C3 : Roadmap and Evolution of High Performance and Large-scale Storage Network in Alibaba

Operational Systems Track
Paper # , 12 pages excluding reference

Abstract

Pangu storage network (PNet) is the key component of Alibaba's cloud storage system. To address the performance, availability, scalability and isolation challenges in large scale cloud storage system, PNet has gone through three stages of development: 1) **C**oordination, the enhancement stage, which successfully deployed a high-performance RDMA network for Pangu; 2) **C**oalescence, the co-design stage, which further improved the performance by offloading some bottleneck storage processing procedures to network, while improved the availability by onloading some network information to storage. 3) **C**ustomization, the in-depth customization stage, which supported cloud-native to interconnect large-scale heterogeneous storage clusters and provided multi-tenant QoS isolation through hardware-software co-design including, software-defined RDMA devices and end-to-end QoS isolation. Marching through such **C3** stages, PNet has supported Pangu to be the worldwide ZB-level cloud storage system.

1 Introduction

Alibaba Cloud is one of the top cloud providers in the world. Pangu, its unified storage platform, provides a scalable, high-performance, and reliable storage services for Alibaba's core businesses, such as Taobao, Tmall, AntFin, Alimama, and etc. On top of Pangu, many cloud storage services, such as Elastic Block Storage (EBS), Object Storage Service (OSS), Network-Attached Storage (NAS), and Tablestore, have been built for enterprises and developers from all over the world [1].

This paper describes our experience in building the Pangu storage network (PNet), the key network component of Pangu. Born with Pangu together, PNet experienced rapid developments of both the backend (e.g., the storage media), and the frontend (e.g., the cloud computing service) of this unified storage platform. These developments pose non-trivial design challenges for PNet. First, the fast increasing performance of storage media makes the network become the bottleneck with regard to the overall performance of Pangu. Specifically, the mainstream storage media has evolved from Hard Disk Drive (HDD) to Serial Advanced Technology Attachment Solid State Drive (SATA SSD), to Non-Volatile Memory Express (NVMe) SSD and Storage Class Memory (SCM), each of which has a performance substantially superior over its predecessor. For example, NVMe SSD has an accessing latency in the order of microsecond, while SCM such as the Intel Optane Persistent Memory has an accessing latency in the order of nanosecond [44]. As another example, the capacity of Intel P4510 reaches 8TB, and the recent QLC SSD reaches 16TB [19], marking an increase of nearly one order of magnitude. As a result, the performance of PNet must keep up with such ultra-low accessing latency and high data capacity of storage media, and maintain high scalability.

Second, the fast increasing demand for high-performance, high availability storage from cloud customers requires PNet to develop new availability and isolation mechanisms. In particular, more and more traditional enterprises start to migrate their IT business from their private infrastructures to the cloud. As a result, many cloud-based applications are evolving into cloud-native ones [31], which requires a unified, multi-tenant, and large-scale storage service with high availability and performance isolation.

We summarize the design challenges PNet faced in the past six years as *performance*, *availability*, *scalability*, and *isolation*. In the beginning, we addressed these challenges by upgrading the network bandwidth from 1 Gbps to 25 Gbps, and to 100 Gbps and adopting new network technologies in PNet such as Remote Direct Memory Access (RDMA) [13, 23].

However, from our experience in developing and operating PNet, one big lesson we learned, and the most important experience we want to share with the community, is that these challenges *cannot* be addressed by simply applying new network technologies.

Instead, we find out a key design principle running through the evolution of PNet: *the integration between storage and network*. In particular, PNet experiences three stages of such storage-network integration: Storage & Network Coordina-

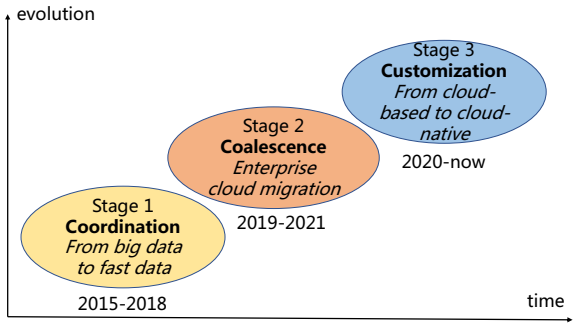


Figure 1: Evolution of PNet.

tion, Network & Storage Coalescence, and Storage-oriented Network Customization, which we refer to as **C3** in this paper, as shown in Figure 1.

Storage & Network Coordination. Since 2015, in order to make full use of novel storage media in Pangu for fast-data cloud computing, PNet introduced the RoCE protocol for the first time [23]. However, the adoption of RoCE brought challenges like the reliability of PFC (Priority Flow Control) storms, scalability of Queue Pairs (QPs, RDMA connection abstraction) and compatibility with traditional TCP-based applications [26]. As the RoCE RDMA was hard coded into the commercial NIC hardware, we could not address these challenges via the network only. Inspired by Pangu's distributed system technologies of failure tolerance and scalability, we introduced storage & network coordination which leverages the flexible software architecture to address the limitations of hardware-based network protocols. At the end of this stage, we successfully deployed hundreds of RoCE-RDMA-based SSD clusters and enabled the disaggregation of computation and storage resources in our datacenters.

Storage & Network Coalescence. When enterprises migrated their services to the cloud, it significantly affected the availability and performance of Pangu. However, our storage & network coordination was insufficient to meet such requirements due to the interaction overhead between storage and network. As such, in 2019, we summarized the major interaction taxes in Pangu and PNet, including protocol **t**ranslation, data **t**ransfer, and information **t**ransmission tax, which is referred to as **T3**. T3 led to high latency, low throughput, and low availability of Pangu, resulting in obstacles of enterprise cloud migration scenarios. One example is the cloud SAN [15], which should maintain comparable performance with traditional local SAN. Local SAN generally owns a customized network with acceleration of hardware assistance like protocol processing and dual-controller availability. However, cloud SAN uses the public network in cloud with less help from hardware. To address T3, we introduced storage & network coalescence by **o**ffloading storage protocol to network, **o**ffloading storage processing to network, and **o**nloading network information to storage, which is referred to as **O3**. With coalescence between network and storage, PNet successfully supported massive enterprise cloud migration with high availability and performance.

Storage-oriented Network Customization. At the end of 2019, Alibaba started the cloud-native campaign to migrate all its businesses to its public cloud. As such, Pangu needed to provide a unified storage service by interconnecting clusters that used to be exclusive and elegantly isolating multi-tenancy access. As a result, new issues arose for PNet. First, different network configurations were adopted by clusters of lossless-Ethernet and lossy-Ethernet, which made it hard to connect them directly. Second, traditional network isolation mechanisms could not provide end-to-end QoS isolation, because the network was not aware of important application information such as storage traffic-types. Third, hardware-lock-in network technology like Go-Back-N and congestion control was inefficient when connecting thousands of nodes.

To address these issues, we introduced storage-oriented network customization in PNet. Specifically, we separated QoS-aware data path and control path in PNet. Mechanisms such as reliability and congestion control were implemented in the software control plane rather than the hardware data plane. As such, this design adopted a unified lossy-Ethernet configuration and got rid of the PFC issues. Moreover, it adopted selective retransmission, and could even give up the retransmission in PNet when Pangu has given up the upper-layer data transmission. Based on the customized network, we provided a unified storage service for cloud-native applications with a large storage pool in 2021.

Summary This paper presents our experience in building and evolving PNet in three key stages **C3**. At each stage, we aimed to integrate network and storage together to meet the requirements and address the design challenges that arisen in Pangu. We elaborated on our key design decisions to address these challenges and present real production data to demonstrate their efficiency and efficacy. To the best of our knowledge, this is the first paper comprehensively introduces the experiences in building one of the world's largest cloud storage network. We hope our experiences can not only shed light on new industry products but also inspire new research directions.

2 Background

2.1 Pangu in Alibaba Cloud

Pangu Framework. Pangu is a distributed storage system which provides stable, scalable, and high-performance storage services. Diverse cloud storage services such as EBS, OSS, NAS, etc. are built upon Pangu and have served enterprises and developers all over the world. Now, Pangu has been the storage middle platform of Alibaba supporting all core businesses such as Taobao, Tmall, AntFin, DingDing, and Alimama, etc.

