FAB Storage for the Hybrid Cloud

Raju Rangaswami
Florida International University

2022 IEEE International Conference on Networking, Architecture and Storage (NAS) | 978-1-6654-5408-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/NAS55553.2022.9925329

***Abstract*—Storage is the Achilles heel of hybrid cloud deployments of workloads. Accessing persistent state over a WAN link, even a dedicated one, delivers an overwhelming performance blow to application performance. We propose FAB, a new storage architecture for the hybrid cloud. FAB addresses two major challenges for hybrid cloud storage, performance efficiency and backup efficiency. It does so by creating a new FAB layer in the storage stack that enables fault-tolerance, performance acceleration, and backup for FAB storage volumes. A preliminary evaluation of FAB's performance acceleration mechanism when deployed over Ceph's distributed block storage system offers encouragement to pursue this new hybrid cloud storage architecture.**

***Index Terms*—Storage, Hybrid Cloud, Fault-tolerance, Performance Acceleration, Backup**

I. Introduction

The world of IT infrastructure has undergone transformational change in the past decade. Public and private cloud adoption within enterprises are at all time highs and projections for hybrid and multi cloud deployments are robust [1]. Interoperability across public and private clouds is a top-level concern among IT administrators who look for workload mobility, consolidated cloud resource management, and cloud "burstability" to support periods of high demand [2–4]. A foundation for this approach to consuming resources and deploying applications with full flexibility is the "hybrid cloud" architecture.

There are several caveats to adopting a public-private *hybrid cloud* solution architecture for IT infrastructure requirements. The first set of concerns address data location. Verticals such as financial institutions, health-care, and even retail are sensitive to externalizing proprietary and customer data [4, 5] Second, cloud vendor lock-in is a risk for enterprises. Customers are forced to pick one public cloud to run their workloads because data has "gravity" and data movement charges are lop-sided [6–8]. Finally, there are concerns related to long term costs with cloud costs tending to increase over time prompting enterprises to shift workloads back to premises [4, 9].

This paper offers foundational principles of *FAB*, a redesign of the conventional storage stack — as defined by its abstractions and architecture — for the hybrid cloud. It decouples storage location from application workload location, ensuring "plug compatibility" for storage consuming applications. The FAB storage stack provides **F**ault-tolerance, **A**ccelerated performance, and **B**ackup for hybrid cloud workloads. Fault-tolerance in FAB rethinks the control and data path for accesses to storage volumes by designing robust recovery protocols that can handle arbitrary software faults, network connectivity loss, and hardware component failures. Accelerated performance in FAB develops a new I/O path for VMs in the public cloud that delivers low-latency and high throughput storage accesses and optimizes wide-area cross data center network data communication to/from storage. Backup for FAB storage volumes eliminates WAN data movement during backup operations while providing disaster recovery properties with tunable recovery point objective (RPO) for the backed-up data volumes.

We conducted a preliminary evaluation of FAB's performance acceleration benefits by implementing the acceleration mechanisms within FAB layered over a Ceph distributed store instance that is accessed over a local area network. In a hybrid cloud deployment the Ceph store would be accessed over a WAN link and the above experimental setup is rather conservative. Even so, we find that the FAB layer leads to up to 50X improvement in block device latencies and block device throughput as measured by FIO [10] and up to 4.4X improvement in transaction throughput with the SysBench OLTP benchmark [11]. Through the rest of this paper, we discuss the architecture and design of FAB, possible limitations of the solution, and directions for future work.

II. Motivation

In a perfect outcome, hybrid cloud storage solutions will deliver seamless fluidity of data and workloads across all the available hardware resources spread across cloud data centers. The necessary decoupling of persistently stored data from application workloads manifests several fundamental requirements. Foremost is the ability to run workloads unchanged; adopting a hybrid cloud architecture should not require rewriting applications and should support interface compatibility across several storage back-ends. Second, any such decoupling should ensure that there is no data loss on failures or faults that are common at scale. Third, any storage solution should ensure support for disaster recovery (DR) via online backup. Finally, to address the latency sensitive workloads of today, there must be a reliable way to overcome most, if not all, of the WAN latency introduced when decoupling storage from compute in a hybrid cloud deployment.

A. State of the Art

A straightforward solution to the first, second, and third challenge above would create a virtual private network (VPN) that spans the public cloud hosting the application workload and the on-premise or external (co-lo hosting) datacenter that hosts the storage infrastructure. This class of solutions is supported by several industry vendors [9, 12, 13]. The solutions work well when applications can tolerate WAN latencies in the I/O path on average. However for storage performance sensitive applications, these solutions are inadequate, even when "direct connect" links across data centers are deployed [14–16].

B. The Distance between Compute and Storage

Storage architectures have undergone significant transformation over the past four decades as illustrated in Figure 1. From individual servers and direct-attached hard drives, enterprises moved to consuming private cloud IaaS whereby applications get to consume scale-up and subsequently scale-out storage that provide enterprise-class features. In private cloud deployments, clients may consume scale-out storage using self-managed solutions such as Ceph [21], Gluster [22], or external vendor scale-out solutions such as VMware's VSAN [23], Nutanix [24], HPE's Simplivity [25], The alternate to scale-out storage is conventional scale-up storage that may have some scale-out capabilities.

