# 关于远程直接数据存取（RDMA）技术的综述

孙锐

)(华中科技大学 计算机科学与技术学院, 武汉市 中国 430074)

**摘 要** RDMA 技术全称远程直接数据存取，就是为了解决网络传输中服务器端数据处理的延迟而产生的。RDMA 通过网络把资料直接传入计算机的存储区，将数据从一个系统快速移动到远程系统存储器中，而不对操作系统造成任何影响，这样就不需要用到多少计算机的处理功能。它消除了外部存储器复制和上下文切换的开销，因而能解放内存带宽和 CPU 周期用于改进应用系统性能。由于 RDMA 网络的高吞吐量、低延迟、CPU 效率和远程内存操作等高级特性，支持 RDMA 的网络在数据中心部署中越来越青睐。最先进的远程直接内存访问(RDMA)技术，如无限带(IB)或 RDMA 聚合以太网(RoCE)，正广泛应用于数据中心应用，并在云环境中获得关注。但是 ReDMArk 表明，当前基于 ib 的架构的安全机制对于网络内攻击者和位于终端主机上的攻击者都是不够的，因此不仅影响了 RDMA 应用程序的保密性，而且也影响了 RDMA 应用程序的完整性。因此，RDMA 架构的安全性也是至关重要的。此外，生产级云存储系统必须性能好且易于使用。存储媒体的快速发展使得网络落后，导致了新的云存储一代的主要性能瓶颈。在无损结构上运行的远程直接内存访问(RDMA)可能会克服这一瓶颈。所以研究者可以借鉴了 RDMAi 方面的经验。

**关键词** 远程直接内存访问，数据中心，读写安全，云存储

## 引言

对于数据中心应用来说，网络通信是决定数据中心应用程序性能的关键因素之一。这些应用程序需要以来自网络的高吞吐量和低延迟来得到良好性能，这是具有远程直接内存访问(RDMA)能力的网络所承诺的。这些网络不仅已经成为一种商品，而且现在也成为了数据中心的一部分。支持 RDMA 的网络通过绕过传统网络协议栈，避免了它们的开销。此外，它提供了可靠的远程内存的单边读写，提供了现成的基于硬件的可靠的包交付，不需要远程 CPU 参与。尽管 RDMA 很受欢迎，但关于在有效利用 RDMA 的高性能和扩展连接数量之间找到平衡，仍存在着长期的争论。特别是，一些工作在可靠的连接上使用单边操作，以确保没有数据包丢弃。另一方面，一些工作放弃了这一点，而支持双边操作，而不是不可靠的数据报，后者提供传统的 RPC，如 API，并可以扩展到大量节点，没有任何硬件问题，比如在支持 RDMA 的 NIC(RNIC)上的缓存刷新。与此同时，其他工作使用了点对点的优点，因为它们使用特定的 RDMA 操作专门化各种算法阶段。虽然单边操作不涉及远程 CPU，但它是以硬件上的可伸缩性瓶颈为代价的。特别是，为每对连接维护连接信息的 RNIC，防止其内存有限而遭受缓存抖动。不幸的是，这些信息可能会溢出主机内存，导致缓存丢失的巨大损失，因为 RNIC 必须通过 PCIe 获取数据，这可能是几微秒。同时，开发人员通过使用不可靠的数据报(UD)来减少这种冲击，从而避免成对连接信息。

近年来，许多最先进的系统开始利用远程直接内存访问(RDMA)原语作为一种通信机制，从而实现高性能保证和资源利用率保证。在公共云中的部署，如微软 Azure 和 IBM 云，正在成为可用的，越来越多的系统使用 RDMA 进行高性能通信。然而，RDMA 体系结构的设计主要关注性能，而不是安全性。尽管有使用 RDMA 的趋势，但在上层协议中使用 RDMA 通信可能涉及的潜在安全影响和危险仍在很大程度上未得到研究。当前

