When Cloud Storage Meets RDMA

Yixiao Gao, *Nanjing University and Alibaba Group;* Qiang Li, Lingbo Tang, Yongqing Xi, Pengcheng Zhang, Wenwen Peng, Bo Li, Yaohui Wu, Shaozong Liu, Lei Yan, Fei Feng, Yan Zhuang, Fan Liu, Pan Liu, Xingkui Liu, Zhongjie Wu, Junping Wu, and Zheng Cao, *Alibaba Group;* Chen Tian, *Nanjing University;* Jinbo Wu, Jiaji Zhu, Haiyong Wang, Dennis Cai, and Jiesheng Wu, *Alibaba Group*

https://www.usenix.org/conference/nsdi21/presentation/gao

This paper is included in the
Proceedings of the 18th USENIX Symposium on
Networked Systems Design and Implementation.
April 12–14, 2021

978-1-939133-21-2

When Cloud Storage Meets RDMA

Yixiao Gao[♠♡], Qiang Li[♡], Lingbo Tang[♡], Yongqing Xi[♡], Pengcheng Zhang[♡], Wenwen Peng[♡], Bo Li[♡], Yaohui Wu[♡], Shaozong Liu[♡], Lei Yan[♡], Fei Feng[♡], Yan Zhuang[♡], Fan Liu[♡], Pan Liu[♡], Xingkui Liu[♡], Zhongjie Wu[♡], Junping Wu[♡], Zheng Cao[♡], Chen Tian[♠], Jinbo Wu[♡], Jiaji Zhu[♡], Haiyong Wang[♡], Dennis Cai[♡], and Jiesheng Wu[♡]

[♠]Nanjing University, [♡]Alibaba Group

Abstract

A production-level cloud storage system must be high performing and readily available. It should also meet a Service-Level Agreement (SLA). The rapid advancement in storage media has left networking lagging behind, resulting in a major performance bottleneck for new cloud storage generations. Remote Direct Memory Access (RDMA) running on lossless fabrics can potentially overcome this bottleneck. In this paper, we present our experience in introducing RDMA into the storage networks of Pangu, a cloud storage system developed by Alibaba. Since its introduction in 2009, it has proven to be crucial for Alibaba's core businesses. In addition to the performance, availability, and SLA requirements, the deployment planning of Pangu at the production scale should consider storage volume and hardware costs. We present an RDMA-enabled Pangu system that exhibits superior performance, with the availability and SLA standards matching those of traditional TCP-backed versions. RDMA-enabled Pangu has been demonstrated to successfully serve numerous online mission-critical services across four years, including several important shopping festivals.

1 Introduction

Alibaba Group [12] is a China-based multinational technology company specializing in e-commerce, e-finance, and cloud computing. Numerous companies, including Alibaba, have moved their core business systems onto clouds. As a fundamental part of information technology (IT) infrastructure, a cloud storage provides a storage service to tenants both inside and outside the cloud provider. In 2009, Alibaba introduced Pangu [18], a cloud storage system that has subsequently played a crucial role in many Alibaba core businesses. As of 2020, Pangu has been deployed in hundreds of clusters, and it has been managing hundreds of thousands of storage nodes. Furthermore, it supports the real-time access to exabyte-level data in numerous production environments.

In order to ensure comparability to local physical storage clusters, a cloud storage system must meet the following requirements:

(i) *High performance:* Small latency and high throughput provide competitive advantages across many scenarios.

(ii) *High availability:* System disruptions incur significant financial/reputation loss for both tenants and their cloud providers.

(iii) *Service-Level Agreement (SLA):* A cloud storage system must be resilient, and thus its performance should gracefully downgrade when various software/hardware failures happen.

The rapid advancement in storage media has left networking lagging behind, resulting in a major performance bottleneck for new cloud storage generations. Networking is not a problem for traditional storage systems built with Hard Disk Drives (HDDs). However, the access latency of current Non-Volatile Memory Express (NVMe) disks is at the microsecond level [50] and the total throughput of a storage node can exceed 100Gbps. In contrast, the latency of traditional network stacks (*e.g.*, TCP/IP) can reach milliseconds [13], while the bandwidth per kernel TCP thread is only tens of Gbps at most [51].

Remote Direct Memory Access (RDMA) running on lossless fabrics offers a promising solution to the network bottleneck in cloud storage. By implementing its entire protocol stack on host NICs, RDMA is able to provide both microsecond level access latency and a per-connection throughput of approximately 100Gbps with almost zero CPU consumption [23]. The application of RDMA over Commodity Ethernet (RoCE) in data centers relies on the Priority Flow Control (PFC) mechanism to provide a lossless fabric.

In this paper, we present our experience in introducing RDMA into Pangu's storage networks (*i.e.*, the network among storage nodes). Our objective is to provide an RDMA-enabled Pangu system that exhibits superior performance, with availability and SLA standards equal to that of traditional TCP-backed versions. Our experience spans 4 years and will continue with the development of RDMA. We faced a number of challenges specifically related to cloud storage, with additional problems associated with RDMA. We have developed a number of solutions to allow for RDMA to function in a production-level cloud storage, several of which are engineering-level work-arounds. However, overcoming the aforementioned RDMA issues proves to be a complicated task. Here, we expose the practical limitations of the production systems in order to facilitate innovative research and applications in this area.

In addition to the performance, availability, and SLA requirements, the deployment planning of Pangu at the production scale should consider storage volume and hardware costs. Following the *availability-first* principal, RDMA communication is enabled only inside each *podset* [13]. Such a podset contains a group of leaf switches, and all Top-of-Rack (ToR) switches connected to these leaf switches. The podsets are connected via spine switches. This setting is currently the optimal balance between application demands, performance, and availability/SLA control. Storage node configurations are carefully planned to match the disk throughput with the network bandwidth. We adopt the hybrid deployment of RDMA/TCP in Pangu to exploit TCP as the last resort for the system (§3).

The performance optimization aims to minimize latency while maximizing throughput. We leverage *software-hardware co-design* to minimize performance overhead. We build a software framework in Pangu that integrates RDMA with Pangu's private user-space storage platform designed for new storage media. By eliminating data-copy operations, the latency of a typical block service request is reduced to tens of microseconds. We observed that the memory bandwidth becomes a bottleneck when upgrading Pangu to a 100Gbps network. By exploiting the RDMA features and offloading critical computations, Pangu is able to saturate the underlying networks. Furthermore, we leverage a new thread communication mode in Pangu to reduce the performance pitfall caused by a large number of Queue Pairs (QPs, RDMA connection abstraction) per node (§4).

Previous studies have reported the risks of large-scale RDMA deployment [13]. RDMA-enabled Pangu clusters do encounter such problems, including PFC deadlocks [13], PFC pause frame storms, and head-of-line blocking [27, 44]. We determined several PFC storms to be attributed to a previously unexplored source that consequently invalidates an earlier solution [13]. In order to guarantee availability, we apply the *escape-as-fast-as-possible* design principle to handle PFC storms. We bring up a fine-grained switching mechanism between RDMA/TCP traffic in Pangu and it handles PFC storms regardless of their causes (§5).

In order to meet the SLA standards, we adopt the design principal of *exploiting storage semantics whenever useful* in Pangu. By taking advantage of its ability to control the application layer, Pangu performs the real-time checking and alarming for a large number of storage service and network

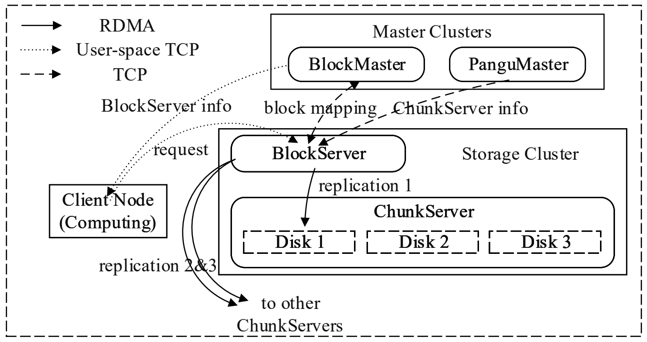


Figure 1: Pangu block storage service framework.

metrics. With the help of the dual-home topology feature, we optimize the fail-over performance of Pangu by reducing the connection recovery time. We also fix network problems by exploiting application controls, for example, blacklisting problematically connected nodes (§6).

We share our experience in adopting the RDMA-enabled Pangu system and discuss several potential research directions (§7). This system has successfully served numerous online mission-critical services under the scope of Alibaba over the past four years, including several important shopping festivals (*e.g.*, Double-11 [8]). Sharing our experience in integrating RDMA into Pangu can be helpful for other RDMA-enabled systems.

2 Background

2.1 Pangu in Alibaba Cloud

Pangu Framework. Pangu is a distributed file system developed by Alibaba Cloud. Released in 2009, it plays a major role in the core Alibaba businesses (*e.g.*, e-business and online payment, cloud computing, enhanced solid state drive backed cloud disk, elastic compute service, MapReduce-like data processing, and distributed database). In this paper, we focus on the network features of Pangu.