The public cloud has witnessed the rapid adoption of cloud-native storage solutions such as AWS EBS [26], AWS S3 [27], Google persistent disk [28], and Azure SimpleStor [29]. We anticipate that the class of applications and workloads that demand either on-premise data deployment or cloud-independent storage deployment will continue to sustain and expand for reasons outlined earlier. Cross-cloud network latencies will dominate as high-performance storage devices (NVMe Flash [19], 3D-XPoint [20]) and lean scale-out software stacks get increasingly commoditized.

C. The Inevitability of Compute-Local Storage

Storage demands that compute be deployed close to itself so workloads may perform acceptably. If data is to be resident in customer-owned infrastructure, running workloads in the public cloud is challenging. Recent efforts [12, 13] attempt to address the performance challenge when accessing storage across data centers with large caches provisioned at the data center running the application workload. Such caches are able to reduce the data movement over the WAN network by a large margin for *read* accesses. These caches, unfortunately, do not address *write* accesses.

Koller et al. demonstrated that relative to write-through caching, write-back caching improves response times by two orders magnitude for transaction processing workloads when caching data for iSCSI-based storage [30]. More recently, Rodriguez et al. have argued for building low-latency data-center scale caches that can absorb writes to slower storage devices within the same data-center [31]. To put this previous work in context for hybrid cloud workloads that access storage across data centers, the network-attached storage system would be accessed over a WAN, further amplifying the need for handling reads and writes locally. Read *and* write caching bring storage working-sets closer to compute and provide a huge I/O performance boost. However, for such caching to be usable in a production environment, it must be infused with high availability (HA) and disaster recovery (DR) capabilities [32–34]. While the demand for compute-local storage in hybrid cloud deployments are rather clear, there is no simple or obvious solution.

III. FAB Cloud Storage Architecture

The evolution of enterprise infrastructure to adopt hybrid IaaS cloud models requires bridging gaps in how storage is architected and delivered, ideally without changing storage consumption models. As discussed previously (§II), any hybrid cloud storage solution must ideally address four requirements to effectively decouple storage from application workloads: *(i)* backwards compatibility of consumption interfaces, *(ii)* fault-tolerance, *(iii)* disaster tolerance, and *(iv)* high-performance. FAB is a new storage stack designed to help meet these requirements.

A. Storage Abstraction

Ideal candidates for FAB are storage abstractions that allow a straightforward decoupling of storage from compute. A well-established service model for the public and private clouds is *Infrastructure-as-a-Service (IaaS)* wherein the consumption of storage by the systems software — bare metal, or virtualized, or containerized — is as a *block device*. This lowest common denominator storage abstraction enables the broadest range of storage provider applications such as filesystems, databases, object stores, or key-value stores.

The block device abstraction has simple semantics and a simple consistency contract. First, the unit of access is a fixed size sector, *512-byte* or *4KB*, and it supports primarily two operations – *read* and *write*. Second, any acknowledged write may not be lost or corrupted. Third, reads must always return the latest

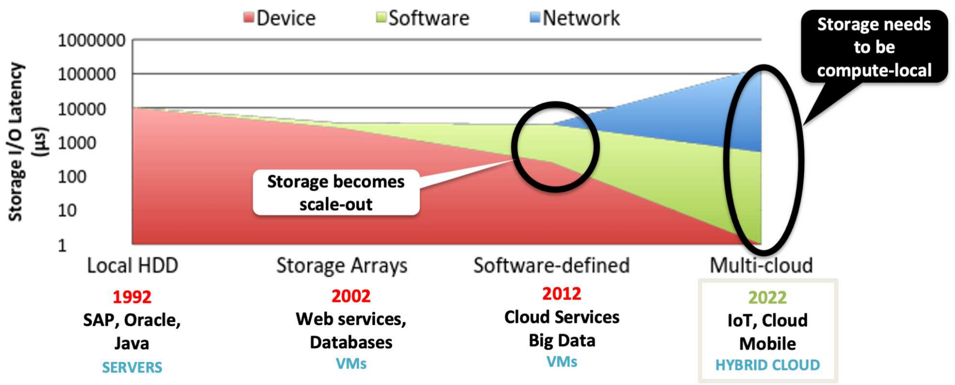


Fig. 1: **Storage Latency Trends.** The storage stack is divided into device, network, and software. Device and software latencies have consistently improved over time [17–20]. Today, hybrid cloud deployments make cross data-center network latencies dominate storage I/O latency.

written version of the block. Fourth, writes issued simultaneously may be executed in any order. Finally, to guard against data loss, a block device typically implements fault-tolerance and disaster recovery capabilities.

The FAB hybrid cloud storage solution is designed around the block device abstraction because it allows for a straightforward decoupling of compute and storage. The block abstraction serves as a simple starting point for optimizations such as read caching and write buffering that are well-understood at the block device layer [35]. The block device also enables simple approaches to build fault-tolerance and disaster recovery capabilities because of its simple interface and consistency contract.