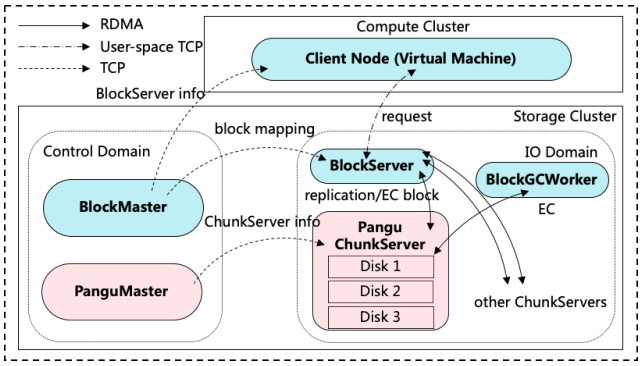


Figure 2: Block storage service framework built on Pangu

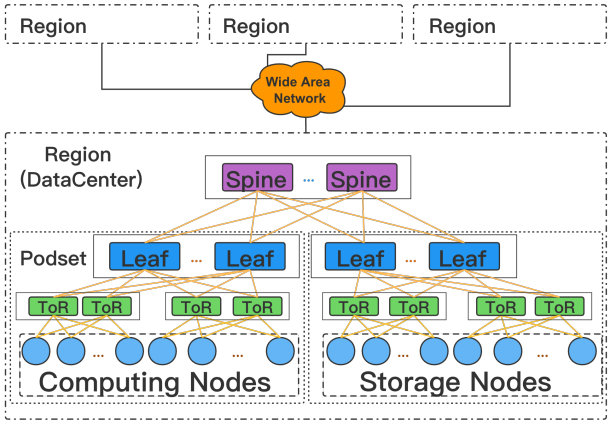


Figure 3: Topology of Pangu Network

We take EBS as an example to describe Pangu's framework. Figure 2 presents the I/O workflows of EBS and Pangu. EBS clients are deployed in the compute clusters while BlockServer and ChunkServer run on the storage clusters. Virtual block devices are deployed in compute nodes, and send read/write requests to BlockServer for I/O processing. On BlockServer, requests are resolved to chunk access to Chunkserver. Then, ChunkServer is responsible for accessing data in storage devices. It is notable that, the BlockGCWorker, there, is responsible for data compaction. Blockserver and Chunkserver are managed by BlockMaster and PanguMaster separately, which can isolate failure nodes.

Storage Protocol. Pangu storage protocol is mainly applied between Client and Chunkserver. Here, file is the logical unit, which provides persistence for the upper-layer users through Pangu Client and chunk is the logical unit, which hides the complexity of bottom storage devices in Chunkserver. Client provides per-file configurations to users. Users can adopt various options of the number of replications, priority, user-group, etc. Files are stored based on the configuration of either multi-replica or Reed-Solomon encoding for persistence. For RS(8,3) encoding, the data is split into 8 equal chunks, and 3 parity chunks which are generated from the data chunks. For replication, data chunks are the same size and multiple copies are created in Chunkserver. For fault tolerance, chunks of a file are stored in different fault domains. Besides, background services like garbage collection are running to delete unused data and repair replication when there are damaged chunks.

Cloud Demands for Pangu. Since 2016, Pangu has experienced three stages with different cloud demands. In the first stage, from 2016 to 2018, based on big data, more and more cloud customers enjoyed fast data. While big data applications emphasized data volume, fast data applications mean velocity and variety are key, which brought demands for high-performance cloud storage. In the second stage, from 2019 to 2021, while early cloud customers were internet-related enterprises, more and more traditional enterprises migrated to the cloud. These enterprises used to adopt local-room and exclusive SAN (Storage Area Network [16]) storage. When migrating to the cloud, they still wanted to maintain comparable capability with cloud storage. In the third stage, from 2020 to now, cloud-native has become popular, and more and more enterprises have enjoyed its simple deployment and elastic resources on demand. Cloud-native enjoys a unified storage service, by which storage resources are merged as a unified resource pool. As mentioned by [37], merging exclusive pools of different applications and providing a unified storage service not only help utilize storage performance and capacity but also help these applications enjoy an elastic storage service on demand. Accordingly, this brings about demands of elastic scalability and elegant isolation among applications for cloud storage.

Table 1: Configurations of 25 and 100 Gbps nodes

Hardware	25Gbps	100Gbps
CPU	Xeon 2.5GHz, 64 cores	Xeon 2.5GHz, 96 cores
Memory	DDR4-2400, 128GB	DDR4-2666, 128GB x 3
Storage	1.92TB SSD x 12	3.84TB SSD x 14
Network	ConnectX-4/5 Dual-port	ConnectX-5 Dual-port
PCIe	PCIe Gen 3.0 x 8	PCIe Gen 3.0 x 16

2.2 Pangu Storage Network

Network Topology. Alibaba Cloud operates 24 regions around the world with more global regions set to follow. Each of these is composed of hundreds of thousands of servers, and supports diverse services such as ECS, ODPS, OSS etc [3]. As shown in Figure 3, Pangu network is divided into three layers. Among different regions, data centers of Pangu are connected by Wide Area Network (WAN). In a region, there are dozens of computing and storage podsets [25]. And in a podset, Pangu adopts the Clos topology [25] in the data center. To be consistent with the common dual-home practice, we deploy Mellanox ConnectX series dual-port RNICs to connect a host with two distinct ToR switches in the data center.

Lossless and Lossy Network. The typical hardware configurations for 25Gbps and 100Gbps RNIC storage nodes are given in Table 1, and they belong to lossless and lossy network respectively. For lossless network, RoCEv2 adopts PFC on RNICs (such as ConnectX-4) and switches to prevent packet loss due to buffer overflow. For lossy network, PFC is disabled on both RNICs (such as ConnectX-5/6) and switches. As there exists packet drop, RNICs need enable mechanisms like Selective-Retransmission (SR) [36] to improve transmission efficiency.

2.3 Challenges of Storage Network

Different from most other Cloud Service Providers (CSP), the storage service of most Alibaba businesses are provided by a unified storage middle platform Pangu. To be the storage network of Pangu, PNet faces following challenges:

- **Availability**. Availability is critical for cloud storage systems. With disaggregation of computation and storage resource, a storage cluster will serve thousands of VMs (Virtual Machine). For example, a EBS cluster usually support more than 10000 VMs. When failure happens, it would lead to corrupted data or corrupted computations of many cloud customers using the VMs. In Alibaba cloud, more than 50% of incidents are related to the network, and this brings challenges for the availability of PNet.

- **Scalability**. The scalability of Pangu is important for both performance and cost. Furthermore, it facilitates data processing with all data stored in a storage cluster. For example, one ODPS cluster which is used for data processing has more than 10000 nodes in Alibaba. Meantime, cloud-native becomes popular and it need unified storage service with a large-scale cluster. The storage network should be able to scale out with the increasing number of storage servers and provide consistent network service for growing storage services.

- **Performance**. Many applications have stringent requirement on latency and throughput, such as online web search, in-memory database, streaming-processing workloads [34, 42]. Specifically, with the emergence of high-speed storage media e.g., NVMe SSDs and SCM, the access latency of storage systems has decreased in the order of microsecond and nanosecond respectively [43]. Besides, the throughput of a storage node with 12 NVMe disks is exceeded 100Gbps. Therefore, it is important for PNet to maintain high network performance.

3 Storage & Network Coordination

Pangu was born with Alibaba Cloud found in 2009. Till 2016, Pangu adopted both HDD disks and 1Gbps/10Gbps Ethernet. In 2016, the new NVMe SSD devices were introduced in Pangu and brought new performance challenges. At the same time, cloud customers started to shift from big data to fast data, which also brought stringent performance demands. Thus, there came the first-stage revolution of PNet for high-performance Pangu.