Pangu provides numerous storage services, including elastic block service, object storage service, store service, *etc.* We take the block service as an example to demonstrate the system framework. Fig. 1 presents the I/O workflows of Pangu. Virtual block devices contain continuous address spaces that can be randomly accessed by applications. A Pangu client in a computing node organizes data into fixed-sized (*e.g.*, 32 GB) *segments*, while the BlockServers and ChunkServers run on storage nodes. Each segment is aligned to a BlockServer for I/O processing. On the BlockServers, a segment is divided into blocks and replicated to the ChunkServers, which are in charge of the standalone back-end storage of the blocks and device management.

The BlockMasters manage metadata such as the mapping between a segment and its located BlockServer and the BlockServer's living states. The PanguMasters manage the states of the ChunkServers. These master nodes are synchronized using consistent protocols, such as Raft [36].

All data communication in Pangu is in the form of Remote Procedure Calls (RPCs). Each ChunkServer initiates the RPC clients/servers, and storage operations are performed by issuing pre-registered RPCs. An RPC client can simultaneously use different RPC channels (*i.e.*, connections via RDMA, kernel TCP, user-space TCP, or shared memory)according to the required RPCs.

Cloud Storage Requires RDMA. The principal performance metrics for storage services are read/write throughput and access latency. Low latency and high throughput prove to be advantageous for numerous application scenarios. Many customers expect similar performance of the cloud storage to that of the local physical storage. For example, the Alibaba e-commerce database requires extremely low latency in order to ensure fast responses due to the potentially large peak number of transactions per second (*e.g.*, 544,000 orders per second at peak hours [8]). Moreover, the enhanced SSD service promises 1 million IOPS, 4GB/s throughput, and 200µs latency for 4KB random writes [17].

The latency of traditional network stack (*e.g.*, TCP/IP) is generally within hundreds of microseconds [13]. The maximum achievable TCP bandwidth per kernel thread can reach tens of Gbps [51]. In contrast, the access latency of current NVMe SSDs is only at the microsecond level, while the read/write bandwidth of a single device is at the GB/s level [49]. The total throughput of each storage node (generally with 8-16 NVMe disks) can exceed 100Gbps and the incoming Storage Class Memory (SCM, *e.g.*, Intel 3D-XPoint) can even achieve nanosecond level latency [35]. Thus, networking is currently the primary performance bottleneck for cloud storage.

RDMA is an alternative networking choice for cloud storage. By implementing its entire protocol stack on host NICs, RDMA provides both microsecond level access latency and a per-connection throughput close to 100Gbps with almost no CPU consumption [23]. RDMA has successfully been integrated into numerous network-bottlenecked systems, for example, key-value stores [22, 33], distributed transactions [6, 24, 48], and graph queries [40], demonstrating an improved performance compared with non-RDMA predecessors.

2.2 Challenges

Besides performance, availability and SLA are also critical for a successful cloud storage system.

Availability. System disruptions incur significant financial/reputation loss for both tenants and their cloud providers. In 2018, Amazon S3 experienced a system disruption that lasted for 4 hours [2], affecting Amazon Elastic Compute Cloud , Amazon Elastic Block Store volumes, and AWS Lambda [3]. This disruption also had an impact on tens of thousands of websites built on the Amazon storage service, including Netflix [34], Spotify [43], Pinterest [37], and Buzzfeed [5]. Similar events have occurred with Google Cloud and Microsoft Azure [4, 9].

Service-Level Agreement. Software and hardware failures are extremely common in distributed systems. A cloud storage system should exhibit graceful performance downgrade with the occurrence of various failures. Distributed storage systems include mature node monitoring and fail-over mechanisms. A single storage node failure has a minimal impact on the service quality. In our experience, the most challenging aspect of ensuring a stable performance lies in the storage networks. Network failures generally result in a larger affected range compared to storage node failures.

In addition to its superior performance, customers of our RDMA-enabled Pangu require the same levels of availability and SLA standards to that of traditional TCP-backed versions.

2.3 State-of-the-art Work Do Not Fit

Unknown PFC Storm Sources. PFC runs under a hop-by-hop mechanism, with the possibility of PFC storms, spreading into the whole cluster. A PFC storm can seriously affect cluster availability and is the most well-known issue of RDMA. In 2016, Microsoft presented its experience in the deployment of RDMA [13], where they revealed that a bug in the receiving pipeline of an RDMA-capable NICs (RNICs) causes PFC storms. The problem was fixed by building watchdogs on the NICs and switches. However, we identified an additional type of PFC storms that originates from switches, implying the complexity of PFC storms with multifarious sources. The Microsoft solution [13] fails to solve this new problem (§5).

Practical Concerns that Limit Design Options. We are not able to simply treat RDMA as a black-box and wait for future research and technical advances to solve the current problems. Despite the large number of recent studies [10, 22, 29, 33, 40, 48], a production-level comprehensive PFC-free solution is still premature. The application of RDMA over lossy Ethernet has been explored in previous work [7, 11, 15, 26], allowing for the bypass of the PFC mechanism. However, such solutions rely on new hardware features.

The deployment of new hardware is a long process, with several months or even years of testing, followed by the subsequent introduction to business applications. For example, the process of testing Pangu with CX-4 RNICs, a joint collaboration with NIC providers, lasted for over two years. There is a tension between the fast growth of new RDMA demands and the long update cycles of new hardware. To date, these PFC-free proposals are not mature enough for large-scale business deployment, particularly for the availability and SLA standard requirements of cloud storage systems.

Furthermore, large-scale industry deployment is generally associated with multiple generations of legacy RDMA NICs. For example, we have already deployed several Mellanox NIC generations (*e.g.*, CX-4, CX-5), with the number of each reaching tens of thousands. It is operationally infeasible and costly to replace all legacy NICs in the running nodes,

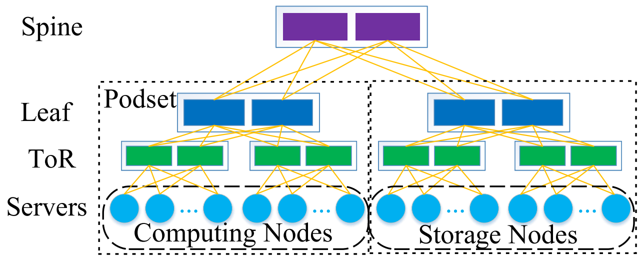


Figure 2: Topology of Pangu.

Hardware	25Gbps	100Gbps
CPU	Xeon 2.5GHz, 64 cores	Xeon 2.5GHz, 96 cores
Memory	DDR4-2400, 128GB	DDR4-2666, 128GB $\times 3$
Storage	1.92TB SSD$\times$12	3.84TB SSD$\times$14
Network	CX-4 Lx Dual-port	CX-5 Dual-port
PCIe	PCIe Gen 3.0	PCIe Gen 3.0

Table 1: Example configurations of 25/100Gbps nodes.

while upgrading the firmware of tens of thousands of running servers is both time-consuming and error-prone. Thus, the need for new hardware features or firmware should be minimized.

Domain Knowledge of Distributed Storage Should be Exploited. Existing work largely ignores potential help from the application layer. Storage service metrics, rather than networking metrics, are a key concern for cloud service applications. We take into account such storage semantics in the design of Pangu when improving the engineering trade-off and the decision making process for various networking problems.

3 RDMA Deployment

3.1 Consideration in Deployment Planning

The deployment planning of storage clusters governs the network topology, RDMA communication scope, storage node configurations, *etc.* Multiple factors must be considered, including matching the storage volume with demands, controlling hardware costs, optimizing performance, and minimizing availability and SLA risks. The final outcome is a trade-off among all these factors.

For example, Microsoft deploys RDMA at the scale of an entire Clos network [13]. Thus, if not prevented, PFC storms could spread across the whole network and bring down an entire cluster. This amount of risk is unacceptable in a production-level storage system.

3.2 Deployment Choices of Pangu

The key principle employed by our RDMA deployment is *availability-first*.

Network and Node Configurations. Fig. 2 displays the Clos-based network topology of Pangu. Consistent with the common dual-home practice, we deploy Mellanox CX series dual-port RNICs to connect a host with two distinct ToR switches. In particular, two physical ports are bonded to a single IP address. Network connections (*e.g.*, QPs in RDMA) are balanced over two ports following a round-robin fashion.

Total bandwidth	TCP bandwidth ratio	TX pauses
25Gbps	40%	0
30Gbps	45%	1Kpps
32Gbps	50%	8Kpps
35Gbps	46%	15Kpps

Table 2: TX pauses in hybrid RDMA/TCP traffic.

When one port is down, the connections on this port can be migrated to another port.

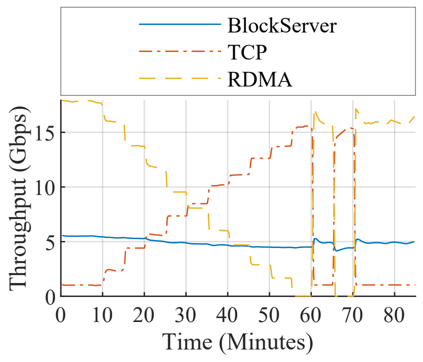
Table 1 reports typical hardware configurations for 25Gbps and 100Gbps RNIC storage nodes. The number of SSD per node is determined by the total RNIC bandwidth versus the throughput of a single SSD, allowing the I/O throughput to match the network bandwidth. Note that the SSD types in the 25Gbps and 100Gbps configurations are distinct, resulting in disproportional numbers. Computing and storage nodes are deployed in different racks within a single podset. The numbers of computing and storage nodes are then calculated according to the computational demands.