B. Architecture

Workloads running in the public cloud get access to a FAB block device that spans multiple cloud data centers. The FAB storage architecture is designed meet the requirements of hybrid cloud deployments of enterprise applications and services. Creating a decoupling of storage from compute while maintaining performance, fault-tolerance, and disaster tolerance capabilities prompts a rethink of the conventional storage architecture. FAB is a seamless extension of current storage stacks that introduces a new FAB Layer consisting of the FAB Block Device (FBD) and the FAB Cluster, both co-located in the public cloud data center with the storage-consuming application VM. The FBD exports storage to applications and interacts with the FAB Cloud Node VM that is part of a the FAB Cloud Cluster which in turn serves all the FBDs and interfaces with the storage back-end over a high-performance WAN link [14–16]. The FAB Cluster also interfaces with the backup storage service provider which would typically be located in the same cloud datacenter as the workload but it could also be located in a different datacenter. Although the FAB Layer creates backups,

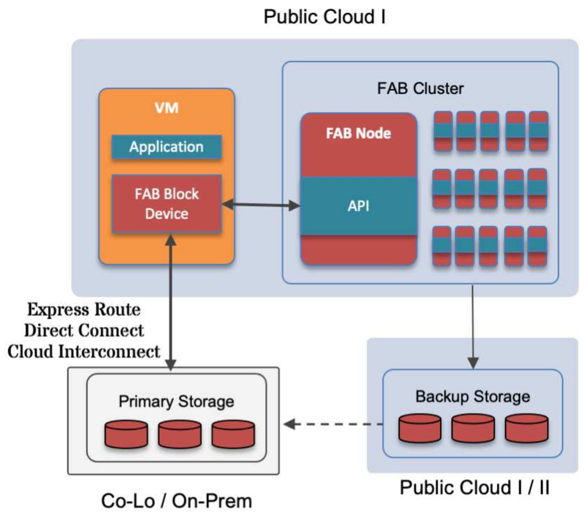


Fig. 2: **FAB hybrid cloud storage solution.**

recovery of the Primary Storage after a failure is done directly from the backup storage, bypassing the FAB Layer. The fault-tolerance of the FAB Layer is the subject of the next section (§IV).

The FAB Layer gets deployed in the public clouds where storage consuming workloads reside. This tier is termed **FAB Layer** because it implements **F**ault-tolerance, performance **A**cceleration, and **B**ackup for FAB storage volumes. Storage consuming applications interact directly with an FBD serviced by a FAB Node that is part of the FAB cluster implementing fault-tolerance, performance acceleration, and disaster recovery capabilities. The FAB Layer and its interaction with both the storage consuming workload and the storage back-end located at the storage owner's data center premises are shown in Figure 2.

The FAB Layer manages primary data in motion, i.e., data that is accessed by the application VMs in the public cloud that reside there either (relatively) permanently or temporarily to handle unexpected loads via cloud bursting. FAB's volume storage is consumed similar to how public cloud VMs consume block storage service (e.g., EBS [26], persistent disk [28], etc.) provided by the public cloud vendors. There are key differences, though. First, FAB is not a storage end-point; it serves as a read cache and a fault-tolerant write buffer. Second, FAB utilizes storage at rest outside of the public cloud in two ways. At rest data is shipped to storage arrays owned and managed by the data owner. At rest data is also backed up to meet a specified *recovery point objective (RPO)* to a cloud backup storage service co-located with the data generating workload or to one that is located at an alternate public cloud provider, as shown in Figure 2. And third, FAB storage volumes offer high performance because workloads perform storage operations over the local network via the FAB Layer for all writes and for reads that hit in the FAB read cache.

In summary, FAB introduces a new public-cloud resident layer to the storage stack for hybrid storage. In hybrid cloud implementations, Primary Storage is expected to manage data at rest at the data owner's datacenter or in external co-location hosting providers. A well-designed Primary Storage would provide *core* storage features including namespace management, access control, encryption, data redundancy, snapshots, and thin provisioning. Additionally, it would implement critical *offline* features of storage including data deduplication, compression, and garbage collection; the division of responsibility as suggested by the core and offline feature set is typical in hyperconverged storage implementations [23, 24]. With FAB, clients can run workloads in any cloud while provisioning storage in a cloud-independent manner, enabling workloads to fully exploit the offerings of the public cloud ecosystem.

IV. Fault-tolerance

The FAB Layer is designed to provide fault-tolerance properties for FAB storage volumes upon failures of FAB block device and other components within the FAB cluster. Such fault-tolerance must ensure data durability and data consistency in spite of software and hardware failures or unavailability at the public cloud layer. Failure tolerance for the Primary Storage is discussed later (§VI).