的 RDMA 技术包括多个明文访问令牌，以强制隔离和防止对系统内存的未经授权的访问。由于这些令牌以明文传输，任何获取或猜测它们的实体都可以读写在网络中任何机器上使用 RDMA 暴露的内存位置，不仅损害应用程序的保密性，而且损害应用程序的完整性。为了避免这些访问令牌的损害，RDMA 架构依赖于隔离和底层网络是一个受到良好保护的资源的假设。否则，位于两个通信方之间的路径上（例如，被窃听器或恶意交换机）的攻击者可以窃听绕过数据包的访问令牌。这些也是设计者需要关心和解决的问题。不幸的是，RDMA 数据包的加密和认证，不是当前 RDMA 规范的一部分。虽然 IPsec 传输最近可用于 RoCE 通信，但 IPsec 标准不支持无限带通信。此外，应用程序级加密(例如，基于 TLS)是不可能的，因为 RDMA 操作可以在不需要 CPU 参与的情况下进行处理。由于 TLS 不能支持纯粹的单边通信例程，应用程序需要在解密之前将数据包存储在一个缓冲区中，这完全否定了 RDMA 的性能优势。这也将是研究者接下来需要考虑的问题所在。

RDMA 技术近些年还被引进了大公司的存储系统中，RDMA 引入阿里巴巴开发的云存储系统盘古存储网络的经验。自 2009 年推出以来，它已被证明对阿里巴巴的核心业务至关重要。除了性能、可用性和 SLA 需求外，盘古公司在生产规模上的部署计划还应考虑存储容量和硬件成本。研究者提出了一个支持可数字化数据库的盘古系统，它表现出优越的性能，可用性和 SLA 标准与传统的 tcp 支持的版本相匹配。支持 RDMA 的盘古已经被证明成功地提供了许多在线关键任务服务，包括几个重要的购物节。截至 2020 年，盘古系统已经被部署在数百个集群中，并且它已经管理了数十万个存储节点。此外，它还支持在许多生产环境中实现对 eb 级数据的实时访问。

## 原理和优势

对于解决 RDMA 的可扩展性问题，RDMA 通行机制是通过，RDMA 主机通过创建队列对(QP)来建立通信，其中每个 QP 由一个发送队列和一个接收队列组成。应用程序通过用户空间库向这些队列提交请求。每个 QP 都与一个完成队列(CQ)相关联，该队列包含指示先前提交的动词的完成状态的事件。RNIC 对 CQ 执行完成事件的 DMA，应用程序对其进行轮询。系统，如 FaRM 和 Storm，使用 RCqp 来利用低延迟和 cpu 效率高的内存动词。为了解决可伸缩性问题，FaRM 使用线程之间的 QP 共享，并使用 2GB 的大页面注册内存区域。Storm 使用配备了改进缓存管理的近期 rni 卡(Mellanenox 连接 X-4)，注册物理内存以消除 RNIC 将虚拟地址转换为物理地址所需的元数据。然而，RC 面向连接的特性意味着状态 RNIC 缓存将随着集群大小的增加而增加，这使得它对于高扇存、高扇出的通信模式尤其糟糕。为了验证这一点，研究者使用连接 e-5 网卡运行了一个实验，有 22 个客户端节点和一个服务器节点。客户端向服务器发出 16 字节的 RDMA 读取。随着 qp 数量的进一步增加，读取性能在 176-704 个 QPs 之间达到峰值，随后急剧下降。这是由于 RNIC 无法缓存与所有连接对应的状态。Flock 旨在提供一个可扩展的 RDMA 通信框架，针对一个具有挑战性的流量模式：一个高扇入，高扇出的网络，在今天的数据中心中普遍使用。此外，Flock 将所有 RDMA 原语——rpc 和内存——暴露给应用程序，并通过利用脱离头盔的硬件支持的可靠连接(RC)将软件诱导的开销最小化。此外，Flock 也不同于传统的 RDMA 堆栈设计。例如，研究者重新讨论了已知存在同步开销的 QP 共享的思想。Flock 解决了这种开销并保持了可伸缩的应用程序性能，即高吞吐量和低延迟，有三个贡献：连接处理抽象、基于合并的 Flock 同步和共生发送控制调度的负载控制机制，包括发送端线程调度和接收端 QP 调度。