RDMA Scope. In order to minimize the failure domain, we only enable RDMA communication within each podset and among storage nodes. The communication between computing and storage nodes is performed via a private user-space TCP protocol (Fig. 1). This is attributed to the complex hardware configurations of computing nodes, which update rapidly. Thus, TCP can be effectively applied as a hardware-independent transport protocol. User-space TCP is more convenient for upgrade and management compared to kernel TCP, while kernel TCP is selected for cross-podset communication due to its generality.

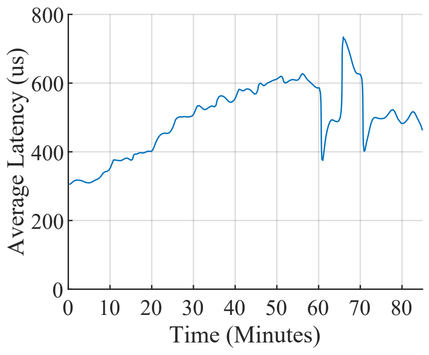
The production deployment is an additional concern for podset-level RDMA. In many datacenters, podsets are located in different buildings. For cross-building RDMA links, the base link delay is much larger, while the PFC mechanism requires much larger headroom buffer. In order to enable RDMA, the PFC/ECN thresholds located on the spine switches must be carefully adapted and tested. This is a tough task and at present, does not result in sufficient gains.

RDMA/TCP Hybrid Service. To the best of our knowledge, previous research on RDMA deployment does not explore RDMA and TCP hybrid services. We keep TCP as the last resort in Pangu following the *availability-first* principal. Despite current progress, RDMA devices are far from flawless. Thus, when either availability or SLA are threatened, switching affected links from RDMA to TCP can maintain the available bandwidth. This escape plan does not impact the unaffected RDMA links.

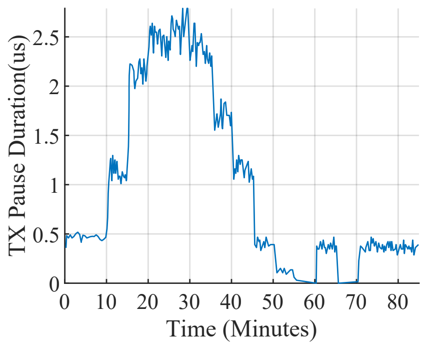
However, during the hybrid deployment process, we determined that coexistent TCP traffic provoked a large number of TX pauses (*i.e.*, PFC pause frames sent by NICs), even if RDMA/TCP traffic are isolated in two priority queues. Table 2 reports the TX pause generation rate in Pangu under different loads with approximately 50% TCP traffic. The tests are performed on Mellanox CX-4 25Gbps dual-port RNICs. Such a large number of TX pauses are detrimental to the



(a) Throughput of RDMA/TCP/BlockServer.



(b) Latency of BlockServer requests.



(c) Average TX pause duration.

Figure 3: RDMA/TCP hybrid deployment tests at different ratios (from 0% to 100% TCP).

performance and may result in PFC storms. We investigated this problem together with Mellanox and determined that the processing of TCP in the Linux kernel is highly I/O-intensive. Kernel TCP initiates too many partial writes on NICs' PCIe bus. As the PCIe bandwidth is consumed, the receiving pipeline of a NIC is slowed down. The buffer overflows and the NIC subsequently begins to transmit PFC pause frames.

In order to optimize the memory access of TCP, we make several adjustments on the data access procedure. First, disabling the Large Receive Offset (LRO) can reduce the memory bandwidth usage. This is attributed to the access of multiple cache lines when the LRO is enabled. Furthermore, enabling NUMA also improves the efficiency of memory accesses, which subsequently aids in relieving the pressure of PCIe. We also allocate a larger buffer on the RNICs for RDMA traffic to prevent TX pauses. Finally, making application data cacheline-aligned is a common optimization practice that improves memory efficiency [23].

3.3 Evaluation

We test several RDMA/TCP traffic ratios to investigate the effects of RDMA/TCP hybrid deployment. Each computing node runs FIO with 8 depths (inflight I/O requests), 8 jobs (working threads), and 16 KB block size in order to write virtual disks. Note that one write request on a BlockServer generates three data replicas. We enable all optimizations approaches detailed in §3.2 for the TCP kernel.

Fig. 3(a) depicts the BlockServer bandwidth with varying RDMA/TCP ratios. The workload starts at 10 minutes with 100% RDMA traffic. Afterwards, in every 5 minutes, the workload contains 10% more TCP traffic and 10% less RDMA traffic. At 60, 65, 70 minutes we change the TCP traffic ratio to 0%, 100%, and 0% respectively in order to explore the performance of Pangu with quick traffic switching between RDMA and TCP. The average BlockServer throughput exhibits minimal reduction as the RDMA traffic ratio decreases.

Fig. 3(b) presents the BlockServers' average request latency for the same workload as that in Fig. 3(a). The average latency under 100% RDMA traffic is approximately half of the latency under 100% TCP traffic, while the tail latency under 100% TCP is more than 10× larger compared to 100% RDMA traffic. RDMA presents great latency advantages compared to TCP. Fig. 3(c) demonstrates the average TX pause duration per second for this workload. Only a limited number of TX pauses are observed. When the TCP bandwidth ratio is around 50% at 30 minutes, the pause duration reaches a peak value.

Overall, these results demonstrate the stable performance of our RDMA/TCP hybrid mechanism.

4 Performance Optimization

4.1 Performance Hurdles

The performance optimization of Pangu aims to *minimize latency while maximizing throughput*.

RDMA-Storage Co-Design. Integrating the RDMA protocol stack with the storage backend is challenging. It must cover key performance points such as thread modeling, memory management, and data serialization. The thread model directly affects latency due to communication costs among threads. Well-designed memory management and data serialization are key to achieving zero-copy during data access. Here we present a brief introduction on the design of these components for storage purposes.

The User Space Storage Operating System (USSOS) is a unified user-space storage software platform that aims to support new storage media such as NVMe SSD and persistent memory. Its design principles (*e.g.*, memory management, shared memory mechanism, and user-space drivers) are based on well-known user-space technologies (*e.g.*, DPDK [19] and SPDK [42]). Related tests reveal that enabling USSOS in Pangu can improve CPU efficiency by more than 5× on average.

As a central part of USSOS, the User Space Storage File System (USSFS) is a high-performance local file system designed for SSDs. By running in the user space, USSFS is able to bypass the kernel to avoid user-/kernel-space-crossing overhead. USSOS divides disks into "chunks" which ChunkServer uses in its APIs (*e.g.*, create, seal, and delete).

Components	Average Utilization	Peak Utilization	Maximum Physical Capacity
Physical CPU utilization ratio	66%	70%	100%
Memory read/write throughput	28GB/s / 29GB/s	33GB/s / 32GB/s	61GB/s in total (1:1 read/write)
SSD PCIe throughput (socket 0 + socket 1)	550MB/s + 550MB/s	1000MB/s + 1000MB/s	3.938GB/s + 3.938GB/s
Network PCIe RX throughput	10GB/s	11GB/s	15.754GB/s
Network PCIe TX throughput	8GB/s	9GB/s	15.754GB/s

Table 3: Measured resource utilization of Pangu in 100Gbps network with 1:1 read/write ratio.

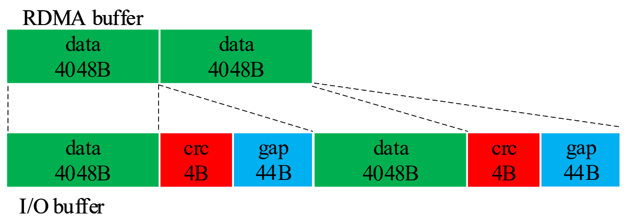


Figure 4: Potential triggering of data copying by CRC.

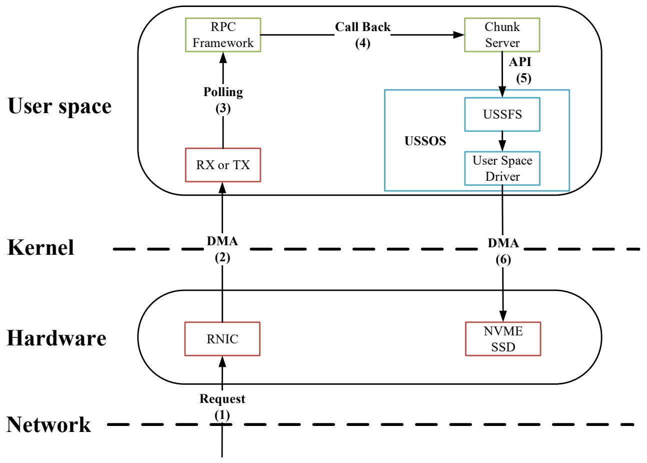


Figure 5: Integrated network/storage processing.

USSOS directly writes data and metadata to disks and uses polling to perceive completion events. For different block sizes, USSFS is able to improve IOPS by 4-10× compared to the Ext4 file system.