A. Control and Data Path

Figure 3 presents the three major architectural components relevant to FAB's data path. The FBD serves as a single entry point for FAB storage clients and instantiates the FAB storage volumes consumed within the application VM. The FAB Node implements fault-tolerance, acceleration, and backup for FAB storage volumes. And the Primary Storage (PS) is the final resting place for all persistent data located in a different data center.

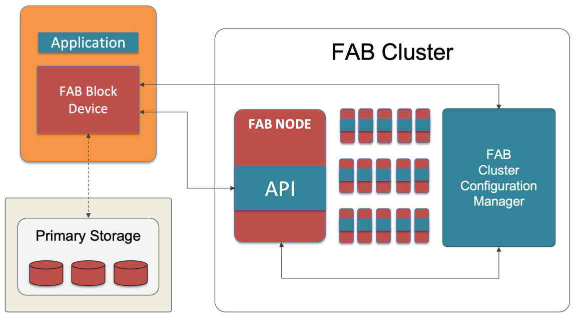


Fig. 3: **FAB's Data-path Architecture.**

The FBD first communicates with a known FAB Cluster Configuration Manager (CCM), a named service that is discoverable, to open the FAB volume on behalf of the requesting application. The CCM is in charge of tracking status information such as the liveness of various FAB entities as well as FBD to FAB Node mappings. Upon receiving an open request for a FAB volume and corresponding Primary Storage instance from an FBD instance, the CCM performs access control and looks up the FBD-FN mappings before responding to the FBD with the FAB Node IP corresponding to the FAB volume, if it already exists, else assigns a FAB Node to the volume dynamically. The CCM is designed as a highly available distributed configuration management service running on the FAB Cluster [36]. Once the FAB node IP is obtained, the FBD is ready to handle I/O requests.

B. Fault-tolerant Design

FAB must address arbitrary failures including client nodes (i.e., FBD failures) and loss of network connectivity to components within the FAB cluster and outside. The design decisions depend on the properties of the underlying infrastructure and the durability requirements of the application managing the data.

First, let's assume that the FBD and the FAB cluster are being managed within independent failure domains. Fault-tolerance of a block devices can be achieved by synchronously replicating writes within the FBD and the assigned FAB node. In case of FBD failure, the FAB Node informs such change of state to the CCM. The FAB Node then starts replicating its *write log* to other nodes in the FAB cluster to protect against a near-term failure within the FAB cluster prior to the recovery of the FBD. The replicas are only necessary until recovery is completed. Consequently, there is little benefit in using data reduction techniques and/or erasure coding. In case of a FAB Node failure, the dependent FBD informs the CCM of this change of status. The CCM chooses a replacement FAB node in the FAB cluster to take over operations for the FBD. The FBD then synchronously replicates the it's local *write log* to the replacement FAB Node prior to handling further I/O for the volume.

Next, let's assume that the FBD and the FAB cluster nodes are in the same failure domain and must tolerate two simultaneous failures. In this case, the FBD and the FAB cluster run a consensus protocol (e.g., Raft [37]) to synchronously replicate writes across two FAB nodes and the FBD itself, thereby creating two additional copies, before acknowledging the client. The CCM mappings are modified to include the full membership information as FBD-FN1-FN2. In case of a FAB node failure, the CCM simply assigns another FAB node to join the consensus cluster. In case of FBD failure, the CCM awaits connection to the same block device from another FBD before informing it of the associated mapping thereby initializing the membership for the consensus protocol. The consensus protocol ensures that the state of the write log of the new FBD is equivalent to the state of the log of the prior FBD instance serving the same block device.

Solutions outlined above guarantee durability of writes in the presence of faults. More aggressive performance optimizations are possible using techniques that effectively shift the durability point from the write to a later time that preserves a useful notion of consistency [38] and/or exploit the asynchrony in failures across different parts of a distributed system [39].

Finally, in case the CCM receives a FAB block device open request for a device that is already being serviced by a FAB Node, it verifies that the FAB block device client is indeed alive. If so, the block device open request is allowed only if the device is being opened in *read-write* mode by at most one of the previously active or newly requesting clients.

V. Performance Acceleration

When building a cross-cloud storage solution, one of the primary challenges is delivering adequate performance while ensuring data durability and availability. This tension is highlighted by an application of Little's Law [40] to storage performance. This application ties together quantities such as I/O latency, I/O throughput, and I/O parallelism and may be expressed as $io\_throughput = io\_paralleism/io\_latency$. Given a workload with a given I/O parallelism level, its I/O throughput can be increased by reducing I/O latency and vice-versa.

A straw man cross-cloud storage solution which handles storage accesses over a WAN link will be severely limited in I/O throughput. Wide-area network links have high latency and demand a lot of workload I/O parallelism to saturate storage throughput. This implies that storage I/O resource consumption is tightly coupled with compute resource consumption as more compute cycles become necessary to generate higher levels of I/O parallelism. Furthermore, I/O parallelism is a property of the application itself and is not easily controlled.