Table 2: The storage spec and performance

Storage Type	latency	Write (MB/s)	Read (MB/s)
HDD	2.9ms	32KB: 5.7 320KB:37.5 3200KB:78.3	86.5
NVMe SSD	30us	1800	2700

In 2016, NVMe SSD started to attract the attention of CSPs. As shown in Table 2 [10,17], compared to HDD, SSD latency decreased by up to 100× from the order of millisecond to the order of microsecond. And its bandwidth grew from tens of MB/s to thousands of MB/s. Furthermore, compared with HDD, SSD was more friendly for small messages read/write. The write bandwidth of HDD is 5.7MB/s for 32KB IO requests, While the write bandwidth of NVMe SSD could reach 1800MB/s, which is very important and helpful for latency-sensitive applications like DB (Database). The development of NVMe SSD needed the corresponding high-performance network for the distributed storage service.

In addition to the development of storage media, the demand of more and more cloud customers acquire for "fast data" in addition to "big data". Online data processing like Apache Flink [4] and Apache Storm [5] became popular, which needed high-performance storage access. In Alibaba, the Flink-similar applications required several-millisecond-level latency with GB/s-level bandwidth, which also brought challenges for high-performance network.

3.1 Challenges

In order to make full use of NVMe SSD and satisfy the fast-data demand, PNet introduced RDMA for the first time. There were two options: IB (InfiniBand) RDMA and RoCE RDMA. Due to the compatibility and ecology problems of IB, the RoCE was the only choice for cloud networks. However, RoCE RDMA was immature in both hardware and its mechanisms like PFC flow control. For example, we came upon with firmware bugs of QP resource leaking, where NIC must be reset to recovery. For PFC, we met the new cause of PFC memory MCE (Memory Check Error) faults, where NIC keeps sending PFC frames to the network when its pipeline is blocked as a result of MCE faults. Besides the reliability issue of RoCE RDMA, there were problems of QPs scalability [21], and performance issue of incompatible traditional TCP-based software [23]. We discuss these problems briefly in this paper.

Reliability. PFC storm is a common factor affecting the reliability of RDMA networks. PFC runs under a hop-by-hop mechanism, with the possibility of PFC storms, spreading

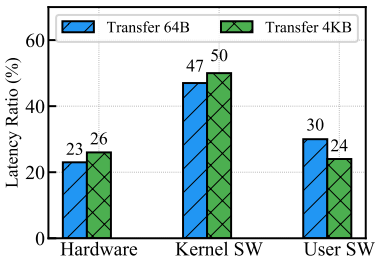


Figure 4: Communication time ratio at 25Gbps bandwidth.

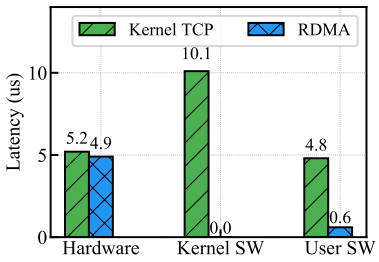


Figure 5: Latency break down between RDMA and kernel TCP over PNet for 4KB transmission.

into the whole cluster. PFC storms can severely affect cluster reliability, which is the most well-known problem in deploying RDMA [23, 25]. It was difficult to solve the problem that plagues large-scale deployment of RDMA networks by simply upgrading the capabilities of network systems.

Scalability. As mentioned in [23], Pangu adopts a full-mesh topology, and there is all-to-all communication among all machines. Thus, a large number of active QPs would be used in each machine of Pangu. In this scene, RNICs' performance would drop drastically due to frequent cache miss of QPs' contexts occurred in RNICs and restrict the scalability of RDMA networks [21, 26–28, 35, 39].

Performance. Before 2016, Pangu storage protocol stack and network protocol stack were designed for the HDD disks and kernel TCP. As latency of disks was in the order of millisecond, less attention was put on storage software, which was not well optimized. As shown in Figure 4, for transferring a message of 64B, 30% of the overhead is caused by RPC-layer, including serialization/de-serialization, and memory copy. When the message size is 4KB, the transmission overhead of RPC layer reaches 24%. When replacing TCP with RDMA, as shown in Figure 5, the kernel software overhead decreased from 10.1µs to few µs and user software overhead also decreased significantly, from 4.8µs to 0.6µs.

3.2 Methodology: Collaboration

As the stack of RoCE RDMA was lock-in with commercial NIC hardware, the above challenges could not be solved by the network only, we provided the methodology of storage & network coordination for the first time. It took advantage of Pangu's distributed system technology like failure tolerance and scalability to remove RoCE obstacles. Part of these methods were discussed in [23].

Reliability. PNet adopted mechanisms like blacklisting and RDMA/TCP switching, which are usually used by distributed storage systems, and eliminated RoCE risks like PFC storms. Taking PFC storms as an example, we found just disabling PFC in switches and RNICs was not enough. Firstly, disabling PFC leads to packets drop. As the retransmission scheme of ConnectX-4 is Go-Back-N, it is low-efficient and the whole traffic of clusters would oscillate. Meanwhile, because of the retransmission timeout, the long tail latency of dozens of milliseconds is not acceptable. Secondly, PFC may not work because PFC triggering mechanisms of RNICs and switches are not matched in some cases. We do met such cases online, where RNICs keep sending PFC frames, but it could not trigger the up-link switches to send PFC pause frames. Thirdly, the time to trigger the PFC of all network devices may take a long time, as some links between switches may have no traffic at that time, which can not trigger PFC frames. And then after a while, traffic are recovered automatically.

In order to overcome these problems, we coordinated network and storage. For severe packet drop, PNet actively switches to TCP using selective retransmission instead of Go-Back-To-N. For PFC disabling is not triggered in some cases, we adopt blacklist mechanism of Pangu to isolate the failure nodes. For long time to enable PFC in network, we adopt an active detection mechanism to check each link periodically to trigger PFC as soon as possible. Through these coordinated measures, we eliminated RoCE risks like PFC storms.

Scalability. PNet provided methods like shared link, which reduced the number of QPs by one order of magnitude, and made RoCE extend to a large scale. Pangu uses shared Link mode to solve the QPs scalability problem [23] at the application level instead of the network layer. In the shared link mode, Pangu greatly reduces the number of QPs through the shared group mode to alleviate cache miss. The shared link mode is implemented in the application layer and leaves RDMA libraries untouched. The solution implemented at the application level did not need to modify the underlying RDMA libraries, which facilitates upgrades, and does not suffer from CPU inefficiencies and low performance due to lock contention in previous solutions [28].

Performance. PNet provided novel storage software stack like user-level and R2C (Run-to-Complete), which coordinated between USSOS (User Space Storage Operating System Pangu) with RDMA verbs [23] and reduced software overhead like kernel and memory copy to utilize RDMA performance.

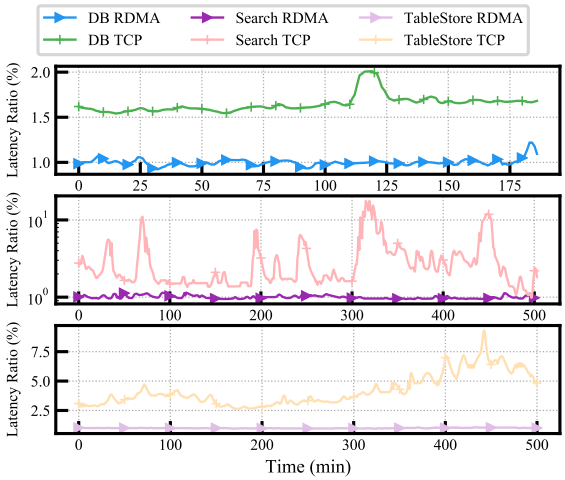


Figure 6: End-to-End Latency Comparision Between RDMA and Kernel TCP over Pangu in Database, Search and TableStore.