A run-to-completion model is considered as the optimal approach for the integration of the RDMA network stack with the storage stack. This model has previously been explored in studies discussing disaggregated storage (*e.g.*, Reflex [25], i10 [16]). However, these studies were published after the introduction of RDMA to Pangu in 2017. Reflex and i10 focus on remote direct I/O while a ChunkServer in Pangu is applied as a local storage engine for distributed storage. Google's Snap [31] leverages a separate network process to unify network functionalities and reduce the number of network connections.

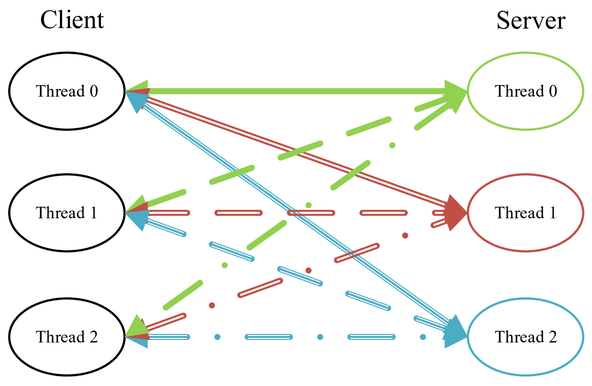
Memory Bottleneck with 100Gbps networks. Deploying 100Gbps networks can achieve lower latency and higher throughput. With faster network, now the memory throughput becomes a bottleneck in our system.

In order to obtain the upper bounds of the memory access throughput, we test the memory throughput using the Intel Memory Latency Checker (MLC) tool [20]. Table 3 details the measured usage of the hardware resources. In our test, the maximal achievable memory bandwidth is 61GB/s with a 1:1 read/write ratio. However, the average memory throughput with Pangu's workload is already $29\text{GB/s} + 28\text{GB/s} = 57\text{GB/s}$. This indicates the memory to be the bottleneck rather than the network.

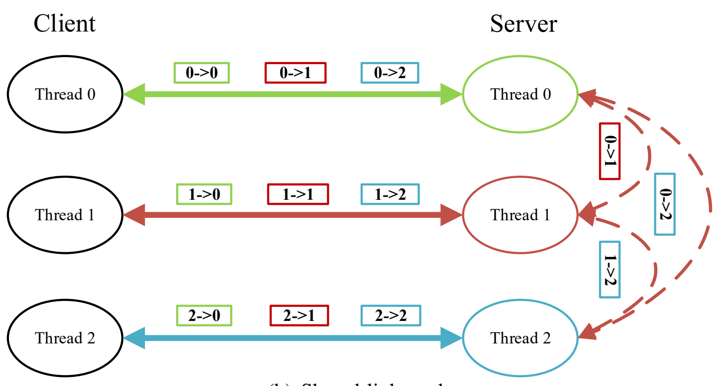
By monitoring the memory usage in Pangu, we determined that both the verification and data copy processes require optimization. Data integrity is one of the most significant features of distributed storage. We adopt Cyclic Redundancy Check (CRC) for application-level data verification in Pangu. As shown in Fig. 4, the received data is split into chunks of 4KB, with a 4B CRC value and a 44B gap added to each chunk. This operation is a memory- and computation-intensive operation as the calculations are applied to the entire dataset. The data are also copied when they are written into the disks in order to include CRC footers. Copying is not performed in other components due to the remote-memory access semantic of RDMA.

Large Number of QPs. We used to adopt the full-mesh link mode among running threads in Pangu in order to maximize throughput and minimize latency (Fig. 6(a)). Assume that each ChunkServer has 14 threads, each BlockServer has 8 threads, and each node contains both ChunkServers and BlockServers. For the full-mesh mode in a cluster of 100 storage nodes, there could be $14 \times 8 \times 2 \times 99 = 2,2176$ QPs in each node. RNICs' performance drop dramatically for large numbers of QPs due to cache miss [21]. In particular, the number of RX pauses (*i.e.*, PFC pause frames received) is very high.

Previous studies have demonstrated the same issue [10, 23, 47]. In order to solve this problem, FaSST [24] shares QPs among threads, which subsequently lowers the CPU efficiency and performance due to the lock contention of QPs between threads. An alternative heuristic is the inclusion of a dedicated proxy thread that manages all receive and send requests [41]. However, switching to/from a dedicated proxy thread increases latency. Furthermore, it is difficult to saturate the full network bandwidth with a single thread. Moreover, the proxy solution is not transparent to the underlying RDMA libraries.



(a) Full-mesh mode



(b) Shared-link mode

Figure 6: Different link modes for send/receive RPC requests

4.2 Designs

The designs related to performance in Pangu are based on the principle of *software-hardware co-design to minimize performance overhead*.

Storage-RDMA Unified Run-to-Completion Stack. We adopt a run-to-completion thread model for both storage and network to achieve low latency. Fig. 5 demonstrates the procedure used to process requests. When a write RPC is received by a node, the RNIC posts it to the user space via DMA. The RPC framework obtains the request using polling and subsequently hands it over to a ChunkServer module for processing. The ChunkServer then informs USSFS to allocate a "chunk" resource to the request. Finally, a user-space driver interacts with NVMe SSDs to store the data. These operations are generally performed in a single server thread without thread switching. This run-to-completion model minimizes the latency. In order to reduce the blocking time caused by large jobs, large I/O requests are split into smaller requests when submitted by applications. This optimization ensures a quick response to I/O signals. An additional optimization strategy for large I/O requests involves the passing of auxiliary work (*e.g.*, formatting and CRC calculation) to non-I/O threads, where they are subsequently processed. These optimizations reduce the average latency of a typical storage request (*e.g.*, 4KB size) to less than 30µs.

The data formats are unified as I/O vectors. An I/O vector is transmitted without copying via a single RDMA verb using scatter-gather DMA (the transfer of discontinuous data through a single interruption) in network. Serialization is not necessary due to RDMA semantics.

Zero-Copy & CRC Offloading. As discussed in §4.1, in Pangu, data has to be copied once on the I/O path as each 4KB chunk is verified and attached with a CRC footer. Here, we leverage the User-Mode Memory Registration (UMR) [32] feature of RNICs to avoid such data copy. UMR can scatter RDMA data on the remote side through the definition of appropriate memory keys. Thus, data can be formatted and organized according to storage application formats. We use UMR to remap the continuous data from the sender into an I/O buffer at the receiver, which contains 4KB data, a 4B footer, and a 44B gap in each unit. Following the CRC calculation, the filled I/O buffer can be directly applied for disk writing. Besides, the CRC calculation is able to be offloaded to capable RNICs (*e.g.*, Mellanox CX-5), thus lowering CPU and memory usage. The 4KB data are posted to the RNIC and the 4B CRC checksum is then generated.

Shared Link. We adopt the shared link mode, an effective solution for reducing the number of QPs in Pangu. The shared link mode is implemented in the application layer and leaves RDMA libraries untouched. A *correspondent* thread in the destination node is assigned to each thread in the source node (Fig. 6(b)). The thread's requests to the node are sent to its correspondent thread, which subsequently dispatches requests to correct target threads.

Consider a daemon with N threads, each thread polls N request/response queues to obtain the requests/responses. Note that there is only a single producer/consumer for each request/response queue. Thus we use lock-free queues for each request/response queue to avoid contention. According to our test, this design adds approximately 0.3 µs latency.

In the shared link mode, there is resource overhead at the correspondent thread during request dispatching when the source thread sends too many requests. Pangu supports shared groups, where threads in a node can be divided into several groups. A correspondent thread only relays requests for its group members. Returning to the previous example, the number of QPs in the *All Shared* mode is now reduced to $(8+8) \times 99 = 1,584$. If the threads are divided into 2 shared groups, the number of QPs will be $(8 \times 2 + 8 \times 2) \times 99 = 3,168$.

4.3 Evaluation

Zero-Copy & CRC Offloading. We use FIO with 16 jobs and 64 I/O depth to test a virtual I/O block device on a single ChunkServer. Fig. 7(a) demonstrates the memory bandwidth usage (including read/write tasks) when UMR zero copy and CRC offloading are used. The memory bandwidth usage is reduced by more than 30%, revealing that these measures are able to relieve the pressure of memory usage. Fig. 7(b) depicts

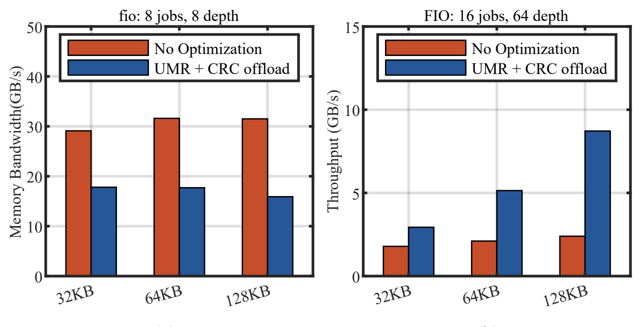


Figure 7: Performance of UMR zero copy + CRC offloading

the improvement in throughput following the optimization. The throughput of a single ChunkServer thread is improved by approximately 200% for a block size of 128KB.