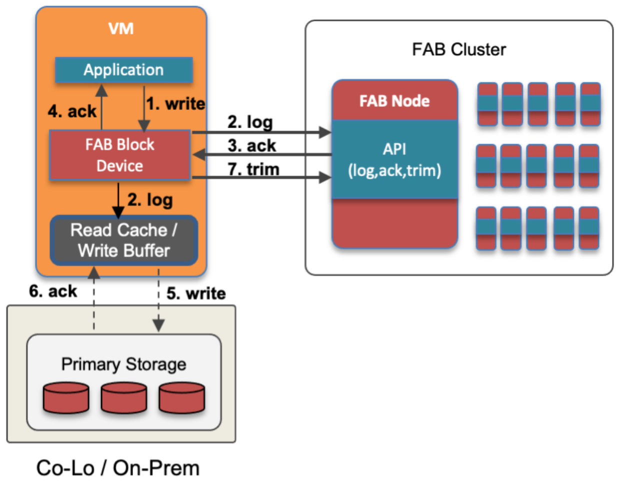


Fig. 4: **FAB's Write I/O Path**

A. FAB I/O Path

The FAB Layer is designed to decouple I/O handling from the access to the Primary Storage over the WAN as much as possible. The FAB Layer's I/O path spans the FBD and its associated FAB node within the FAB cluster. Read I/Os are handled by FBD by consulting in sequence, the local cache first, and then the FAB Node, for cached content. For local read caching at the FBD, we anticipate the use of host-side SSDs as the caching device [30, 41–46]. Aggressive prefetching techniques that allow the handling of all reads from the FBD and/or FAB Node can further reduce WAN accesses for reads.

The write I/O path for FAB storage volume accesses is more involved and is depicted in Figure 4, assuming that the FBD and the FAB cluster reside in independent failure domains. To optimize write I/O latencies, writes are handled at the FBD by simply **log**ging the write to its local write buffer and the FAB Node. Upon receiving an **ack**nowledgment from the FAB Node, the write is acknowledged to the application VM. Next, it is *processed* for transmission to the Primary Storage in the background. Processing for transmission could result in immediate or delayed transmission or even elimination of transmission via *coalescing* as discussed later (§V-B). When such processing is completed for the I/O in question, the FAB Node is sent a *trim* message so that the FAB Node may delete the corresponding I/O payload and entry from its log to reclaim space.

B. WAN Optimizations

Decoupling acknowledgments to applications from updates over the WAN also effectively decouples the I/O parallelism achievable over the WAN link from the application's I/O parallelism. This allows applications

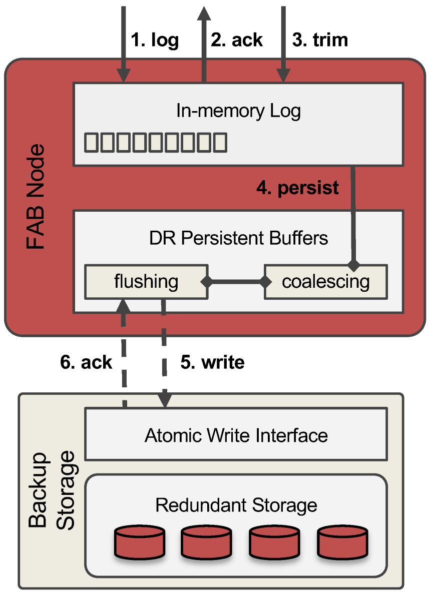


Fig. 5: **FAB's DR path.**

to achieve higher I/O throughput by simply provisioning adequate WAN link bandwidths. However, WAN bandwidths are both expensive and limited. To further decouple application I/O throughput from WAN bandwidths, data reduction techniques including data coalescing, compression, and deduplication are valuable. While compression and deduplication are well-established WAN optimization techniques, data coalescing for storage updates while ensuring no data loss on failures has not been possible. FAB systems can leverage the fault-tolerance capabilities of the FAB cluster to pursue aggressive data coalescing at the FBD prior to WAN transmission.

VI. Back-up for Disaster Recovery

Modern day backup architectures include or are transitioning to include public cloud infrastructure as backup storage providers [47–49]. In hybrid cloud deployments, when backup is deployed for on-prem Primary Storage, the resulting data movement overhead is exorbitant. In particular, data gets moved twice across WAN links, once to move the application generated data from the public cloud to the on-prem Primary Storage and subsequently from Primary Storage to the public cloud storage backup service.

FAB co-designs the backup architecture with the hybrid cloud storage architecture to make backup orders-of-magnitude more efficient. First, by making the public-cloud resident FAB Layer create storage backups, clients can reduce the amount of data moved across WAN links by co-locating the data generating workload with the backup service in the same public cloud data center. Second, since the backup would be performed in the I/O path of the data generating workload, the increased access allows for greater control over the granularity at which backups are created without proportionately increasing resource overhead. This benefit makes features such *continuous data protection* more accessible to production storage workloads, which in turn enable zero RPO backups.

