Assise: Performance and Availability via Client-local NVM in a Distributed File System

Thomas E. Anderson [1] Marco Canini [2] Jongyul Kim[3†] Dejan Kostić[4] Youngjin Kwon[3]
Simon Peter[5] Waleed Reda[4, 6⋆] Henry N. Schuh[1†] Emmett Witchel [5]

[1]University of Washington [2]KAUST [3]KAIST
[4]KTH Royal Institute of Technology [5]The University of Texas at Austin [6]Université catholique de Louvain

1. Introduction

Byte-addressable non-volatile memory is becoming commercially available. NVM provides near-DRAM performance at a fraction of the cost, driving the adoption of node-local NVM at scale [6]. The increasing availability of NVM upends the current model of client-server distributed file systems (e.g., Ceph [5]). Such systems commonly separate their storage servers from the client to simplify resource pooling. This separation comes at a performance cost, which becomes more pronounced as we move from HDD/SSD to NVM.

In the client-server model, file system clients maintain a volatile cache, managed by the OS, and the kernel needs to mediate every (meta-)data operation. On a cache-miss, multiple network round-trips are incurred to fetch data and metadata. These operations often take orders of magnitude longer than the sub-microsecond latencies offered by NVM. Secondly, client-local caches are managed at fixed page-block granularity, resulting in the amplification of small IO operations typical of many distributed applications. Upon a failure, applications need to reconstruct the client's volatile cache from scratch, necessitating long fail-over times to restore application-level service. These issues prevent existing distributed file systems from fully leveraging the performance, byte-addressability, and persistence capabilities of NVM.

Assise [1] is a POSIX-compatible distributed file system that addresses the problems of the client-server model and provides low tail-latency, scalability, high availability, and efficient bandwidth use. Assise unleashes the performance of NVM via pervasive and persistent caching in client-local NVM. In doing so, Assise avoids kernel, network, and IO amplification overheads resulting in orders of magnitudes better performance. Moreover, with its use of client-local non-volatile caches, Assise provides near-instantaneous recovery with strong consistency semantics and even faster fail-over to *hot replicas* that mirror an application's local client cache.

2. Assise Design

We describe the design principles of Assise. Figure 1 shows a simplified version of our IO architecture.

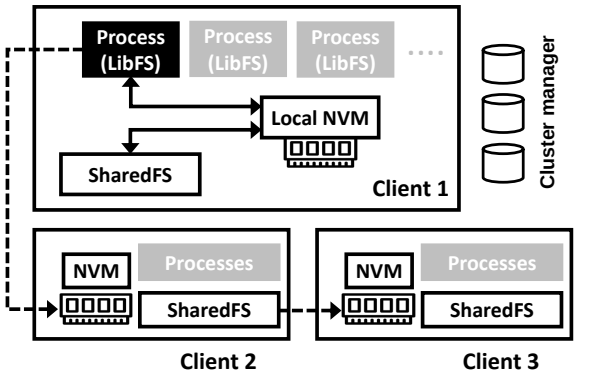

Figure 1. Assise's IO architecture with a 3-node replica group. Solid line = Loads & Stores. Dashed line = RDMA.

Client-local NVM caching. Assise moves file system data and metadata management to client-local NVM, managed by a client-local file system daemon called SharedFS. To bypass the kernel for process-local IO, Assise provides a file system library (LibFS). LibFS intercepts POSIX file system calls and executes them in userspace. LibFS writes data and metadata at byte granularity, as opposed to blocks, to an operation log in process-private NVM. In doing so, Assise provides direct userspace data access to client-local NVM, without block amplification, reducing IO latency. Logging is a natural way to provide prefix crash consistency [2].

Userspace remote IO with RDMA and hot replicas. Assise chain-replicates userspace operation logs to hot replicas via RDMA. We leverage the write order properties of RDMA to provide prefix crash consistency. Given that clients replicate hot data to other clients' NVM caches, applications on failing clients can be quickly restarted on any other hot client replica.

Linearizability via leases. Assise uses leases [3] to provide single-writer, multiple-reader access to files and directories. Leases allow process-local access to a file or directory without need for synchronization, until they expire or are revoked. Revocations can be triggered if other processes request the same lease. Leases provide low-overhead metadata management for workloads with temporal locality. To acquire a lease, processes first contact a *cluster manager* — a replicated service that arbitrates leases across SharedFS instances.

⋆ Lead student author.

† Co-student authors.

Scalability via lease delegation. To avoid the cluster manager becoming a bottleneck, we introduce a scalable lease delegation mechanism. This mechanism allows lease management responsibility to transfer from the cluster manager to client-local SharedFSes. This allows processes on the same client to hand-off leases to each other with only a context switch to their local SharedFS.