Shared Link. We tested the shared link mode with several shared QP groups in a cluster of 198 computing nodes and 93 storage nodes. The background workload compromises 4KB random writes with 8 threads and 8 I/O depths. Fig. 8(a) presents the throughput in the All Shared, 2 Shared Groups, and 4 Shared Groups modes, whereby a performance trade-off can be observed. The All Shared mode exhibits slightly lower throughput but generates the lowest number of PFC pauses. Note that the reduction in bandwidth at 5 and 24 minutes in the All Shared mode is attributed to the garbage collection mechanism in Pangu. Fig. 8(b) presents the TX pause duration with 1, 2, and 4 Shared Groups, respectively. The lower the number of groups, the fewer the PFC pauses are generated due to the reduction in QP number. We use the All Shared Group mode in our scale and configuration framework.

5 Availability Guarantee

5.1 PFC Storms

A New Type of PFC Storm. The PFC storm previously discussed in [13] originates from the NICs, with a bug in the receiving pipeline acting as the root cause. Fig. 9(a) depicts the phases of this PFC storm: (1) The bug slows down the NIC receive processing, filling its buffer; (2) the NIC transmits the PFC pauses to its ToR switch in order to prevent packet drop; (3) the ToR switch pauses the transmission; (4) the ToR switch's buffer becomes full and starts to transmit the PFC pauses; and (5) the victim ports are paused and are unable to transmit.

We encountered a different type of PFC storm when operating RDMA in Pangu. The root cause is a bug in the switch hardware of a specific vendor. The bug reduces the switching rate of the lossless priority queues to a very low rate. Fig. 9 compares the two types of PFC storms. As an example, we assume that the bug occurs in a down port of a ToR switch: (1) due to the low transmitting rate, the switch's buffer becomes full; (2) the switch transmits the PFC pauses to the connected ports; and (3) the additional switches and NICs stop the transmissions. The leaf switches and NICs connected to this ToR switch receive continuous pause frames and thus the storm spreads.

State-of-the-Art Solutions. Guo *et al.* [13] built a NIC-based watchdog to continuously monitor transmitted pause frames, disabling the PFC mechanism if necessary. In addition, watchdogs were also deployed on the switches for disabling the PFC mechanism when switches receive continuous pause frames and are unable to drain the queuing packets. The switches can subsequently re-enable PFC in the absence of pause frames over a specific period of time. Thus, PFC storms can be controlled via these two watchdogs during phase (2).

However, this solution is unable to completely solve the PFC storms originating from switches. In particular, the TX pause watchdogs on the NICs will not work since the NIC only receives PFC storms from the switches. Furthermore, current switch hardware does not support the monitoring of pause frame transmissions. If a storm occurs on a ToR switch, even though the watchdogs on other switches are able to stop its spread, the ToR switch will continue to send pauses to end-hosts in the rack. The RDMA traffic via this ToR is consequently blocked.

Challenges. This new type of PFC storms invalidates Guo *et al.*'s solution, which focuses on insulating the PFC pause sources to prevent the storm from spreading. This methodology fails when the source is a ToR switch as all the hosts in the ToR are paused by the storm. Therefore, in order to achieve high availability, a general solution is required in Pangu to handle all PFC storm types, particularly those with unknown causes.

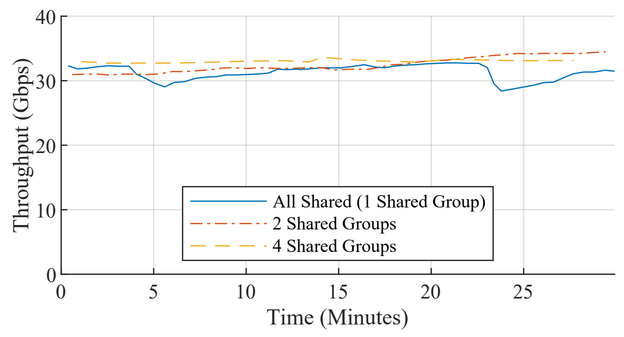
Ensuring the service quality of Pangu while simultaneously solving PFC storms is challenging. PFC storm detection must be timely and accurate to rapidly protect network traffic. In terms of availability, the overall convergence time of the PFC storm should be controlled to at most the minute level.

5.2 Design

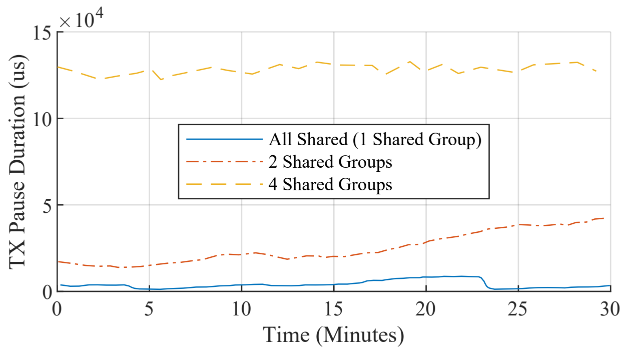
Our design principle of handling PFC storms is *escaping as fast as possible*. Despite new PFC storm solutions [11, 21, 26], we still resort to engineering-level work-arounds due to practical considerations (§2.3).

In Pangu, each NIC monitors the received PFC pause frames. For continuous pause frames, the NIC determines the presence of a PFC storm. Two work-around solutions are available for administrators in the case of a PFC storm.

Workaround 1: Shutdown. This solution, denoted as the "shutdown" solution, shuts down NIC ports affected by PFC storms for several seconds. The dual-home topology provides an emergency escape for PFC storms, whereby QPs will disconnect and connect again via another port. This method works together with the optimization to reduce the length of the QP timeout. This optimization is discussed further in §6.2. Although this solution is simple and effective, it is sub-optimal due to the loss of half of the bandwidth.

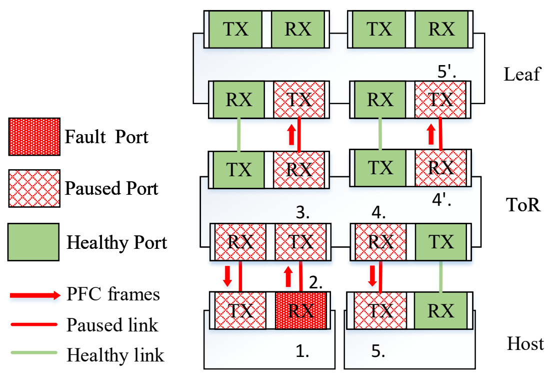


(a) Throughput in different thread groups.

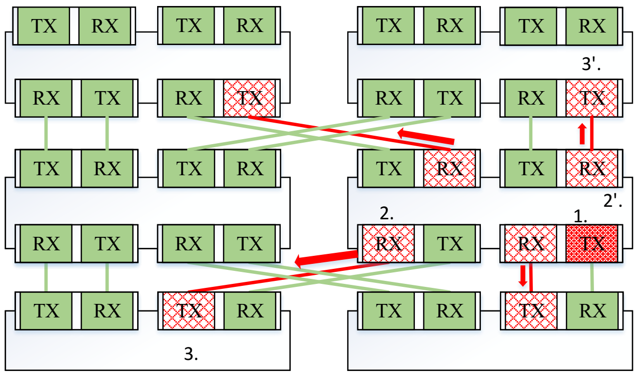


(b) Pause time in different thread groups.

Figure 8: Throughput and pause for different types of thread groups.



(a) The PFC storm originates in NICs.



(b) The PFC storm originates in switches.

Figure 9: Different types of PFC storms.

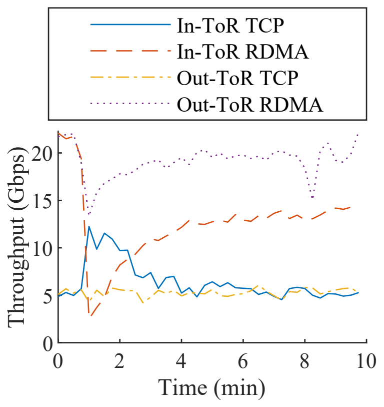
Workaround 2: RDMA/TCP Switching. In this solution, the affected RDMA links in a PFC storm are switched to TCP links. It compromises a more complex procedure compared to the shutdown solution, yet it is able to maintain the available bandwidth. We adopt a method similar to PingMesh [14] to detect the RDMA links affected in PFC storms. At each T ms, every worker thread picks a server and separately pings all its threads via the RDMA and TCP links. If the RDMA ping fails and the TCP ping succeeds for more than F times, the traffic on this RDMA link is switched to the TCP link. Once the RDMA ping has succeeded more than S times, the traffic on the switched TCP link is switched back to the RDMA link. For $T = 10$ ms and $F = 3$, bad RDMA links can be detected in approximately 10 seconds in a podset of 100 storage nodes. By switching the RDMA traffic to the TCP connections, the throughput can recover to more than 90% in less than 1 minute.