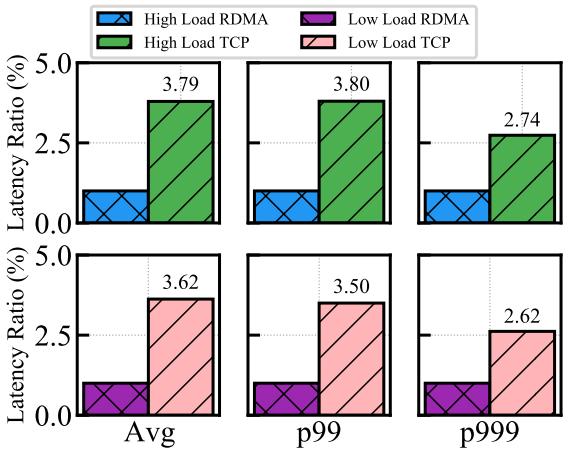


Figure 7: End-to-End Latency Comparision Between RDMA and Userspace TCP over Pangu.

3.3 Results

Figure 6 shows the end-to-end results obtained from online environment. For database applications, PNet could reduce the latency of SQL query by 42% in comparison to Kernel TCP. For search applications, PNet could not only reduce the search latency by 31% in comparison to Kernel TCP. For TableStore [18] applications, PNet achieves 65% lower latency than kernel TCP. All these results prove that RDMA does help applications in real online application in comparison to Kernel TCP.

Besides, we compare RDMA with DPDK-based Userspace TCP in EBS scene, where more than 10000 cloud block devices exists and storage cluster of 5PB capacity. Figure 7 shows the results of Userspace TCP and RoCE RDMA with high and low workload respectively. With the same configuration (CPU, memory, network), RDMA could achieve 25% more available bandwidth for users than Userspace TCP in both high and low workloads. And Userspace TCP causes 3.6x more end-to-end average latency than RDMA, 3.5x and 2.6x higher end-to-end latency in 99th and 999th percentile latency. More throughput could improve cluster utilization, while lower end-to-end latency could improve the SLA of Pangu. With less CPU usage, RDMA has a much better performance and SLA than Userspace TCP. As a result, RoCE RDMA is the optimal choice for Pangu to implement high-performance storage network.

Through the coordination between storage and network, we successfully deployed hundreds of RoCE-RDMA based SSD clusters all over the world and firstly provided cloud block service of ESSD, whose latency was 100 microseconds and reached 1 Million [2] IOPS. As a result of the improved performance of Pangu, disaggregating computation and storage became possible. For breaking the fixed binding of computation and storage resources, it made them scale separately and improved resource efficiency, which reduced the whole cost of the cloud. Nowadays, computing and storage disaggregation is popular and widely accepted in data centers [22, 24, 40, 41].

4 Network & Storage Coalescence

Base on high performance PNet of the first stage, more and more traditional enterprises migrated to the cloud. These enterprises used to adopt local-room and exclusive SAN (Storage Area Network [16]) storage. When migrating cloud, they still wanted to maintain comparable capability with cloud storage. Thus, it needs to further improve capability like availability and performance of Pangu to migrate enterprises cloud.

The rapid development of cloud computing has promoted more enterprises to deploy their business on the cloud. They required enterprise-level features such as high performance, scalability, and availability [30]. At the same time, many big data applications showed their strict restriction on latency and throughput performance, such as online analytical processing (OLAP) and online transaction processing (OLTP). For example, as enterprise-level platform deployed on the cloud, SAP HANA [14] promised to provide high-performance in-memory database. EMC [7] provided storage solutions to empower business organization and improve application performance. These developments in the new era brought new demands of extreme performance and availability for the storage network.

Enlightened by the datacenter tax profiled by Google [29], we found that the performance of Pangu suffered from the **interaction tax** between the network layer and storage layer. As illustrated in Figure 8, network layer and storage layer had their own protocols, buffers, and service information, their

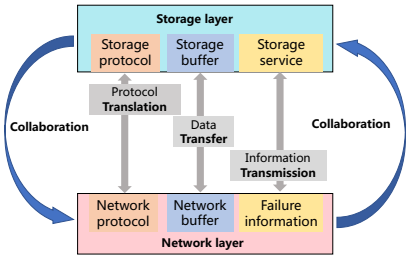


Figure 8: Interactive tax.

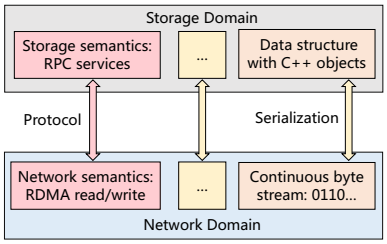


Figure 9: Protocol translation tax.

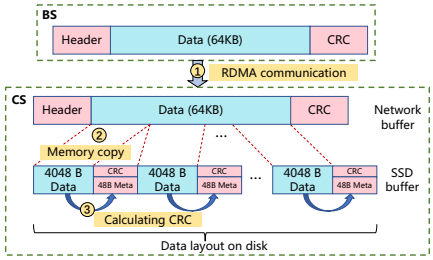


Figure 10: Data transfer tax.

collaboration caused inevitable overheads during the data translation, data transfer and information transmission.

4.1 Challenges

As we found, existing overheads of providing network service for storage were neither in the storage layer nor in the network layer but lied at the interaction tax between two layers. As illustrated in Figure 8, we take the three main interaction tax as an example: protocol translation tax, data transfer tax, and information transmission tax. For short, we named the interaction tax as **T3**.

Protocol translation tax. Protocol translation tax was referred to the translation from storage protocol to network protocol, RPC took the main responsibility for this tax. As shown in Figure 9, for a RPC initiated from storage domain (such as Blockserver) to network domain (PNet), the discrete objects in the storage layer would be firstly serialized into a continuous byte stream for inter-node network communication. The byte stream had to be deserialized into structured data at the Chunkserver side. The mismatch between storage protocol and network protocol format required redundant serializaiton/deserialization operated through Protobuf. As enterprise-level applications migrated to the cloud, the latency brought by RPC could not be ignored. As a result, we had to design a new method to cope with the heavy overhead produced by RPC.

Data transfer tax. We leveraged data transfer tax to denote the overhead caused by data movement between network and storage in Pangu. A typical Pangu Write operation is illustrated in Figure 10. Firstly, the data from Blockserver was transferred at continuous network memory buffer through RDMA communication in PNet, and then it was sliced and copied to the SSD memory buffer according to Pangu's data format. Finally, the CRC of each piece of data was calculated and written on disk together with those data. Transforming discrete memory buffers and meta information into continuous buffer consumed excessive memory bandwidth and CPU. The redundant data movement inevitably occupied memory bandwidth and caused data transfer tax, and data transfer tax became heavy under the circumstance of write amplification.

As illustrated in Figure 11, when storing data on Pangu, the data was firstly written in three replicas for fault tolerance, then Pangu leveraged a background process to read a replica and conducted EC algorithm for the purpose of economizing storage space. EC(8, 3) was used in our EC algorithm, and the stored data would be operated $3 + 1 + 1.375 = 5.375$ times in the whole Write process and caused write amplification. Under this circumstance, the effective network bandwidth leveraged by users was only 1.16GBps in a 50Gbps network, $1.16\text{GB} \times 8 \times 5.375 \approx 25\text{Gbps}$. However, SSD bandwidth increased with the higher capacity of the disk, such as Intel P4510 SSD even achieves 3GBps per SSD [12]. As a result, network bandwidth became a bottleneck as the capacity of storage node increased, the sale model changed from capacity-based sales to performance-based sales.

We introduced 100Gbps lossy RDMA network to increase throughput of Pangu, while it also brought new challenges to PNet. In our practice, the memory bandwidth reached its bottleneck before network achieved 100Gbps, the memory pressure adversely affected the network and resulted in heavy packet loss. The network failed to make full use of its advantages.

Information transmission tax. We used information transmission tax to denote the long failure notification time between network and storage. For example, the heartbeat mechanism in distributed system was always time-consuming in detecting failure node due to its non-deterministic decisions and the long information transmission time. The reconnect mechanism with traditional TCP took several minutes to avoid network failure node. However, availability was an essential challenge for a storage system to build stable cloud storage service. If the SLA we promised was 99.975% availability in a month. In other words, the maximum unavailable time of our cloud storage service could not exceed 4.3 minutes in a month, and violating these SLA causes tenants dissatisfaction and CSP reputation/monetary loss. Similar promises had been made by other CSPs [6, 8]. The long information notification time from network to storage hindered the fast recovery of Pangu's storage service. High availability required seconds level fault discovery and recovery which usually took almost several minutes.