在 RDMA 安全机制方面，在研究者的对手模型研究者考虑三方：一个 RDMA 服务主机一个或几个 RDMA 应用程序，客户端通过 RDMA 交互，和一个对手可以合法连接到 RDMA 服务，但试图违反 RDMA 的

安全机制。利用所发现的漏洞，对手可以在RDMA 网络中发起攻击，例如，通过使用未经授权的访问内存区域或通过使用 DoS 中断通信。此外，RDMA 中的漏洞也可能被误用作应用程序级攻击（如恶意软件）的攻击向量。这里有四个安全弱点。首先，RDMA 允许一台机器通过网络直接访问远程机器上的数据。由于单边 RDMA 操作的网络卸载，所有的内存访问都是使用 RNIC 上的专用硬件执行的，而没有任何 CPU 交互。这使得内存访问对应用程序完全不可见，并限制了它们检测攻击的能力。其次，现有的 RDMA 网络协议不提供任何对 RDMA 数据包的身份验证或加密的机制。对手可以欺骗包头中的任何字段，或更改 RDMA 消息的包有效负载中的任何字节。再者，为了减少 rnic 上的状态开销，RDMA 连接管理器在默认情况下为单个进程中的所有已建立的 QPs 和内存注册使用单个保护域。因此，单个进程的所有 qp 都可以彼此访问内存。尽管如此，即使是使用本机连接接口的开发人员似乎也选择对所有的 QP 和内存注册使用单一的 PD。最后，隐式随需寻呼(ODP)允许进程注册其完整的内存地址空间。此特性用于高性能通信设置，其中频繁注册通信缓冲区的开销会导致性能下降。ODP 不需要注册内存，因为任何内存地址都可以由 RNIC 按需注册。如果启用了 ODP，攻击者可以远程访问进程的整个内存空间，从而导致很高的攻击潜力。虽然默认该功能被禁用，但高性能通信系统的最新进展使该特性在 IB 部署中获得了吸引力。由于有这些问题的存在，所以，启用 RDMA 的系统的开发人员必须意识到 RDMA 网络带来的威胁，并应该采用缓解措施，如使用内存窗口，为每个连接使用一个单独的 PD，以及提出算法来随机化 QPN 和 rkey 生成。

## 研究进展

### FLOCK 框架

首先研究者介绍一下解决扩展问题的 flock 框架，图 1 显示了运行中的 Flock。每个发送方节点和接收方节点都由 qp 连接。在与连接句柄抽象建立连接之后，发送方使用 Flock 同步发送请求，即，发送方的一个线程收集并合并一组请求，并使用活动 QP 将这些请求发送到服务器。发送方还要求通过信用更新方案保持 QP 的活动状态，并向服务器提供其他 QP 利用率指标。服务器接收到消息后，执行两个任务：首先，它使用请求调度程序来处理消息中的所有请求（2）。其次，它启动了针对 QP 调度的 QP 调度程序，它根据接收到的利用率指标激活/停用它们（3）。然后，服务器会响应已处理过的请求，以及需要时更新的信用证（4）。一旦发送方收到响应，其响应调度器就会通知适当的应用程序线程（5）。现在，发送方使用线程调度程序进行发送方端线程调度，即，线程调度程序根据服务器响应将线程从停用的 QP 迁移到活动的（6）。