5.3 Evaluation

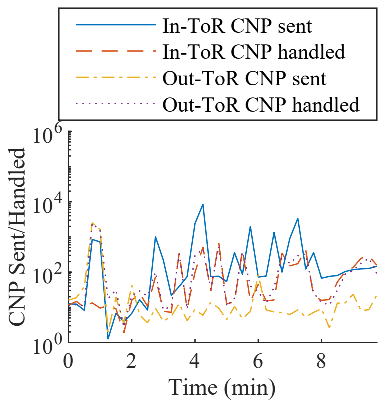
We simulate PFC storms by injecting the aforementioned bug into a switch for several cases, including the uplink/downlink ports on the ToR and Leaf switches. The RDMA/TCP switching solution exhibits strong performance for all cases. Fig. 10 displays the results for a PFC storm originating from a ToR switch downlink port. Note that the nodes inside the ToR behave differently from nodes outside the ToR. We choose two nodes (inside and outside the ToR) in order to demonstrate the difference. In such a case, the pause frames are transmitted to NICs and leaf switches directly connected to the given ToR switch.

The shutdown solution shuts down the NICs via the watchdogs in the occurrence of a fault due to excessive RX pauses. RDMA links subsequently reconnect through another NIC port, thus recovering traffic. Note that the counters of Congestion Notification Packet (CNP) and PFC frames gradually increase since the system load (at 0 minutes) is larger than the available bandwidth of a single port (25Gbps). The system then reaches a new balance in approximately 30 minutes. However, the shutdown solution has several limitations. For example, computing node requests may not respond within 1 minute (known as I/O hang sensed by applications). The downlink breakdown of a leaf or ToR switch can result in tens to hundreds of hang requests. Furthermore, the shutdown of ports is itself an aggressive action. Hundreds of ports may be shut down due to unexpected pauses. This risk may itself influence the availability of a large number of nodes.

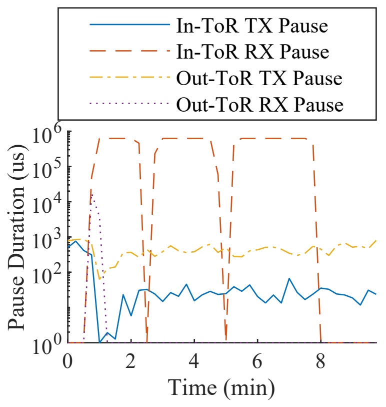
The RDMA/TCP switching solution switches the RDMA traffic that passes through the broken-down switch to TCP.



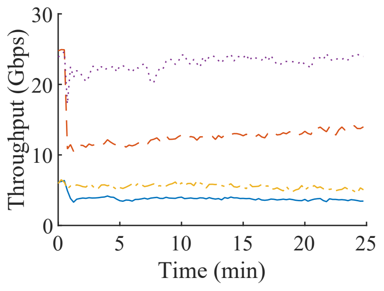
(a) Throughput of RDMA/TCP switching.



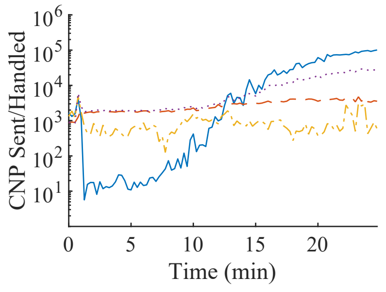
(b) CNP of RDMA/TCP switching.



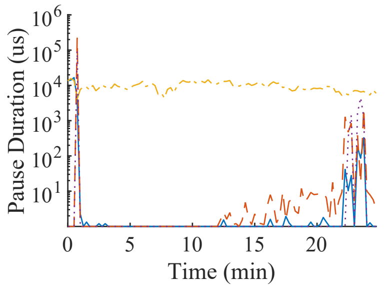
(c) PFC Pause of RDMA/TCP switching.



(d) Throughput of shutdown.



(e) CNP of shutdown.



(f) PFC Pause of shutdown.

Figure 10: Performance of two different solutions for PFC storms.

The RDMA links are then disconnected due to timeout. The QPs are separately distributed over the server's two NIC ports, thus the RDMA links may need several attempts to reconnect successfully. Note that although the pause storm in the ToR is not terminated, it will not spread further as the neighboring switch ports are transformed into the lossy mode via the RX pause watchdogs. The traffic throughput is not impacted during the migration to the TCP, and I/O hangs are not present.

6 SLA Maintenance

6.1 SLA in Pangu

It is commonly-known that network failures are hard to locate and repair. Network failure causes include mis-configuration, hardware/software bugs, and link errors. For example, the mis-configuration of the switch Access Control List (ACL) may only affect a specific flow while other flows behave normally [28, 45]. As a comparison, malfunctions occurring at storage devices or nodes can generally be easily located.

Sometimes network failures may not be explicit. Ideally, when a node breaks down, the heartbeat mechanism should ensure that the unavailability of service daemons (BlockServers and ChunkServers) on the node are informed to their masters (BlockMasters and PanguMasters). However, real situations can be more complicated, failing to be detected with just heartbeats. Connections may suffer intermittent packet loss or long latency rather than simple break downs. We also identified an interesting failure type involving a small number of links that flap between up and down states for a short period of time (*e.g.*, several seconds). This results in an extremely high tail latency for I/O requests, denoted as slow I/O (*e.g.*, over 1 second for storage clients). Hundreds of slow I/Os are observed daily for numerous reasons. Root causes of link flapping include optical ports covered with dust, loose physical interfaces, aging hardware, *etc.*

Previous Research on Network Failures. The majority of previous studies focus on determining the location of network failures (*e.g.*, Pingmesh [14] and 007 [1]). These solutions focus on the system network and can achieve the timely discovery of network errors. However, it may still take hours for engineers to manually check, fix, and replace the failed switches and links. Cloud storage calls for a methodology that integrates the storage and network function modules to ensure stable service quality in failed cases.

6.2 Design

Our SLA design principle aims to *exploit storage semantics whenever useful*. Distributed storage is designed with a redundancy mechanism and its performance is measured via the storage semantics. These semantics, such as data replicas and distributed placements, can be leveraged during a failure to improve system performance.

Network-Integrated Monitoring Service. Monitoring is a necessary component of distributed systems. A comprehensive monitoring system is key for the safe deployment of RDMA in production storage systems, as well as reliable performance.

Both configurations and counters must be monitored. NIC configurations include system environments, PFC/ECN/QoS configurations, and several link configurations. Pangu automatically checks, collects, and reports suspicious terms. Inspecting potential mis-configurations can reduce configuration and environment errors. For example, accidental reboot and software upgrades may reset the QoS, PFC, DCQCN [52] configurations and affect system performance. Monitoring can discover such cases and help fix them in advance.

Counters include storage system performance counters (*e.g.*, IOPS/latency) and network counters (*e.g.*, CNP sent/handled, TX/RX pauses, and RDMA errors on NICs). Congestion control counters that exceed thresholds can result in monitor daemons sending alarms to engineers for diagnosis and repair. Monitoring both storage and network indexes is crucial for the diagnosis of errors and for predictable performance. Storage indexes such as tail latency, slow I/O, and queuing time can directly reflect the status of a system. Moreover, monitoring system performance also help locate errors. For example, network features are unable to quickly reflect the flapping problem described in §6.1. However, this problem can be easily located by monitoring slow I/Os on the endpoints of storage applications.

Faster Timeout for Connection Failures. The basic solution to network failures is to reconnect through an alternative path. Since we use dual-home topology, each single point failure of the network can be bypassed using a different path. Thus, the timeout duration of the QPs is crucial in improving the system performance during a failure. In the NIC manual, the timeout duration of QPs is calculated as $4\mu s \times 2^{\text{timeout}} \times 2^{\text{retry\_cnt}}$, where $timeout$ and $retry\_cnt$ denote the retransmission timeout value and the retransmission retry times respectively. Initially, this value was a constant (approximately 16 seconds) configured in the hardware and cannot be changed. In a combined effort with the NIC providers, we were able to fix this bug. By using a smaller timeout value for QPs, the action time required for reconnecting during network failures was reduced by 4×.

An alternative work-around involves altering the connection path by modifying the source ports of the QPs (rather than a direct reconnection). This can accelerate the link recovery during a fail-over. However, effectively changing the QP source port requires a more recent NIC firmware (MLNX OFED 4.7 or newer) than what is currently deployed in Pangu. We leave this challenge to future work.

Blacklist. We adopt *blacklist* in Pangu to further improve the service quality in fail-over cases. BlockMasters collect information on I/O errors (including timeout) and slow I/Os from clients. If a BlockServer has a large number of slow/error I/Os from multiple clients, the masters adds it to the blacklist for a short period of time. The number of clients and slow/error I/Os that triggers the blacklist is configured according to the scale of the cluster. In order to ensure reliability, the maximum number of BlockServers in the blacklist is usually small (*e.g.*, 3 hosts). This blacklist mechanism temporarily isolates the BlockServers that provide a poor service. The system performance is not affected and engineers have sufficient time to fix the problems.

6.3 Daily Operation Scheme of Pangu

The daily operations of Pangu rely on these modules to work together. The monitoring system collects and reflects the status of Pangu. If abnormal network indicators or I/O metrics are observed, the monitoring system attempts to locate and report them to the administrators. For accidental failures such as link errors, the small QP timeout shortens the time required for failure recovery. The blacklist mechanism is able to determine and isolate nodes with poor service quality. By following these design and operator framework specifications, our RDMA-enabled Pangu system has not experienced any major faults in the last two years.

7 Experiences and Future Work

Monitoring NACK in Lossless RDMA. The operation of RDMA over a lossless fabric is difficult due to PFC risks. However, the lossless fabric increases the effectiveness of NACK events as indicators of the network error location since NACK is usually rare in a lossless fabric.