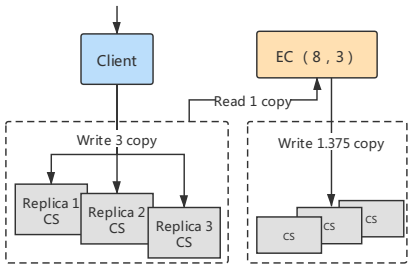


Figure 11: Data written process in Pangu.

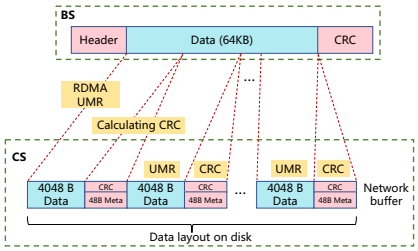


Figure 12: Offloading storage process to network.

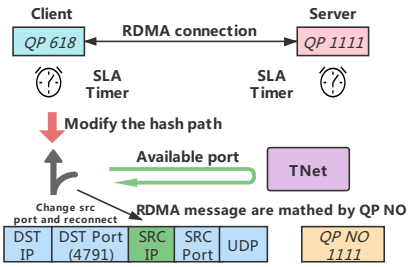


Figure 13: Dynamical source port change.

4.2 Methodology: Coalescence

Based on the analysis above, the overheads of performance and SLA occurred in the interaction phase between network storage and tax has to be paid. With the idea of "coalescence", we further integrated network and storage to eliminate these interaction tax, and **O3** was proposed: offloading storage protocol to network, offloading storage process to network, and onloading network information to storage.

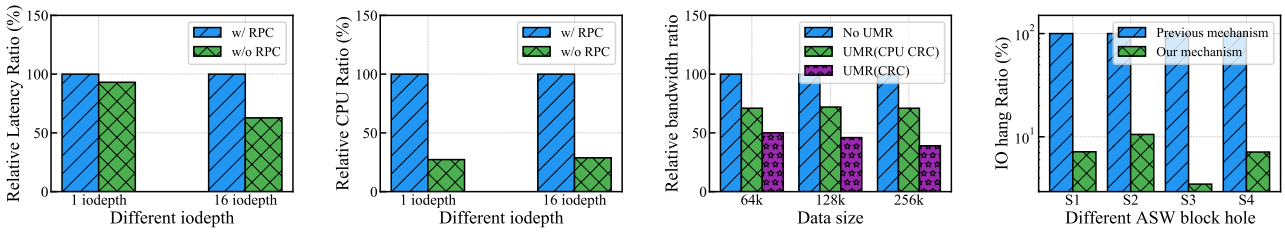
Offloading storage protocol to network. In Pangu, all data communications were in the form of RPC and there were about one hundred RPC services, and Pangu Read/Write operations occupied the most network traffic among these RPC services. Thus, we eliminated the protocol translation tax by targeting at the Read/Write RPC operations. In particular, we designed a customized data structure while maintained the flexibility and performance to replace the protobuf during Read/Write RPC. At the same time, we customized a fast path for storage Read/Write operation by offloading storage protocol to network, which took the advantage of lightweight data header of Read/Write data structure. As a result, the complicated and expensive protocol translation tax was eliminated.

In order to verify the effectiveness of reducing protocol translation tax by offloading storage protocol to the network, we let the computing node run PNet Benchmark with a working thread, 1 and 16 depth (in-flight I/O requests), and 4KB block size sequential write on cloud disks. We enabled our approaches and compared the latency, effective CPU utilization with the previous method (namely with RPC). Figure 14(a) and Figure 14(b) depict the relative CPU and latency ratio with different IO depths. The relative CPU utilization ratio could be reduced by 72.73% with 1 I/O depth and 71.42% with 16 I/O depth. The latency could be reduced by 6.92% and 37.25% with 1 I/O depth and 16 I/O depth, respectively. The results exhibited the outstanding advantages of eliminating RPC on critical data path.

Offloading storage process to network. Data transfer tax brought great pressure on memory bandwidth when processing storage data. The throughput of Pangu could be improved by offloading storage process of Pangu to network. In particular, the RDMA User-Mode-Registration (UMR) was leveraged to inline data format construction for storage layer. Different from data movement procedure in the collaboration stage, as shown in Figure 12, UMR was leveraged to slice the data and perform CRC calculation during the network communication phase. After completion of network communication, ChunkServer could directly receive the continuous data that meets the required storage format of Pangu and directly write the data to disk. The overhead caused by memory copy was removed and the CRC calculation was offloaded to the communication phase, which also reduces CPU utilization and memory bandwidth consumption. As a result, back-pressure caused by memory bottleneck would be reduced and the network would experience less impact.

We sampled the memory bandwidth to verify the effects of offloading storage protocol process to the network on reducing data transfer tax. Figure 14(c) illustrates the relative proportion of memory bandwidth reduction ratio when the written data size was 64K, 128K, and 256k. Taking the original results (no UMR) as baseline, the memory bandwidth was also sampled under two other configurations : using UMR while calculating CRC by CPU, using UMR and calculating CRC by UMR. The results show that memory bandwidth can be reduced by nearly 30% when UMR only sliced data, and nearly 50% if UMR further calculated CRC of the sliced data when the data block is 64K. Moreover, the advantage became more obvious under larger size of data block. Finally, the data movement tax had been effectively reduced by offloading the storage process to network and available throughput sold for users was nearly doubled.

Onloading network information to storage. To reduce the information transmission tax, we leveraged the storage service to solve the network failure information to improve availability of Pangu. Specially, we proposed the method of dynamically changing source port to enable a fine-grained and fast failure information process. As illustrated in Figure 13, for an RDMA connection, we maintained a customized SLA timer on both side of client and server side, the precision of which is in millisecond-level. If the timer exceeded the predefined time threshold due to network failure, we would modify the source port of the RDMA connection quickly



(a) Latency comparison of making storage data path in network.

(b) CPU comparison of making storage data path in network.

(c) Relative bandwidth ratio by offloading storage protocol.

(d) Number of IO hang with different situation of ASW block hole.

Figure 14: The effects of O3 by coalescence.

to change the routing path evade the problematic node in network. In terms of black hole in network, we would directly change the source port to bypass the black hole. As for the flapping node in network, we blacklisted the storage nodes to bypass the flapping nodes.

The effect of reducing information transmission tax by onloading network information to storage was also tested. In production, the proportion of slow IO caused by flapping was reduced by an order of magnitude. We evaluated the proposed method by counting the number of slow IO (The IO requests that failed to be responded within 1 second) in different situations of black holes. As shown in Figure 14(d), dynamical source port changing mechanism reduced the number of slow IO caused by different situations of black holes by an order of magnitude on the whole. As a result, we eliminated the information transmission tax by onloading network information to storage.

4.3 Summary

In this stage, the overheads caused by the collaboration between storage and network were analyzed and they hindered the demands of enterprise-level applications. Specially we defined T3 and corresponding solutions O3 by coalescence between storage and network. As we had verified, O3 demonstrated its ability in reducing T3 tax. As a matter of fact, we had applied the aforementioned methods in our production environment and remarkable effects had been achieved. The great success of coalescence empowered Pangu to satisfy the growing enterprise-level applications and was implemented as our second stage PNet.

5 Network & Storage Customization

5.1 Calling for Multi-tenant Pangu Storage

External factors. The concept of cloud-native had become the most popular theme in the cloud and allowed applications to enjoy simple deployment and elastic resources on demand. Since enterprises migrated their business on the cloud, we had observed that more and more enterprises and developers began to transform their traditional applications into cloud-native applications, including BigData, Middleware, Database, Security, Networking, Storage, and Bare-metal Servers. In cloud-native, storage resources are abstracted into **one unified service** and need interconnect heterogeneous clusters, support heterogeneous workloads on the cloud. In industry, Facebook unified kinds of open-source specialized storage systems into one customized Tectonic filesystem for improving resource utilization [37]. Google proposed its own NICs to support multi-tenant datacenters wherein uncoordinated large-scale distributed applications share the common infrastructures [38].

