performance of Hotpot, while still ensuring correct memory sharing and data reliability.

2 MOTIVATION

DSPM is motivated by three datacenter trends: emerging hardware PM technologies, modern data-intensive applications' data sharing, persistence, and reliability needs, and the availability of fast datacenter network.

2.1 Persistent Memory and PM Apps

NVMs can attach directly to the main memory bus to form Persistent Memory, or PM. If applications want to exploit all the low latency and byte-addressability benefits of PM, they should directly access it via memory load and store instructions without any software overheads [8] (we call this model *durable in-memory computation*), rather than accessing it via a file system [1].

Unfortunately, most previous durable in-memory systems were designed for the single-node environment. With modern datacenter applications' computation scale, we have to be able to scale out these single-node PM systems.

2.2 Shared Memory Applications

Modern data-intensive applications increasingly need to access and share vast amounts of data fast. Distributed Shared Memory (*DSM*) takes the shared memory concept a step further by organizing a pool of machines into a globally shared memory space. Researchers and system builders have developed a host of software and hardware DSM systems in the past few decades [2, 7].

However, although DSM scales out shared-memory applications, there has been no persistent-memory support for DSM. DSM systems all had to checkpoint to disks [10]. Memory persistence can allow these applications to checkpoint fast and recover fast [6].

2.3 Fast Network and RDMA

Datacenter network performance has improved significantly over the past decades. InfiniBand (*IB*) NICs and switches support high bandwidth ranging from 40 to 100 Gbps. Remote Direct Memory Access (*RDMA*) technologies that provide low-latency remote memory accesses have become more mature for datacenter uses in recent years [11]. These network technology advances make remote-memory-based systems [7] more attractive than decades ago.

2.4 Lack of Distributed PM Support

Many large-scale datacenter applications require fast access to vast amounts of persistent data and could benefit from PM's performance, durability, and capacity benefits. For PMs to be successful in datacenter environments, they have to support these applications. However, neither traditional distributed storage systems or DSM systems are designed for PM. Traditional distributed storage systems [3] target slower, block-based storage devices. Using them on PMs will result in excessive software and network overheads that outstrip PM's low latency performance [12]. DSM systems were designed for fast, byte-addressable memory, but lack the support for data durability and reliability.

Octopus [5] is a recent RDMA-enabled distributed file system built for PM. Octopus and our previous work Mojim [12] are the only distributed PM-based systems that we are aware of. Octopus was developed in parallel with Hotpot and has a similar goal as Hotpot: to manage and expose distributed PM to datacenter applications. However, Octopus uses a file system abstraction and is built in the user level without any data reliability guarantee.

3 CONCLUSION

We introduce the DSPM model and use a one-layer approach to built Hotpot by integrating memory coherence and data replication. We believe DSPM will benefit datacenter applications, and we hope more researchers will look into DSPM to expolore more possibilities. The original paper was first published at SoCC'17 [9]. The paper is available at: wuklab.io/hotpot-socc17.pdf.

REFERENCES

[1] S. R. Dulloor, S. Kumar, A. Keshavamurthy, P. Lantz, D. Reddy, R. Sankaran, and J. Jackson. System software for persistent memory. In *Proceedings of the EuroSys Conference (EuroSys '14)*, Amsterdam, The Netherlands, April 2014.

[2] P. Ferreira and M. Shapiro. Larchant: Persistence by Reachability in Distributed Shared Memory through Garbage Collection. In *Distributed Computing Systems, 1996., Proceedings of the 16th International Conference on*, 1996.

[3] S. Ghemawat, H. Gobioff, and S.-T. Leung. The Google File System. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP '03)*, Bolton Landing, New York, October 2003.

[4] Intel Corporation. Intel Non-Volatile Memory 3D XPoint. http://intel.ly/2sq0w09.

[5] Y. Lu, J. Shu, Y. Chen, and T. Li. Octopus: an rdma-enabled distributed persistent memory file system. In *2017 USENIX Annual Technical Conference (USENIX ATC 17)*, Santa Clara, CA, 2017.

[6] D. Narayanan and O. Hodson. Whole-System Persistence. In *Proceedings of the 17th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '12)*, London, United Kingdom, March 2012.

[7] J. Nelson, B. Holt, B. Myers, P. Briggs, L. Ceze, S. Kahan, and M. Oskin. Latency-Tolerant Software Distributed Shared Memory. In *Proceedings of the 2015 USENIX Conference on Usenix Annual Technical Conference (ATC '15)*, Santa Clara, California, July 2015.

[8] S. Pelley, P. M. Chen, and T. F. Wenisch. Memory persistency. In *Proceeding of the 41st Annual International Symposium on Computer Architecuture*, ISCA '14, Piscataway, NJ, USA, 2014.

[9] Y. Shan, S.-Y. Tsai, and Y. Zhang. Distributed shared persistent memory. In *Proceedings of the 8th Annual Symposium on Cloud Computing (SOCC '17)*, Santa Clara, CA, USA, September 2017.

[10] M. Stumm and S. Zhou. Fault tolerant distributed shared memory algorithms. In *Proceedings of the Second IEEE Symposium on Parallel and Distributed Processing 1990*, Dec 1990.

[11] S.-Y. Tsai and Y. Zhang. Lite kernel rdma support for datacenter applications. In *Proceedings of the 26nd ACM Symposium on Operating Systems Principles (SOSP '17)*, Shanghai, China, 2017.

[12] Y. Zhang, J. Yang, A. Memaripour, and S. 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 (ASPLOS '15)*, Istanbul, Turkey, March 2015.