In order to detect and locate network problems, we build a subsystem based on packet loss in Pangu. In particular, Out-Of-Sequence (OOS) counters on RNICs and packet drop counters on switches are gathered. A packet loss is classified as either explicit or implicit based on whether it is recorded by switch counters. The former is easy to locate by checking the switch counters. However, determining the location of the latter is complex as RNIC counters do not distinguish between flows. By monitoring NACK in the networks, we can extract flows' five tuples and locate the root of a problem.

Building a System-Level Benchmark. To evaluate the system network performance and SLA, a representative benchmark must be constructed. Building the benchmark based on just the network metrics is simple. However, storage features such as replica completion time, slow I/O, and failure recovery should not be ignored. To measure the storage system performance and the SLA, we build a benchmark at the storage service level. The system evaluation indexes include FIO latency, IOPS, SLA with network errors, *etc.* Each upgrade in Pangu (for network and other components) is evaluated with the benchmark, allowing us to measure the overall system performance.

Congestion Control for Fail-Over Scenarios. In §3, we introduced the dual-home topology adopted in Pangu. Dual-home topology is also crucial to fail-over cases since it provides a backup path on NICs. However, we encounter a problem when deploying dual-home topology in practice.

According to our test, when one ToR switch is down, the RX pause duration can increase to 200ms per second. This is due to the transfer of the traffic from the broken ToR switch to the other switch. DCQCN handles the traffic burst poorly under this asymmetric topology. We adapt several DCQCN parameters as a temporary solution and leverage the fluid models [53] to analyze the available choices, including canceling the Fast Recovery stage and extending the rate increase duration. When removing the Fast Recovery stage in DCQCN, the pause can be eliminated yet the flow tail latency increases due to the slow recovery of the flow rate. In contrast, extending the duration of the rate-increase can result in a sharp reduction in the pause but only slightly increases the flow tail latency. In our experience, extending the rate-increase duration in DCQCN is effective for storage traffic patterns. The bandwidth drops slightly while the number of RX pauses is dramatically reduced.

This problem of DCQCN in fail-over scenarios (*e.g.*, asymmetric topology and traffic burst) indicates their important role when designing congestion control. We adopt parameter tuning to fix this problem at the price of a slight performance loss. The storage network still requires a flexible, robust and well-implemented congestion control mechanism that functions well in all scenarios. In 2019, Alibaba designed HPCC [30], a novel congestion control algorithm for the RDMA network. Adapting and integrating HPCC with the storage networks is left for future work.

Slow Processing of RDMA Read. The majority of large RPCs in Pangu are transferred via RDMA READ. We observed that when a NIC receives too much RDMA requests within a short period, it will send out many PFC frames. This is due to the slowed receiving process that results from cache misses. When a NIC is preparing for an RDMA READ, it accesses the QP context in its cache. Processing many RDMA READs consumes an excessive amount of cache resources. For slow RX rates, the NIC sends out PFC pause frames to prevent packet drops. We are currently working with the RNIC provider to solve this problem.

Lossy RDMA in Storage. Lossy RDMA is supported by Mellanox CX-5 and CX-6 RNICs. Note that CX-6 supports Selective Repeat (SR) retransmission. SR might be the ultimate step required to effectively eliminate PFC. The construction of lossy RDMA is a focal point for all RDMA-based systems. We tested lossy RDMA with Pangu over an extensive period and will deploy it for new clusters.

However, enabling the lossy feature with early generation RNICs (*e.g.*, CX-4) that have limited hardware resources and do not support SR is hard, and many production RDMA-based systems still host early generations RNICs.

NVMe-Over-Fabric. The ChunkServer data flow in Pangu is processed by CPUs. However, with NVMe-Over-Fabric, NICs can directly write the received data into NVMe SSDs. This CPU-bypass solution can save CPU costs and reduce latency. We are currently building our specialized storage protocol (and corresponding hardware) based on NVMe-over-Fabrics. A customized storage protocol for Pangu with hardware support can allow for more flexibility and control.

8 Related Work

PFC in RDMA PFC storm is the most well-known issue of RDMA. Several studies [11, 30, 52] focus on controlling network congestion to reduce the numbers of generated PFC pauses. DCQCN [52] is integrated in Mellanox RNICs. In Pangu, we tune several parameters in DCQCN to improve its performance in fail-over scenarios. However, PFC storms still occur due to hardware bugs [13]. In this paper, we present a different hardware bug that originates from switches. Existing solutions to remedy PFC storms include deadlock elimination [38] and performance optimization [46]. These solutions require switch modification. In Pangu, we combat PFC storms by switching affected links from RDMA to TCP without the need for any switch changes.

System & Network Co-Design. Recently, there have been increasing amount of work that adopts system and network co-design, including RPC systems [21, 47], distributed memory systems [39], key-value stores [22], distributed databases and transaction processing systems [6], and graph-processing systems [40]. We co-design our storage system and RDMA in Pangu. To our best knowledge, we are the first to share the experience of employing RDMA networks in large-scale distributed storage systems.

9 Conclusions

As a distributed storage system, Pangu has provided storage services to tenants both inside and outside of Alibaba for over a decade. In order to overcome the challenges of rising high-speed storage media and growing business requirements, we integrate RDMA into the storage network of Pangu, providing a common solution to different types of PFC storms. This allows for the safe deployment of RDMA. Pangu has successfully moved to a 100Gbps network by solving several new problems, such as the memory bandwidth bottleneck and QP number explosion. Furthermore, we improve the system performance of Pangu in fail-over cases.

Acknowledgment

We are really grateful to Yiying Zhang for shepherding our paper. We also thanks anonymous reviewers for their constructive suggestions. Yixiao Gao and Chen Tian are supported in part by the National Key R&D Program of China 2018YFB1003505, the National Natural Science Foundation of China under Grant Numbers 61772265, 61802172, and 62072228, the Fundamental Research Funds for the Central Universities under Grant Numbers 14380073, the Collaborative Innovation Center of Novel Software Technology and Industrialization, and the Jiangsu Innovation and Entrepreneurship (Shuangchuang) Program.

References

[1] Behnaz Arzani, Selim Ciraci, Luiz Chamon, Yibo Zhu, Hongqiang (Harry) Liu, Jitu Padhye, Boon Thau Loo, and Geoff Outhred. 007: Democratically finding the cause of packet drops. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, pages 419–435, Renton, WA, April 2018. USENIX Association.

[2] Amazon AWS. Summary of the amazon s3 service disruption in the northern virginia (us-east-1) region. s https://www.usatoday.com/story/tech/news/2017/02/28/amazons-cloud-service-goes-down-sites-scramble/98530914/, 2020.

[3] Amazon AWS. Summary of the amazon s3 service disruption in the northern virginia (us-east-1) region. s https://aws.amazon.com/cn/message/41926/, 2020.

[4] Microsoft Azure. Azure status history. s https://status.azure.com/status/history/, 2020.

[5] Buzzfeed. Buzzfeed website. s https://www.buzzfeed.com/, 2020.

[6] Yanzhe Chen, Xingda Wei, Jiaxin Shi, Rong Chen, and Haibo Chen. Fast and general distributed transactions using rdma and htm. In *Proceedings of the Eleventh European Conference on Computer Systems*, page 26. ACM, 2016.

[7] Inho Cho, Keon Jang, and Dongsu Han. Credit-scheduled delay-bounded congestion control for datacenters. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, pages 239–252. ACM, 2017.

[8] Alibaba Cloud. How does cloud empower double 11 shopping festival. s https://resource.alibabacloud.com/event/detail?id=1281, 2020.

[9] Google Cloud. Google cloud networking incident no.19005. s https://status.cloud.google.com/incident/cloud-networking/19005, 2020.

[10] Aleksandar Dragojević, Dushyanth Narayanan, Orion Hodson, and Miguel Castro. Farm: Fast remote memory. In *Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation*, pages 401–414, 2014.

[11] Yixiao Gao, Yuchen Yang, Tian Chen, Jiaqi Zheng, Bing Mao, and Guihai Chen. Dcqcn+: Taming large-scale incast congestion in rdma over ethernet networks. In *2018 IEEE 26th International Conference on Network Protocols (ICNP)*, pages 110–120. IEEE, 2018.

[12] Alibaba Group. Alibaba group website. s https://www.alibabagroup.com/en/global/home, 1999-2020.

[13] Chuanxiong Guo, Haitao Wu, Zhong Deng, Gaurav Soni, Jianxi Ye, Jitu Padhye, and Marina Lipshteyn. Rdma over commodity ethernet at scale. In *Proceedings of the 2016 conference on ACM SIGCOMM 2016 Conference*, pages 202–215. ACM, 2016.

[14] Chuanxiong Guo, Lihua Yuan, Dong Xiang, Yingnong Dang, Ray Huang, Dave Maltz, Zhaoyi Liu, Vin Wang, Bin Pang, Hua Chen, et al. Pingmesh: A large-scale system for data center network latency measurement and analysis. In *ACM SIGCOMM Computer Communication Review*, volume 45, pages 139–152. ACM, 2015.