Internal factors. Pangu had served diverse internal and external business systems on Alibaba Cloud. For each application, exclusive Pangu clusters were deployed and served for the application to guarantee the isolation of stability, performance, and SLA. However, separating applications in different clusters resulted in poor resource utilization and high cost. Cloud storage also needs merge massive storage clusters into an abstract storage pool and exploited the spare resource to support multi-tenancy.

5.2 Challenges

5.2.1 Cluster Island

To allow applications share Pangu clusters and smoothly migrate among heterogeneous clusters, PNet should be able to connect them and provide a flatten network domain. Unfortunately, in Pangu clusters, RNICs are used widely and consist of different generations, vendors and hard to interconnect with each other, resulting in cluster island. Currently, Pangu experienced three main problems.

Configuration of lossy & lossless network conflicts. In Pangu clusters, both lossy and lossless network configuration were deployed. Lossless network required that PFC should be enabled both on end-hosts and switches, while lossy network did not rely on it. Though lossless and lossy network

both need to support ECN function in end-hosts and switches, switches with shallow buffer could only provide 2 hardware queues for RDMA traffic (similiar to [25]) and different ECN thresholds for lossy and lossless conflicted. For example, the typical ECN_{max} for lossless network was 300KB while for lossy network was 125KB in 25Gbps network. Interconnecting these clusters could lead to conflict configurations.

Hardware differences in RNICs lead to inconsistent service. RNICs manufactured by different generations were hard to interconnect with each other. For example, ConnectX-4 RNICs were deployed in early servers and ran over a lossless network while ConnectX-5/6 RNICs ran over a lossy network for eliminating the risk of PFC by new features, such as Select Repeat (SR), slow start, adaptive retransmission and etc. To connect these RNICs in different clusters, we had to downgrade ConnectX-5/6 RNICs to lossless mode while still suffered from the risk of PFC, which was unacceptable both in costs and development of technology. Moreover, hybrid deployment of RNICs would lead to inconsistent service. We observed great bandwidth unfairness among ConnectX-4 RNICs and ConnectX-5 RNICs in our test environments where ConnectX-4/5 RNICs were configured as lossless mode and located in the same cluster. In future, Intel RNICs (such as E810), Infrastructure Process Unit (IPU) (such as Mount Evans) [11], and RNICs manufactured by Alibaba ourselves also support RDMA transport on-chip. These RNICs may contribute to the heterogeneous network configuration in the near future.

Hardware lock-in leads to inflexible implementations. Control parts of RDMA RC transport was fully implemented in hardware and inflexible to select different implementations under specific circumstances. Control parts included congestion control and reliability which could be optimized for different applications and pure parameters tuning was not enough. For example, switches in the data center usually consisted of independent vendors and the function set supported was different. Our historically deployed switches did not support ECN function, making ECN-based congestion controls unavailable. Congestion controls for optimizing single flow were not suitable for Coflows [20] and solutions utilizing precise network telemetry had been proved to be effective than them using passive feedback [33]. Go-Back-N had been proved inefficient under low packet loss ratio [25]. Retransmission in RC was inflexible to implement application-aware strategies such as dynamically changing timeout, retry count, or actively giving up without disconnecting QPs. In a heterogeneous environment, a one-size-fits-all hardware solution was not optimal. *These three problems made it impossible to connect heterogeneous clusters and build a unified storage service from Pangu.*

5.2.2 Unguaranteed End-to-End Isolation for Multi-tenancy Storage

To accommodate diverse tenants on unified storage service, guaranteeing performance isolation was the critical capability of Pangu. In Pangu, each request from tenants would experience three processing stages before being written to disks: 1. client library of Pangu embedded into applications (EBS, ElasticSearch, etc) 2. host NICs and switches 3. storage nodes (*ChunkServer*) of Pangu.

Software-based QoS is not efficient. In Pangu, User-space storage and network stack were leveraged to achieve extreme low latency and high throughput, lacking necessary QoS mechanisms. Software-based QoS mechanisms were easy to implement but experienced three problems. One problem was that these mechanisms consumed CPU resources which was expensive, such as frequent information collection and bandwidth schedule over network communication. Another problem was that software-based QoS could not meet the demand of microsecond-level latency and could only provide bandwidth control in coarse-grained styles. In addition, software-based QoS introduced extra and unavoidable latency jitters, resulting in worse QoS.

End-to-End isolation involves storage and network. For storage disk, interference between Read and Write operations led to unguaranteed QoS [32]. [1] In Pangu, an individual thread monopolizes a single disk in a User-space manner, making the CPU of the thread a contention resource. For network, diverse traffics with different requirements coexisted in Pangu, including online data traffic from tenants, offline data traffic from garbage collection, monitor traffic from periodical heartbeats, and admin traffic from cluster operations. Unfortunately, the number of hardware queues in RNICs and switches were limited. As a result, isolation for multi-tenancy must be enforced both in storage and network.

5.3 Methodology: Customization

To achieve the interconnection among heterogeneous clusters and guarantee QoS among multiple traffics, we proposed a new model for storage network and reconstructed the framework of the end-host network. Figure 15 describes the new model of the storage network.

Releasing reliability from hardware. To implement application-aware reliability, PNet leveraged UC transport of RDMA [9] and built the reliability in software, as shown in Reliability part of Figure 15. By using UC, PNet reserved the core capability of RDMA, including offloading protocol pack/unpack, large data transmission, and zero copy. Implementing reliability in software usually consumes extra CPU resources and latency. PNet split large data block into multi-

[1] Due to the Flash Translation Layer (FTL) of SSD, Write operations will lead to Garbage Collection (GC) in SSD and generate inner amplification, interfering the performance of Read operations

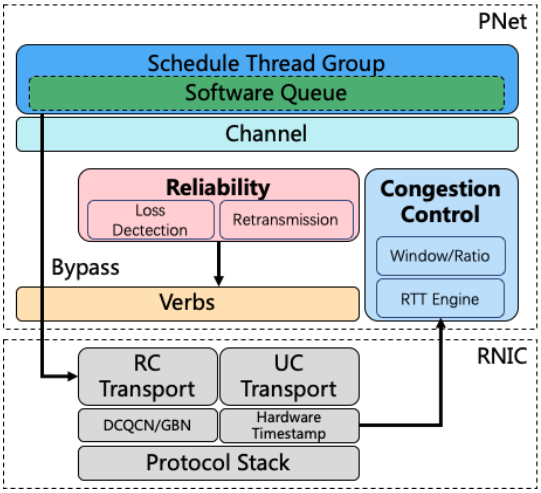


Figure 15: New network model of PNet.

ple data chunk and adaptively adjusted the size of chunk to balance CPU consumption and incurred latency. Based on chunks, PNet implemented the numbering and acknowledgement mechanisms to detect data loss and executes application-aware SR for them. Finally, PNet could guarantee the reliability of data transmission and break the hardware lock-in.

Releasing congestion control from hardware. To adapt to diverse network environments, PNet leveraged Round-Trip-Time (RTT) as the signal of network congestion to interconnect heterogeneous clusters shown in Figure 15. Each host measured the RTT of packets individually. Switches simply forwarded packet normally and participated in the congestion control implicitly. As a result, congestion control could be implemented on end-hosts while zero-touching network switches, significantly alleviating the deployment complication of the storage network. To achieve precise RTT measurement on end-hosts, PNet and RNICs need to cooperate to trace inflight packets and their acknowledgments. As shown in the Hardware Timestamp part of Figure 15, tracepoints were recorded for each WQE in both end-hosts. By these tracepoints, PNet could compute the precise value of end-to-end RTT, distinguish and eliminate extra processing latency from end-hosts. Finally, congestion control of RDMA could be fully developed and run in application level without hardware lock-in. New congestion control algorithms could be implemented in hundreds of code lines and tested in 1 day. For intra-pod communication, PNet could run over RC and maximize the advantage of hardware offloading.