3. Evaluation

Assise is implemented in 28,982 lines of C code (LoC). Our experimental testbed consists of 5× dual-socket Intel Cascade Lake-SP servers running at 2.2 GHz, with a total of 48 cores (96 hyperthreads), 384 GB DDR4-2666 DRAM and equipped with 6 TB Intel Optane DC PMM per machine. The DIMM slots on each socket are fully populated. All nodes use a 40-GbE ConnectX-3 NIC and are connected via an InfiniBand switch. We compare Assise to Ceph [5]. For Assise, we assign a 3 GB LibFS cache and partition it to a 1-GB NVM process-local update log and a 2-GB DRAM read cache. Similarly, we limit Ceph's kernel buffer cache to 3 GB. For this evaluation, we focus primarily on scalability. Our OSDI paper [1] includes an in-depth evaluation of Assise's latency and availability — showing Assise's recovery performance for the most common crash types.

Parallel mail delivery. We evaluate scalability with Postfix, a multi-process mail delivery agent. Postfix can generate hundreds of processes & performs file creations and renames to deliver mail. We configure this benchmark to replay 70 GB worth of emails. Postfix uses a pool of mail delivery processes that pull mail from an incoming mail queue and deliver it to one or more mailboxes. We use the Maildir format, where each mailbox has its own directory, and one file is created per mail message delivered to a mailbox. For our workload, we use an email trace with an average mail size of 200 KB [4]. We setup postfix to perform mail delivery on 3 nodes, and use 1 machine to generate the workload. We use two configurations for mail delivery:

1. **Round-robin**: The workload generator balances mail operations uniformly over machines, which can incur synchronization across nodes when mail directories are accessed concurrently.
2. **Sharded**: The workload is sharded across nodes by mailbox, reducing access to shared directories across file system clients.

Figure 2 shows the throughput of Assise's round-robin and sharded configurations over an increasing number of delivery processes, compared to Ceph. Assise's round-robin configuration scales better than Ceph and is limited only by network bandwidth. Ceph's scalability is limited by central metadata management. Ceph also suffers from kernel overheads and IO amplification, which is why its performance is substantially worse than Assise even at lower process counts. Sharding can achieve even better scalability for Assise, as hierarchical lease delegation can adapt to the application's sharding pattern, eliminating cross-client synchronization. Ceph's performance does not change when sharding.

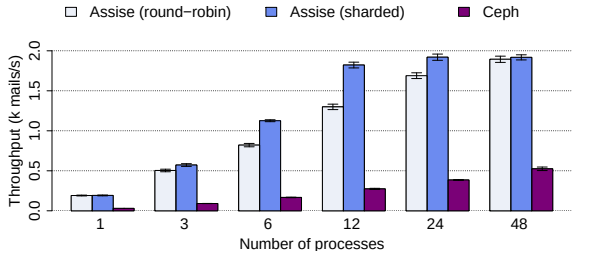

Figure 2. Postfix mail delivery throughput scalability.

4. Conclusion

Assise is a distributed file system that provides low tail latency, high throughput, scalability, and high availability with a strong consistency model. To take advantage of low-latency NVM, Assise demonstrates that filesystem metadata and data should be colocated with applications. Colocation not only enables high performance, but also fast recovery. Assise uses hot replicas in NVM to minimize application recovery time and ensure data availability, while leveraging hierarchical lease delegation to provide scalability with linearizability and prefix crash consistency. In comparing with several state-of-the-art file systems, our results [1] show that Assise improves write latency up to 22×, throughput up to 56×, fail-over time up to 103×, and scalability up to 6× versus Ceph, while providing stronger consistency semantics. Assise is available at `https://github.com/ut-osa/assise`.

References

[1] T. E. Anderson, M. Canini, J. Kim, D. Kostić, Y. Kwon, S. Peter, W. Reda, H. N. Schuh, and E. Witchel. Assise: Performance and availability via client-local NVM in a distributed file system. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*. USENIX Association, Nov. 2020.

[2] V. Chidambaram, T. S. Pillai, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau. Optimistic crash consistency. In *24th ACM Symposium on Operating Systems Principles*, SOSP '13, pages 228–243, 2013.

[3] C. Gray and D. Cheriton. Leases: An efficient fault-tolerant mechanism for distributed file cache consistency. In *12th ACM Symposium on Operating Systems Principles*, SOSP '89, pages 202–210, 1989.

[4] B. Klimt and Y. Yang. The Enron corpus: A new dataset for email classification research. In *European Conference on Machine Learning*, pages 217–226. Springer, 2004.

[5] S. A. Weil, S. A. Brandt, E. L. Miller, D. D. E. Long, and C. Maltzahn. Ceph: A scalable, high-performance distributed file system. In *7th Symposium on Operating Systems Design and Implementation*, OSDI '06, pages 307–320, 2006.

[6] ZDNet, July 2018. `https://www.zdnet.com/article/google-cloud-taps-new-intel-memory-module-for-sap-hana-workloads/`.