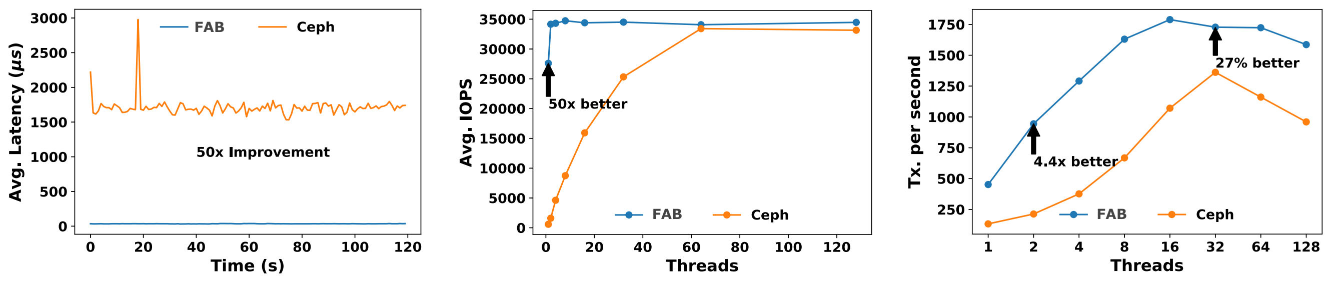
A. Backing Up FAB Volumes

The FAB Layer provides backup for disaster recovery so that the FAB storage volumes continue to be available despite Primary Storage unavailability. FAB enables a range of interesting design choices for backup granularity, tunable RPO, performance, and fault-tolerance. First, because the FAB Layer has access to the I/O stream as it is being modified, there is a compelling opportunity to build backup solutions that offer continuous data protection (CDP) with tunable RPO. With CDP, recovery is possible with little to no data loss on failure. Implementing CDP at the block storage level demands that backups "mimic" the updates to primary storage as closely as possible.

To enable efficient CDP capable backups, FAB employs a tiered buffering architecture at the FAB Node using DRAM, cheaper non-volatile memory [50], and flash-based SSD storage that is even cheaper. The FAB Node uses an in-memory log to buffer "untrimmed" writes. It also implements DR persistent buffers to coalesce over customer defined RPO windows for FAB Backup Storage. Figure 5 provides an illustration. The tiered buffering capability allows for aggressive and tunable coalescing at the FAB node so that FAB volumes can be configured for specific RPO guarantees which may be revised over time depending on customer need. To ensure that the Backup Storage is always point-in-time consistent, an atomic write ingestion mechanism at the backup storage becomes necessary as suggested by Koller *et al.* [30].

VII. Preliminary Evaluation

In this section, we discuss a limited evaluation of the FAB approach to building hybrid cloud storage. The evaluation is limited because it is focused on the performance acceleration aspect of the FAB architecture. In particular, we evaluated the importance of a fault-tolerant and high-performance FAB Layer by comparing the performance of a 4-node 10gE Ceph storage cluster with a similarly configured, but additionally FAB-enabled, Ceph store. More specifically, in the FAB variant, an application VM running the workload communicated with a FAB node to *(i)* log writes synchronously while awaiting responses from the Ceph store asynchronously, and *(ii)* to lookup/fetch reads, which if not found not found were fetched from the Ceph store synchronously. In contrast, the workload running the Ceph configuration directly communicated with the



(a) Flexible I/O Tester latency. (b) Flexible I/O Tester IOPS. (c) MySQL transaction rate.

Fig. 6: **Performance data for LAN deployments of the FAB stack.** The results represent an underestimate of the actual performance difference that may be expected when the primary storage layer is accessed across a WAN link in a hybrid cloud deployment.

Ceph store to perform I/O operations. Experiments conducted at the application VM used the Flexible I/O tester [10] workload generator and the SysBench OLTP benchmark [11].

The default (Ceph) distributed store performs distributed synchronous operations to serve reads and writes. Our goal was to get broad estimates for how accessing data synchronously at the FAB node over a 10gE LAN link performs in terms of storage latency and throughput when compared to accessing the Ceph distributed storage system. It is important to point out that this experimental setup reflects a significant underestimate of the performance potential of FAB's tiered storage architecture. If the Ceph storage cluster located on-prem were to be accessed over a WAN link by a workload running in a public cloud (as in Figure 2), such access would incur substantial additional overhead due to orders of magnitude higher latencies and reduced bandwidth of the WAN link relative to the LAN link as configured in these experiments. Nevertheless, the findings reported in Figure 6 are instructive and inform of the potential for performance acceleration.

FAB's tiered storage architecture leads to 50X improvement in storage latencies relative to a distributed storage system accessed over a local area network. Figure 6c points out that storage I/O performance improvements do not entirely translate to application-level performance improvements due to software overheads and limited inherent I/O parallelism in the workload. Nevertheless, there are significant application-level performance improvements for the SysBench OLTP benchmark. These application-level improvements would of course depending on workload and/or hardware on which the workload is deployed and the actual WAN latencies involved in a hybrid cloud storage deployment. Overall, given these storage latency and throughput improvements, it is reasonable to expect that FAB's performance acceleration oriented design will significantly reduce the impact of a hybrid cloud deployment's limitations on storage performance.