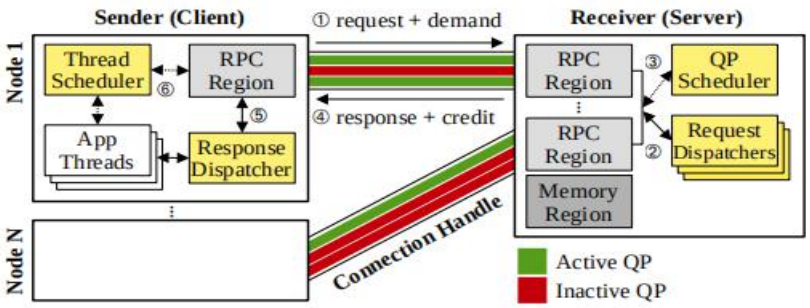


图 1　FLOCK 框架调度图示

为了有效地利用 qp，研究者在 qp 之上引入了一个连接句柄抽象，它允许多个线程共享它们。Flock 的主要 api，研究者将其分为连接设置和内存区域注册、RPC 客户端/服务器（发送方/接收方）和内存 api。发送方可以发出 RPC 和内存操作，而接收方只需要处理传入的 RPC 请求，因为内存操作不消耗接收端的任何 CPU。研究者的编程 API 是基于连接句柄的，它在两个 RDMA 节点之间建立一对一的连接，如图 1 所示。当用户建立到远程节点(fl_connect)的连接时，Flock 会创建一个具有代表性的连接句柄，所有其他 api 都会操作该连接句柄以与远程节点通信。在内部，Flock 管理一个连接句柄的一组 RCqp。每个连接句柄附加一个或多个 RDMAMRs，其中单独的 MRs 用于 RPC 和内存操作。至于研究者的实现，Flock 使用了本机操作系统线程。与之前在高性能网络[33,34]中进行的工作相比，它不实现或依赖于特定的线程库。但是，它可以很容易地扩展到与用户级线程库集成。

由于多个线程共享一个 QP，所以主要的挑战在于有效地利用 RNIC。为了解决这个问题，研究者采用了一种名为 Flock 的领导者-追随者协调形式同步 QP 共享。充当前导符的应用程序线程合并来自其他并发线程——追随者——的 RPC 请求，然后为合并后的消息发出一个 RDMA 写入操作。为了以动态的方式有效地选择一个领导者，Flock 为每个 QP（图 2 中的 3,4）维护一个线程组合队列(TCQ)。每个 TCQ 维护一个并发线程的队列，其中锁尾指向队列尾部。应用程序线程首先通过使用原子交换操作更新 Flock 尾部，将自己排队到 TCQ 以访问 QP。TCQ 遵循类似于 MCS 队列锁[26]的更新协议；如果交换操作后 Flock 尾部为空，则线程位于 TCQ 的头，并成为领导者（3）。否则，它将成为一个追随者（4）。领导者根据其他并发线程所请求的有效负载为其他并发线程提供内存缓冲区。每个线程首先将其有效负载复制到所提供的缓冲区中，并更新其复制完成标志。领导者对每个线程的复制完成标志进行轮询；一旦设置，它最终设置消息头和金丝雀，并为合并的消息发出一个 RDMA 写入（图 2 中的 5）。请注意，领导者为追随者提供了有限数量的缓冲区，以确保领导者的应用程序级进度。在达到这个限制时，领导者将执行上述复制、合并和写入消息的步骤。然后它将领导交给 TCQ 中的第一个追随者，他们的请求不是合并信息的一部分。研究者想强调的是，应用程序线程是 Flock 中的领导者或追随者的概念是一个短暂的概念，具有基于 TCQ 的可伸缩的领导者选择。在 Flock 中，合并消息中的请求数量取决于使用共享 QP 的并发线程。在没有并发性的情况下，合并的消息只包含属于该引头的请求。另一方面，在并发性的公司中，合并会带来一定的同步成本，但也有几个优点。合并是有益的，因为它减少了提交 RPC 请求所需的 MMIO 操作的数量，节省了进程中的 CPU 周期。此外，发送单条大消息可以比发送许多小消息更有效，因为线路上的每条消息都需要额外的元数据，如网络传输信息。此外，研究者的消息格式要求每个消息头和金丝雀。因此，合并减少了发送的字节数，从而更有效地利用了网络带宽。为了减少共享资源上的争论，领导者除了管理请求缓冲区外，还负责轮询完成事件。