Interconnecting heterogeneous clusters. By onloading the reliability and congestion control to software, PNet could run in a cluster configured with mixed RNICs and allow ConnectX-4 RNICs to run over a lossy network. Figure 16 shows the result of PNet in a hybrid deployment of ConnectX-4/5 RNICs at the same cluster. PFC and DCQCN were both disabled on end-hosts. We took the ib perftest tool

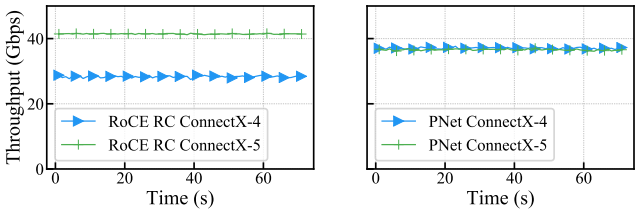


Figure 16: Hybrid tests using PNet and RoCE RC.

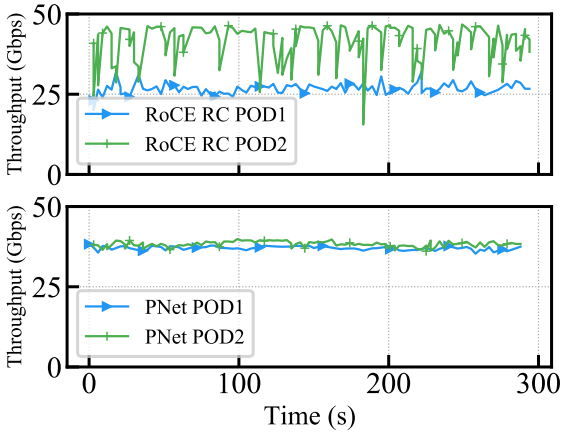


Figure 17: Cross-pods test of PNet and RoCE RC.

(*ib_write_bw*) as the RoCE RC and compared with PNet. The results of RoCE RC showed that machines with ConnectX-5 RNICs achieved throughput at about 41Gbps, 10Gbps higher bandwidth than machines with ConnectX-4 RNICs. In comparison with RoCE RC, PNet in ConnectX-4 machines and ConnectX-5 machines achieved about 38Gbps throughput and could remove the bandwidth gap that existed in RoCE RC. The reason was the impact on congestion control from different implementations of ConnectX-4/5 RNICs, such as QP allocation and scheduling. Different hardware implementation of ConnectX-4/5 RNICs did not impact PNet, because the congestion control and reliability were hardware-independent. In addition, we executed another test to verify the advantage of PNet on two clusters located in two pods. In each cluster, only ConnectX-4 or ConnectX-5 RNICs were deployed and lossless network was configured (intra-pod with PFC, inter-pod without PFC) for RoCE RC. Similar test configurations were used and Figure 17 shows the result. The throughput of RoCE RC in one pod shrunk fiercely from 25Gbps to 45Gbps, while the throughput of the other pod shrunk from 25Gbps to 20Gbps. The unstable and unbalanced throughput was caused by packet loss in inter-pods. A client suffered more throughput loss when it communicated with more servers on the other pod. As a comparison, the throughput of PNet was steady and balanced between two pods due to consistent congestion control and adaptive chunking and retransmission.

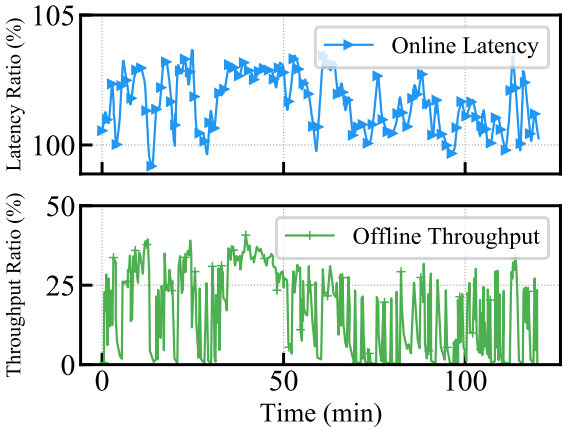


Figure 18: Hybrid of Online Search with Offline AI Training over Same Pangu Clusters.

Enforcing end-to-end QoS. To isolate different traffic from users, Pangu need a systematic mechanism to schedule traffics along three stages. For reducing overhead from software and interference from hardware, Pangu did fewer things as much as possible in software, utilized the scheduling capability of the network, and isolated requests by disks.

1). Traffic Class. Each request was set a specific traffic class *C* by applications before arriving to library of Pangu at client.

2). Class Group & Software Queue. Threads existed in the client library of Pangu and were grouped by class *C*. Requests with class *C* were en-queued into corresponding software queues and scheduled by the corresponding thread group shown in Figure 15.

3). Hardware Scheduling. Before forwarding requests to *ChunkServer*, requests would be en-queued into hardware schedule queues on host NICs, scheduled by configured policies, and tagged with different DSCP values. Three DSCP services were provided, Gold for monitor and admin traffic, Sliver for online traffic, and Copper for offline traffic. Switches directed traffic into different queues based on DSCP in packets and scheduled traffic in queues by configured policies.

4). Interference Control. Each thread in *ChunkServer* served for one specific class *C* and limited Write traffic by granted fewer tokens to the client when Read latency increases. After receiving requests from the client library of Pangu, threads executed disk operations on isolated disks. Interference in CPU and disk could be removed. For responses, a similar process was executed on the reverse path. Finally, requests with different types could be processed separately by consistent rules and interference from software and hardware could be reduced as much as possible.

Figure 18 shows the hybrid result of online search with offline AI training jobs. The baseline of online search latency was sampled without offline jobs. When the throughput of offline AI training jobs reached almost 39%, the latency of online search only fluctuated by 5%. As a result, latency-sensitive applications could deploy and run with throughput-sensitive applications without headache impact.

5.4 Summary

In this stage, Pangu customizes RDMA functionality to interconnect heterogeneous RNICs and solve configuration conflicts among existed lossless network and imported lossy network. Based on a flatten storage network, Pangu customizes the storage and network processing to guarantee end-to-end isolation for multi-tenants, resulting in better resource utilization.

6 Conclusion

Pangu is the unified cloud storage system in Alibaba and has supported its core businesses like E-commerce and cloud computing for 6 years. Now, it has deployed in hundreds of clusters, which composed of hundreds of thousands of nodes. To meet scalability, performance and availability of Pangu, it requires sophisticated design and dedicated implementation effort of storage network. PNet (Pangu Storage Network) has gone through several stages and evolved to large-scale, high-performance and reliable network. We report the evolution roadmap of PNet, and explain the motivation why we need to develop a more advanced network for each stage. We then demonstrate the methodology to overcome the tough challenges of these stages, and the results how PNet helps Pangu to improve its competitiveness.

When operating PNet, storage and network are two individual systems in cloud. The technology development of two systems are not always matched. For example, the network of lossy-RDMA without PFC was not mature, while Pangu needs inter-pod RDMA to support businesses like TableStore, which cannot support cross-pod RDMA network. In aspect of deployment, the 100Gbps Ethernet network was not ready in 2018, while Pangu clusters met throughput bottleneck, resulting in so many 25Gbps racks existed in data center. When the network virtualization was ready, but storage was still made use of underlay network for bare-metal performance. The out of sync between two systems make it hard to land new technology in time. In production, new technologies must prove it value before being online. However, the profit need the participant of storage team. It need more synchronization between the technology demand and development. Last but not least, the deployment of technology should be planed ahead.

References

[1] Alibaba cloud. Alibaba Cloud: Cloud Computing Services. https://www.alibabacloud.com/.

[2] Alibaba inc. Block storage performance. https://www.alibabacloud.com/help/doc-detail/25382.html, 2021.

[3] Alibaba infrastructure. Alibaba Cloud's Global Infrastructure. https://www.alibabacloud.com/global-locations, 2021.

[4] Apache flink. Apache Flink: Stateful Computations over Data Streams. https://flink.apache.org/.

[5] Apache storm. https://storm.apache.org/.

[6] Aws s3 sla. AWS S3 Service Level Agreement. https://aws.amazon.com/cn/s3/sla/, 2021.

[7] Dell emc. Harness the power of storage solutions. https://www.delltechnologies.com/en-us/storage/data-storage.html, 2021.