VIII. Discussion and Future Work

FAB is a novel storage architecture for the hybrid cloud where deployments get impacted by high latency and low bandwidth WAN links when accessing persistent data. Our preliminary, underestimated, evaluation of its performance acceleration benefits are highly encouraging and indicate a need for further study of this new architecture.

There are open questions and challenges around FAB that deserve further study. First, as a primary side-effect, FAB does induce data staleness at the primary storage, since in-flight data from the FAB device to the primary storage is persisted asynchronously relative to the application state. One instance of such a requirement is with storage backup for disaster recovery. Fortunately, FAB's native backup solution offers a way forward since it does not depend on the Primary Storage state at all. If there is an unavoidable need for zero staleness at the Primary Storage, additional FAB protocols that flush out dirty data synchronously become necessary.

Second, one variant of FAB's design that we discussed assumes that the FAB node and the FAB cluster are expected to fail non-concurrently with the FBD. If deploying the FAB cluster in the public cloud cannot satisfy this assumption, applications may be exposed to a limited window of vulnerability where some data loss may manifest with this design variant. Alternate consensus-based log state replication within the FAB stack is necessary under these alternate assumptions. The performance impact of this alternate design are likely to be different and would need additional study.

Third, fault-tolerance, consistency, and availability are hard challenges being affected not just by software and hardware failures but also network faults. Too many ad-hoc implementations of distributed storage break in unexpected ways at scale [51]. To avoid falling prey to a hard-to-reason distributed systems implementations over time, formal verification [52] of the fault-tolerance properties of FAB storage is necessary. We believe that the above challenges for FAB are tractable and addressing them will help create robust and performant hybrid cloud storage systems.

Acknowledgment

This work was supported in part by NSF award CNS-1956229 and research gifts from NetApp and Seagate Technology.