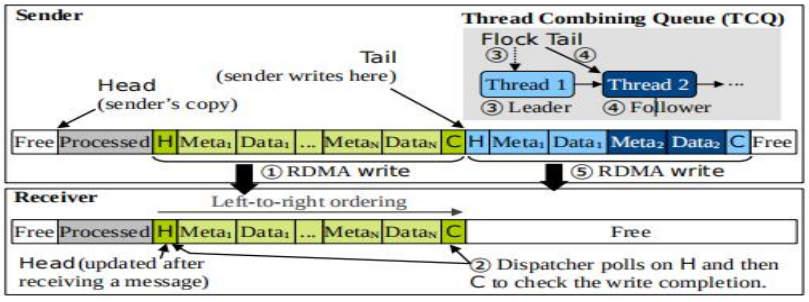


图 2　FLOCK 进程调度图示

服务器通过在引导期间调用fl_reg_handler，为每个RPCID注册一个RPC处理程序。它通过轮询RPC的请求缓冲区来检测包含RPC请求的新的合并消息。然后，它执行与每个请求对应的RPC处理程序，并在其响应缓冲区中准备响应，如图2所示。应用程序可以使用RPC调度程序线程或使用应用程序管理的RPC工作人员池来执行该请求。服务器将请求序列ID标记为其响应元数据，这样客户机就可以用该ID来匹配对相应请求的响应。类似地，线程ID也被标记为响应元数据。由于服务器所使用的消息格式与在客户端上使用的消息格式相似，因此RPC响应也会被合并成更大的消息。此外，服务器支持其控制变量Head和合并响应，服务器使用RDMA写入发布。除了通过Flock同步（4.2）减少线程之间QP共享的开销外，研究者还进一步减少了响应缓冲区上的争用，以保持RPC的可伸缩性。如图2所示，研究者使用一个响应调度程序线程，该线程轮询响应缓冲区，并根据每个响应中标记的线程ID将可用的响应中继到适当的应用程序线程。由于应用程序线程脱节地访问它们各自的响应，因此调度器允许对响应缓冲区及其内存回收的并行数据访问。调度程序线程不与RDMA堆栈交互，或者涉及特定于应用程序的处理，这使得其作业相对轻量级。因此，它可以跨多个qp工作，从而管理多个响应缓冲区。

通过以RC模式为基础，通过共享qp对，和动态调整线程调度和qp活跃状态可以完成RDMA的优化和扩展工作。

## RDMA 安全机制

### 内存保护密钥

其次为了解决RDMA的安全性问题，研究者从内存的角度来解决这个问题。为了保护远程内存不受未经授权的内存访问，IBA要求RDMA读/写请求包含一个远程内存访问密钥rkey，该请求在通信对等点之间进行协商，并在远程RNIC上进行检查。具有无效rkey的数据包会导致连接错误，从而导致断开连接。研究者的要求是包含rkey，攻击者不能禁用驱动程序代码。因此，为了成功地绕过这种保护机制以防止未经授权的内存访问，攻击者需要在其请求中包含一个有效的rkey。研究者分析了不同RNIC模型和驱动程序的rkey生成过程的随机性。对于所有被测试的设备，rkey的生成与要注册的缓冲区的地址和长度无关。访问标志中的更改对rkey的生成没有影响。生成的重键仅依赖于以前的注册/注销操作。此外，研究者还研究了注册/去注册如何影响内存注册。来自博通的RNIC模型总是将rkey值增加0x100，独立于前面提到的因素。因此，假设攻击者能够获得属于这一系列增加键值的一部分的rkey，那么预测之前或随后的rkeys是微不足道的。对于基于mlx4驱动程序的设备，rkey的序列依赖于注册/注销操作。对于连续的注册操作，每个rkey都会增加0x100。但是，在取消注册操作之后，下一个rkey将根据已取消注册的内存区域的rkey而增加0x80000。在多个连续的去注册操作中，去注册内存区域的rkey排队，并且对于每个即将进行的注册操作都有一个密钥退出。密钥生成的算法可以在附录A中找到。所有被测试的AzureHPC实例(A8、A9、H16r)都使用mlx4驱动程序，并使用前面描述的算法分配rkeys。RoCE的软件实现，SoftRoCE，也为每个注册操作增加了rkey0x100，但另外使用线性反馈移位寄存器(LFSR)随机化了最后8位。然而，由于lfsr是确定性的，并且初始种子是已知的，因此所有后续状态都很容易计算。此外，SoftRoCE使用的LFSR实现只生成了15个不同的数字，这并不会增加rkeys的随机性。基于mlx5驱动程序的设备不在后续注册之间使用固定的增加，但仍然严格地使用随机值增加值。对这些值的分析表明，有超过60%的概率，选择值0x101或0x102。因此，即使基于mlx5驱动程序的设备的密钥生成过程比其他驱动程序包含更高的熵，生成的密钥序列仍然可以被对手通过适度的努力来预测。