[15] Mark Handley, Costin Raiciu, Alexandru Agache, Andrei Voinescu, Andrew W Moore, Gianni Antichi, and Marcin Wójcik. Re-architecting datacenter networks and stacks for low latency and high performance. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, pages 29–42. ACM, 2017.

[16] Jaehyun Hwang, Qizhe Cai, Ao Tang, and Rachit Agarwal. TCP = RDMA: Cpu-efficient remote storage access with i10. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*, pages 127–140, Santa Clara, CA, February 2020. USENIX Association.

[17] Alibaba Inc. Block storage performance. s https://www.alibabacloud.com/help/doc-detail/25382.html?spm=a2c5t.10695662.1996646101.searchclickresult.458e478fYtRYOO, 2018.

[18] Alibaba Inc. Pangu, the high performance distributed file system by alibaba cloud. s https://www.alibabacloud.com/blog/pangu-the-high-performance-distributed-file-system-by-alibaba-cloud_594059, 2018.

[19] Intel. Data plane development kit. s https://www.dpdk.org/, 2011.

[20] Intel. Intel memory latency checker. s https://software.intel.com/content/www/us/en/develop/articles/intelr-memory-latency-checker.html, 2020.

[21] Anuj Kalia, Michael Kaminsky, and David Andersen. Datacenter rpcs can be general and fast. In *USENIX NSDI*, pages 1–16, 2019.

[22] Anuj Kalia, Michael Kaminsky, and David G Andersen. Using rdma efficiently for key-value services. In *ACM SIGCOMM Computer Communication Review*, volume 44, pages 295–306. ACM, 2014.

[23] Anuj Kalia, Michael Kaminsky, and David G Andersen. Design guidelines for high performance rdma systems. In *2016 USENIX Annual Technical Conference*, page 437, 2016.

[24] Anuj Kalia, Michael Kaminsky, and David G Andersen. Fasst: Fast, scalable and simple distributed transactions with two-sided (rdma) datagram rpcs. In *OSDI*, volume 16, pages 185–201, 2016.

[25] Ana Klimovic, Heiner Litz, and Christos Kozyrakis. Reflex: Remote flash = local flash. In *Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '17, page 345–359, New York, NY, USA, 2017. Association for Computing Machinery.

[26] Yanfang Le, Brent Stephens, Arjun Singhvi, Aditya Akella, and Michael M Swift. Rogue: Rdma over generic unconverged ethernet. In *SoCC*, pages 225–236, 2018.

[27] David Lee, S Jamaloddin Golestani, and Mark John Karol. Prevention of deadlocks and livelocks in lossless, backpressured packet networks, February 22 2005. US Patent 6,859,435.

[28] Dan Li, Songtao Wang, Konglin Zhu, and Shutao Xia. A survey of network update in sdn. *Frontiers of Computer Science*, 11(1):4–12, 2017.

[29] Hao Li, Asim Kadav, Erik Kruus, and Cristian Ungureanu. Malt: distributed data-parallelism for existing ml applications. In *Proceedings of the Tenth European Conference on Computer Systems*, page 3. ACM, 2015.

[30] Yuliang Li, Rui Miao, Hongqiang Harry Liu, Yan Zhuang, Fei Feng, Lingbo Tang, Zheng Cao, Ming Zhang, Frank Kelly, Mohammad Alizadeh, et al. Hpcc: high precision congestion control. In *Proceedings of the ACM Special Interest Group on Data Communication*, pages 44–58. ACM, 2019.

[31] Michael Marty, Marc de Kruijf, Jacob Adriaens, Christopher Alfeld, Sean Bauer, Carlo Contavalli, Michael Dalton, Nandita Dukkipati, William C. Evans, Steve Gribble, Nicholas Kidd, Roman Kononov, Gautam Kumar, Carl Mauer, Emily Musick, Lena Olson, Erik Rubow, Michael Ryan, Kevin Springborn, Paul Turner, Valas Valancius, Xi Wang, and Amin Vahdat. Snap: A microkernel approach to host networking. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, SOSP '19, page 399–413, New York, NY, USA, 2019. Association for Computing Machinery.

[32] Mellanox. Mellanox rdma progamming manual. s https://www.mellanox.com/sites/default/files/related-docs/prod_software/RDMA_Aware_Programming_user_manual.pdf, 2015.

[33] Christopher Mitchell, Yifeng Geng, and Jinyang Li. Using one-sided rdma reads to build a fast, cpu-efficient key-value store. In *USENIX Annual Technical Conference*, pages 103–114, 2013.

[34] Netflix. Netflix website. s https://www.netflix.com/, 2020.

[35] J. Niu, J. Xu, and L. Xie. Hybrid storage systems: A survey of architectures and algorithms. *IEEE Access*, 6:13385–13406, 2018.

[36] Diego Ongaro and John Ousterhout. In search of an understandable consensus algorithm. In *2014 USENIX Annual Technical Conference (USENIX ATC 14)*, pages 305–319, Philadelphia, PA, June 2014. USENIX Association.

[37] Pinterest. Pinterest website. s https://www.pinterest.com/, 2020.

[38] Kun Qian, Wenxue Cheng, Tong Zhang, and Fengyuan Ren. Gentle flow control: avoiding deadlock in lossless networks. In *Proceedings of the ACM Special Interest Group on Data Communication*, pages 75–89. ACM, 2019.

[39] Yizhou Shan, Shin-Yeh Tsai, and Yiying Zhang. Distributed shared persistent memory. In *Proceedings of the 2017 Symposium on Cloud Computing*, pages 323–337, 2017.

[40] Jiaxin Shi, Youyang Yao, Rong Chen, Haibo Chen, and Feifei Li. Fast and concurrent rdf queries with rdma-based distributed graph exploration. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 317–332. USENIX Association, 2016.

[41] Galen M Shipman, Stephen Poole, Pavel Shamis, and Ishai Rabinovitz. X-srq-improving scalability and performance of multi-core infiniband clusters. In *European Parallel Virtual Machine/Message Passing Interface Users' Group Meeting*, pages 33–42. Springer, 2008.

[42] SPDK. Storage performance development kit. s https://www.spdk.io, 2020.

[43] Spotify. Spotify website. s https://www.spotify.com/, 2020.

[44] Brent Stephens, Alan L Cox, Ankit Singla, John Carter, Colin Dixon, and Wesley Felter. Practical dcb for improved data center networks. In *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*, pages 1824–1832. IEEE, 2014.

[45] Bingchuan Tian, Xinyi Zhang, Ennan Zhai, Hongqiang Harry Liu, Qiaobo Ye, Chunsheng Wang, Xin Wu, Zhiming Ji, Yihong Sang, Ming Zhang, Da Yu, Chen Tian, Haitao Zheng, and Ben Y. Zhao. Safely and automatically updating in-network acl configurations with intent language. In *Proceedings of the ACM Special Interest Group on Data Communication*, SIGCOMM '19, page 214–226, New York, NY, USA, 2019. Association for Computing Machinery.

[46] C. Tian, B. Li, L. Qin, J. Zheng, J. Yang, W. Wang, G. Chen, and W. Dou. P-pfc: Reducing tail latency with predictive pfc in lossless data center networks. *IEEE Transactions on Parallel and Distributed Systems*, 31(6):1447–1459, 2020.

[47] Shin-Yeh Tsai and Yiying Zhang. Lite kernel rdma support for datacenter applications. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 306–324. ACM, 2017.

[48] Xingda Wei, Jiaxin Shi, Yanzhe Chen, Rong Chen, and Haibo Chen. Fast in-memory transaction processing using rdma and htm. In *Proceedings of the 25th Symposium on Operating Systems Principles*, pages 87–104. ACM, 2015.

[49] Qiumin Xu, Huzefa Siyamwala, Mrinmoy Ghosh, Manu Awasthi, Tameesh Suri, Zvika Guz, Anahita Shayesteh, and Vijay Balakrishnan. Performance characterization of hyperscale applicationson on nvme ssds. In *Proceedings of the 2015 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS' 15, pages 473–474, New York, NY, USA, 2015. Association for Computing Machinery.

[50] Yiying Zhang and Steven Swanson. A study of application performance with non-volatile main memory. In *Symposium on Mass Storage Systems and Technologies*, 2015.

[51] Yang Zhao, Nai Xia, Chen Tian, Bo Li, Yizhou Tang, Yi Wang, Gong Zhang, Rui Li, and Alex X. Liu. Performance of container networking technologies. In *Proceedings of the Workshop on Hot Topics in Container Networking and Networked Systems*, HotConNet '17, page 1–6, New York, NY, USA, 2017. Association for Computing Machinery.

[52] Yibo Zhu, Haggai Eran, Daniel Firestone, Chuanxiong Guo, Marina Lipshteyn, Yehonatan Liron, Jitendra Padhye, Shachar Raindel, Mohamad Haj Yahia, and Ming Zhang. Congestion control for large-scale rdma deployments. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, SIGCOMM '15, page 523–536, New York, NY, USA, 2015. Association for Computing Machinery.

[53] Yibo Zhu, Monia Ghobadi, Vishal Misra, and Jitendra Padhye. Ecn or delay: Lessons learnt from analysis of dcqcn and timely. In *Proceedings of the 12th International on Conference on emerging Networking EXperiments and Technologies*, pages 313–327. ACM, 2016.