References

[1] Microsoft, https://blogs.microsoft.com/wp-content/uploads/prod/2022/01/Microsoft-Cloud-Survey-Results-Final.pdf, January 2022.
[2] J. Barr, "Cloudbursting hybrid application hosting," August 2008.
[3] M. Azure, "What is cloud bursting?" azure.microsoft.com/en-us/overview/what-is-cloud-bursting/, 2022.
[4] VansonBourne and Nutanix, "Enterprise cloud index: Multicloud is here to stay," https://www.nutanix.com/enterprise-cloud-index, 2022.
[5] B. Violino, "11 Top Cloud Security Threats," CSO Online, IDG Communications, Tech. Rep., October 2019.
[6] "AWS Direct Connect Pricing," https://aws.amazon.com/directconnect/pricing/.
[7] "Microsoft Azure ExpressRoute Pricing," azure.microsoft.com/en-us/pricing/details/expressroute/.
[8] "Google Compute Platform Cloud Interconnect Pricing," https://cloud.google.com/interconnect/docs#pricing.
[9] B. Cracco, "Why Build an On-Premises Cloud?" NetApp Blog, Tech. Rep., April 2019.
[10] J. Axboe, "FIO: Flexible I/O tester," https://github.com/axboe/fio.
[11] A. Kopytov, "Sysbench manual," *MySQL AB*, 2012.
[12] VMware, "VMware Cloud on AWS," https://cloud.vmware.com/vmc-aws.
[13] Nimble Storage, "Nimble cloud volumes," www.nimblestorage.com/technology-products/nimble-cloud-volumes/.
[14] "AWS Direct Connect," aws.amazon.com/directconnect.
[15] "Microsoft Azure ExpressRoute," azure.microsoft.com/en-us/services/expressroute/.
[16] "Google Compute Platform Cloud Interconnect," https://cloud.google.com/interconnect/.
[17] M. K. Qureshi, V. Srinivasan, and J. A. Rivers, "Scalable high performance main memory system using phase-change memory technology," *ACM SIGARCH Computer Architecture News*, vol. 37, no. 3, pp. 24–33, 2009.
[18] A. M. Caulfield and S. Swanson, "Quicksan: A storage area network for fast, distributed, solid state disks," in *Proceedings of the International Symposium on Computer Architecture*, 2013.
[19] NVM Express, Inc., "NVM Express Overview (White Paper)," http://www.nvmexpress.org/wp-content/uploads/NVMe_Overview.pdf, 2017.
[20] Intel Corporation, "3D XPoint^{TM} Technology Revolutionizes Storage Memory," http://www.intel.com/content/www/us/en/architecture-and-technology/3d-xpoint-technology-animation.html, 2015.
[21] Ceph, ceph.com.
[22] E. B. Boyer, M. C. Broomfield, and T. A. Perrotti, "GlusterFS: One storage server to rule them all," Los Alamos National Laboratory (LANL), Tech. Rep., 2012.
[23] VMware, Inc., "VMware Virtual SAN," http://www.vmware.com/products/virtual-san/, 2013.
[24] S. Poitras, "The Nutanix Bible," https://nutanixbible.com/.
[25] Hewlett Packard, "HPE Simplivity," https://www.hpe.com/us/en/integrated-systems/simplivity.html.
[26] "AWS Elastic Block Storage," aws.amazon.com/ebs.
[27] "AWS Simple Storage Service," aws.amazon.com/s3.
[28] "Google Compute Platform: Cloud Persistent Disk," https://cloud.google.com/persistent-disk/.
[29] M. Azure, "Storsimple," https://azure.microsoft.com/en-us/services/storsimple/.
[30] R. Koller, L. Marmol, R. Rangaswami, S. Sundararaman, N. Talagala, and M. Zhao, "Write policies for host-side flash caches," in *Proc. of USENIX FAST*, 2013.
[31] L. V. Rodridguez, A. Gonzalez, P. Poudel, R. Rangaswami, and J. Liu, "Unifying the data center caching layer: Feasible? profitable?" in *USENIX HotStorage Workshop*, July 2021.
[32] E. B. Nightingale, J. Elson, J. Fan, O. Hofmann, J. Howell, and Y. Suzue, "Flat datacenter storage," in *USENIX Symposium on Operating Systems Design and Implementation*, 2012.
[33] J. Baker, C. Bond, J. C. Corbett, J. Furman, A. Khorlin, J. Larson, J.-M. Leon, Y. Li, A. Lloyd, and V. Yushprakh, "Megastore: Providing scalable, highly available storage for interactive services," in *Proceedings of the Conference on Innovative Data system Research (CIDR)*, 2011, pp. 223–234.
[34] D. Ford, F. Labelle, F. Popovici, M. Stokely, V.-A. Truong, L. Barroso, C. Grimes, and S. Quinlan, "Availability in globally distributed storage systems," in *USENIX Symposium on Operating Systems Design and Implementation*, 2010.
[35] "dm-cache," http://visa.lab.asu.edu/dmcache.
[36] P. Hunt, M. Konar, F. P. Junqueira, and B. Reed, "Zookeeper: Wait-free coordination for internet-scale systems." in *USENIX Annual Technical Conference*, 2010.
[37] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in *USENIX Annual Technical Conference*, 2014.
[38] A. Ganesan, R. Alagappan, A. Arpaci-Dusseau, and R. Arpaci-Dusseau, "Strong and efficient consistency with Consistency-Aware durability," in *18th USENIX Conference on File and Storage Technologies (FAST 20)*, Feb. 2020.
[39] R. Alagappan, A. Ganesan, J. Liu, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau, "Fault-Tolerance, Fast and Slow: Exploiting Failure Asynchrony in Distributed Systems," in *Proceedings of the 13th Symposium on Operating Systems Design and Implementation (OSDI '18)*, Carlsbad, CA, October 2018.
[40] J. D. C. Little, "A proof for the queuing formula: L = w," *Operations Research*, vol. 9, no. 3, pp. 383–387, June 1961.
[41] S. Byan, J. Lentini, A. Madan, L. Pabon, M. Condict, J. Kimmel, S. Kleiman, C. Small, and M. Storer, "Mercury: Host-side flash caching for the data center," in *Proc. of IEEE Symposium on Mass Storage Systems and Technologies*, April 2012.
[42] D. A. Holland, E. Angelino, G. Wald, and M. I. Seltzer, "Flash Caching on the Storage Client," in *Proc. of USENIX ATC*, 2013.
[43] R. Koller, A. J. Mashtizadeh, and R. Rangaswami, "Centaur: Host-side SSD caching for storage performance control," in *Proc. of the International Conference on Autonomic Computing*, July 2015.
[44] D. Qin, A. D. Brown, and A. Goel, "Reliable Writeback for Client-side Flash Caches," in *Proc. of USENIX ATC*, 2014.
[45] E. Lee, H. Bahn, and S. H. Noh, "Unioning of the buffer cache and journaling layers with non-volatile memory," in *Proceedings of the USENIX Conference on File and Storage Technologies*, ser. FAST '13, February 2013.
[46] R. Santana, S. Lyons, R. Koller, R. Rangaswami, and J. Liu, "To ARC or not to ARC," in *Proc. of USENIX HotStorage*, 2015.
[47] Cohesity, "Google Cloud Backup as a Service," https://www.cohesity.com/solution/backup-and-recovery/cloud-backup-service/, 2019.
[48] Rubrik, "A Modern Approach to Backup and Recovery," https://www.rubrik.com/solutions/backup-recovery/, 2019.
[49] NetApp, "Fully managed cloud backup and restore service," https://cloud.netapp.com/cloud-backup-service/, 2019.
[50] Intel Corporation, "Intel Optane DC Persistent Memory," https://www.intel.com/content/www/us/en/architecture-and-technology/optane-dc-persistent-memory.html.
[51] C. Newcombe, T. Rath, F. Zhang, B. Munteanu, M. Brooker, and M. Deardeuff, "How amazon web services uses formal methods," *Communications of the ACM*, vol. 58, no. 4, pp. 66–73, 2015.
[52] L. Lamport, *Specifying systems: the TLA+ language and tools for hardware and software engineers*. Addison-Wesley Longman Publishing Co., Inc., 2002.