[8] Huawei sla. Huawei Service Level Agreement. https://www.huaweicloud.com/declaration/sla.html, 2021.

[9] Ibspec. IBSpec. https://www.afs.enea.it/asantoro/V1r1_2_1.Release_12062007.pdf, 2021.

[10] Intel. Intel SSD DC P3500 Series 2.0TB 2.5in PCIe 3.0 20nm MLC Product Specifications. https://ark.intel.com/content/www/us/en/ark/products/79633/intel-ssd-dc-p3500-series-2-0tb-2-5in-pcie-3-0-20nm-mlc.html.

[11] Intel ipu. Infrastructure Processing Units (IPUs) and SmartNICs. https://www.intel.com/content/www/us/en/products/network-io/smartnic.html#:~:text=Infrastructure%20Processing%20Units%20%28IPUs%29%20from%20Intel%20Programmable%20network,and%20storage%20infrastructure%20functions%20in%20a%20data%20center., 2021.

[12] Intel ssd. Intel Memory and Storage. https://www.intel.com/content/www/us/en/products/details/memory-storage.html, 2021.

[13] Rdmaprogrammingmanaual. RdmaMan. https://www.mellanox.com/related-docs/prod_software/RDMA_Aware_Programming_user_manual.pdf, 2021.

[14] Sap business technology platform. SAP HANA. https://www.sap.com/products/hana/what-is-sap-hana.html, 2021.

[15] Snia. What Is a Storage Area Network (SAN)? https://www.snia.org/education/storage_networking_primer/san/what_san.

[16] Snia. Storage Area Network (SAN). https://www.snia.org/education/online-dictionary/term/storage-area-network.

[17] The storage spec and performance. https://github.com/4paradigm/pafka, 2021.

[18] Tablestore. AliyunTableStore. https://help.aliyun.com/document_detail/27280.html, 2021.

[19] Visiontek 16tb class qlc 7mm 2.5" ssd. https://www.dell.com/en-us/shop/visiontek-16tb-class-qlc-7mm-25-ssd/apd/ab329068/storage-drives-media, 2021.

[20] Mosharaf Chowdhury, Yuan Zhong, and Ion Stoica. Efficient coflow scheduling with varys. In *Proceedings of the 2014 ACM conference on SIGCOMM*, pages 443–454, 2014.

[21] Aleksandar Dragojević, Dushyanth Narayanan, Miguel Castro, and Orion Hodson. Farm: Fast remote memory. In *11th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 14)*, pages 401–414, 2014.

[22] Peter X Gao, Akshay Narayan, Sagar Karandikar, Joao Carreira, Sangjin Han, Rachit Agarwal, Sylvia Ratnasamy, and Scott Shenker. Network requirements for resource disaggregation. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pages 249–264, 2016.

[23] Yixiao Gao, Qiang Li, Lingbo Tang, Yongqing Xi, Pengcheng Zhang, Wenwen Peng, Bo Li, Yaohui Wu, Shaozong Liu, Lei Yan, et al. When cloud storage meets {RDMA}. In *18th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 21)*, pages 519–533, 2021.

[24] Juncheng Gu, Youngmoon Lee, Yiwen Zhang, Mosharaf Chowdhury, and Kang G Shin. Efficient memory disaggregation with infiniswap. In *14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17)*, pages 649–667, 2017.

[25] 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 ACM SIGCOMM Conference*, pages 202–215, 2016.

[26] Anuj Kalia, Michael Kaminsky, and David Andersen. Datacenter rpcs can be general and fast. In *16th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 19)*, pages 1–16, 2019.

[27] Anuj Kalia, Michael Kaminsky, and David G Andersen. Design guidelines for high performance {RDMA} systems. In *2016 USENIX Annual Technical Conference (USENIX ATC 16)*, pages 437–450, 2016.

[28] Anuj Kalia, Michael Kaminsky, and David G Andersen. Fasst: Fast, scalable and simple distributed transactions with two-sided ({RDMA}) datagram rpcs. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 185–201, 2016.

[29] Svilen Kanev, Juan Pablo Darago, Kim Hazelwood, Parthasarathy Ranganathan, Tipp Moseley, Gu-Yeon Wei, and David Brooks. Profiling a warehouse-scale computer. In *Proceedings of the 42nd Annual International Symposium on Computer Architecture*, pages 158–169, 2015.

[30] Andrzej Kochut, Yu Deng, Michael R Head, Jonathan Munson, Anca Sailer, Hidayatullah Shaikh, Chunqiang Tang, Alexander Amies, Murray Beaton, David Geiss, et al. Evolution of the ibm cloud: Enabling an enterprise cloud services ecosystem. *IBM Journal of Research and Development*, 55(6):7–1, 2011.

[31] Nane Kratzke and Peter-Christian Quint. Understanding cloud-native applications after 10 years of cloud computing-a systematic mapping study. *Journal of Systems and Software*, 126:1–16, 2017.

[32] Minkyeong Lee, Dong Hyun Kang, Minho Lee, and Young Ik Eom. Improving read performance by isolating multiple queues in nvme ssds. In *Proceedings of the 11th International Conference on Ubiquitous Information Management and Communication*, pages 1–6, 2017.

[33] 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. 2019.

[34] Luo Mai, Kai Zeng, Rahul Potharaju, Le Xu, Steve Suh, Shivaram Venkataraman, Paolo Costa, Terry Kim, Saravanan Muthukrishnan, Vamsi Kuppa, et al. Chi: A scalable and programmable control plane for distributed stream processing systems. *Proceedings of the VLDB Endowment*, 11(10):1303–1316, 2018.

[35] Christopher Mitchell, Yifeng Geng, and Jinyang Li. Using one-sided {RDMA} reads to build a fast, cpu-efficient key-value store. In *2013 {USENIX} Annual Technical Conference (USENIX ATC 13)*, pages 103–114, 2013.

[36] Radhika Mittal, Alexander Shpiner, Aurojit Panda, Eitan Zahavi, Arvind Krishnamurthy, Sylvia Ratnasamy, and Scott Shenker. Revisiting network support for rdma. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, pages 313–326, 2018.

[37] Satadru Pan, Theano Stavrinos, Yunqiao Zhang, Atul Sikaria, Pavel Zakharov, Abhinav Sharma, Mike Shuey, Richard Wareing, Monika Gangapuram, Guanglei Cao, et al. Facebook's tectonic filesystem: Efficiency from exascale. In *19th {USENIX} Conference on File and Storage Technologies ({FAST} 21)*, pages 217–231, 2021.

[38] Arjun Singhvi, Aditya Akella, Dan Gibson, Thomas F Wenisch, Monica Wong-Chan, Sean Clark, Milo MK Martin, Moray McLaren, Prashant Chandra, Rob Cauble, et al. 1rma: Re-envisioning remote memory access for multi-tenant datacenters. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, pages 708–721, 2020.

[39] Maomeng Su, Mingxing Zhang, Kang Chen, Yongwei Wu, and Guoliang Li. Rfp: A remote fetching paradigm for rdma-accelerated systems. *arXiv preprint arXiv:1512.07805*, 2015.

[40] Jason Taylor. Facebook's data center infrastructure: Open compute, disaggregated rack, and beyond. In *Optical Fiber Communication Conference*, pages W1D–5. Optical Society of America, 2015.

[41] Midhul Vuppalapati, Justin Miron, Rachit Agarwal, Dan Truong, Ashish Motivala, and Thierry Cruanes. Building an elastic query engine on disaggregated storage. In *17th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 20)*, pages 449–462, 2020.

[42] Le Xu, Shivaram Venkataraman, Indranil Gupta, Luo Mai, and Rahul Potharaju. Move fast and meet deadlines: Fine-grained real-time stream processing with cameo. In *18th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 21)*, pages 389–405, 2021.

[43] 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*, pages 473–474, 2015.

[44] Jian Yang, Juno Kim, Morteza Hoseinzadeh, Joseph Izraelevitz, and Steve Swanson. An empirical guide to the behavior and use of scalable persistent memory. In *18th {USENIX} Conference on File and Storage Technologies ({FAST} 20)*, pages 169–182, 2020.