**内存随机分配**

除了与存储器位置相关联的 rkey 外，对手还需要预测相应的存储器地址。通常，诸如地址空间布局随机化(ASLR)等技术会随机安排进程的地址空间位置。这可以防止攻击者通过随机化对象的位置直接引用内存中的其他对象。然而，内存中的后续对象针对随机地址基以连续地址分配。例如，通过 mmap()Linux 系统调用分配的所有对象都被并排放置在 mmap 区域中。由于基于 RDMA 的应用程序运行在目标主机上的单个进程中，因此它们不受 ASLR 的保护，而是将内存中的对象并排分配。假设攻击者知道目标主机上的内存对象的地址，那么预测其他对象的内存地址是可能的。即使内存区域的连续分配不是由 RDMA 协议引起的，但它仍然会影响RDMA应用程序的安全性。QP 编号标识符和分组序列编号

研究者的评估显示，对于所有被测试的设备和驱动程序，QP 编号是按顺序分配的。假设一个对手自己注册了一个 QP 或观察了一个 QP 注册请求，那么预测之前或随后的 QP 数字是微不足道的。此外，由于 IBA 使用 24 位 QP 号，因此在一个 RNIC 中不可能建立超过 2 24 个 QP 连接。RDMA 的实现提供了两种建立 RDMA 连接的方法：一个本机的 RDMA 连接接口或使用 RDMA 连接管理器来建立连接。使用本机连接接口，由应用程序开发人员设置连接参数，如目标 QP 号、本地和远程启动 PSNs。RDMA 连接管理器将这个负担从应用程序开发人员那里转移开，并随机生成一个启动 PSN（使用加密伪随机数生成器），从而使 RDMA 连接建立的过程类似于 TCP 套接字。研究者的分析显示，许多基于 RDMA 的开源应用程序选择使用本机接口并手动设置启动程序 psns。如果起始的 psn 不是在每个 QP 连接的基础上进行随机化的，那么预测已建立的连接的 psn 就会变得更加简单。

**RDMA 应用**

最后我们来说说 RDMA 技术在云存储和集群中的应用情况，存储集群的部署规划控制着网络拓扑结构、RDMA 通信范围、存储节点配置等。必须考虑多种因素，包括将存储卷与需求相匹配、控制硬件成本、优化性能、最小化可用性和 SLA 风险。最终的结果是在所有这些因素之间进行权衡。例如，微软在整个 Clos 网络的规模上部署了 RDMA。研究者的 RDMA 部署所采用的关键原则是可用性优先原则。网络和节点配置。盘古的基于 clos 的网络拓扑结构。与常见的双家庭做法一致，研究者部署了梅拉诺克斯 CX 系列双端口 rnic，用两个不同的 ToR 交换机连接一个主机。特别是，两个物理端口被绑定到单个 IP 地址。网络连接(例如，RDMA 中的 qp)以循环的方式在两个端口上进行平衡。当一个端口关闭时，可以将此端口上的连接迁移到另一个端口。表 1 报告了 25Gbps 和 100GbpsRNIC 存储节点的典型硬件配置。

表 1　存储节点配置

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

每个节点的SSD数量由总RNIC带宽与单个 SSD 的吞吐量决定，允许 I/O 吞吐量与网络带宽相匹配。请注意，25Gbps 和 100Gbps 配置中的 SSD 类型是不同的，从而导致不成比例的数字。计算和存储节点部署在单个播集中的不同机架中。然后根据计算需求计算计算节点和存储节点的数量。为了最小化故障域，研究者只启用了每个播集内和存储节点之间的 RDMA 通信。计算节点和存储节点之间的通信通过私有用户空间 TCP 协议执行。这是由于计算节点的硬件配置复杂，因此更新迅速。因此，TCP 可以作为一个独立于硬件的传输协议有效地应用。用户空间 TCP 比内核 TCP 更方便升级和管

理，而内核 TCP 的通用性被选择用于跨播集通信。生产部署是播客集级 RDMA 所关注的另一个问题。在许多数据中心中，播客集位于不同的建筑中。对于交叉构建的 RDMA 链路，基本链路的延迟要大得多，而 PFC 机制需要更大的净空缓冲区。为了启用 RDMA，必须仔细调整和测试位于脊柱开关上的 PFC/ECN 阈值。这是一项艰巨的任务，目前，还没有带来足够的收益。据研究者所知，以前关于 RDMA 部署的研究并没有探索 RDMA 和 TCP 混合服务。研究者将 TCP 作为盘古遵循可用性优先原则的最后手段。尽管目前取得了进展，但 RDMA 设备还远非完美无瑕。因此，当可用性或 SLA 受到威胁时，将受影响的链路从 RDMA 切换到 TCP 可以保持可用的带宽。此转义计划不会影响未受影响的 RDMA 链接。然而，在混合部署过程中，研究者确定共存的 TCP 流量引发了大量的 TX 暂停(即由网卡发送的 PFC 暂停帧)，即使 RDMA/TCP 流量被隔离在两个优先级队列中。表 2 报告了盘古盘中在 TCP 流量约为 50%的不同负载下的 TX 暂停生成率。测试在麦拉诺克斯 CX-4 25Gbps 双端口 rnic 上进行，可能导致 PFC 风暴。研究者与 Mellanox 一起研究了这个问题，并确定了在 Linux 内核中对 TCP 的处理是高度 I/O 密集型的。内核 TCP 在 NICs 的 PCIe 总线上启动了太多的部分写入。随着 PCIe 带宽的消耗，NIC 的接收管道就会减慢。缓冲区溢出，NIC 随后开始传输 PFC 暂停帧。这就是 RDMA 在集群和云存储中的应用。

## 总结与展望

对于 RDMA 的扩展问题，我们提出了 flock 框架，Flock 使用三种贡献重新访问了 QP 共享的想法。Flock 以连接句柄抽象的形式将间接指向线程之间的多路 qp。它使用了一种领导者-追随者同步机制，允许在线程之间共享 QP，通过有效的网络利用率和减少 CPU 开销来提高性能。控制服务器上最大客户端负载以及确保 qp 的公平利用的关键启用器。还有就是 RDMA 存在很多的安全风险，启用 RDMA 的系统的开发人员必须意识到 RDMA 网络带来的威胁，并应该采用缓解措施，如使用内存窗口，为每个连接使用一个单独的 PD，以及研究者提出的算法来随机化 QPN 和 rkey 生成。

未来，RDMA 技术还将在多个领域发挥它的作用，例如：星云平台在 TensorFlow 框架优化中引入的关键技术之一就是 RDMA 技术。通过 RDMA 消除多 GPU 跨节点通信的网络瓶颈，显著降低了训练任务整个周期中的通信耗时占比，提高了 GPU 集群计算资源利用率和训练效率，也为集群横向扩展到更大规模时的线性加速比提供了保证。类似的例子还有很多，总之随着技术的发展，RDMA 技术将更加安全，更加高效，更加广泛的应用到各个技术领域当中。

## 参考文献


[1] Monga S K, Kashyap S, Min C. Birds of a Feather Flock Together: Scaling RDMA RPCs with Flock[C]//Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles. 2021: 212-227.

[2] Wei X, Dong Z, Chen R, et al. Deconstructing RDMA-enabled distributed transactions: Hybrid is better![C]//13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18). 2018: 233-251.

[3] Rothenberger B, Taranov K, Perrig A, et al. ReDMArk: Bypassing {RDMA} Security Mechanisms[C]//30th {USENIX} Security Symposium ({USENIX} Security 21). 2021.

[4] Gao Y, Li Q, Tang L, et al. When Cloud Storage Meets {RDMA}[C]//18th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 21). 2021: 519-533.