基于 RDMA 的分布式系统研究综述

董梁

(华中科技大学, 计算机科学与技术学院， 武汉 中国 085400)

摘 要 远程内存直接访问(remote direct memory access，RDMA)技术在分布式系统中已经有了广泛的应用，与传统的 TCP/IP 方式不同，RDMA 具有旁路内核，不需要 CPU 参与等优势，随着网络带宽的提升，当前 RDMA 在分布式系统中的性能也大大提升。本文介绍了 RDMA 使用的背景，与 TCP/IP 相比较 RDMA 带来的性能提升以及 RDMA 的优势；为了更清晰的说明 RDMA 在分布式系统中的广泛应用，本文以分布式系统中的事务处理，键值存储以及文件系统等为例，介绍了常用的 FaRM, Pilaf 等 RDMA 应用程序。最后，本文着重介绍了 PRISM，一种基于 RDMA 接口进行的拓展。本文从 RDMA 提供的四种基本原语以及单边和双边操作来说明了RDMA存在的问题。本文说明了分布式系统中应用RDMA有哪一些不足，以及 PRISM 做出了那些改进。最后本文给出了总结和展望。

关键词 **远程内存直接访问； 分布式系统； 键值存储； 文件系统； 拓展接口**

中图法分类号 TP258

Overview of Research on Distributed System Based on RDMA

Dong Liang

(Department of computer science and technology, HuaZhong University of science and technology, Wuhan, China)

Abstract Remote direct memory access (RDMA) technology has been widely used in distributed systems. Unlike the traditional TCP/IP method, RDMA has the advantage of bypassing the kernel and does not require CPU participation. With the increase of bandwidth, the performance of current RDMA in distributed systems has also been greatly improved. This article introduces the background of the use of RDMA, the performance improvement brought by RDMA and the advantages of RDMA compared with TCP/IP; in order to more clearly explain the wide application of RDMA in distributed systems, this article uses transaction processing in distributed systems, Take key-value storage and file systems as examples, and introduce commonly used RDMA applications such as FaRM and Pilaf. Finally, this article focuses on PRISM, an extension based on the RDMA interface. This article explains the problems of RDMA from the four basic primitives provided by RDMA and unilateral and bilateral operations. This article explains the shortcomings of using RDMA in distributed systems, and what improvements PRISM has made. Finally, this article gives a summary and outlook.

Key words remote direct memory access; distribute system; key-value store; file system; extend interface

1 引言

1.1 背景：

当前计算机处于飞速发展的阶段。CPU 的运算速度以及当前网络带宽不断增长，同时，内存日益廉价，这使得构建内存存储系统变为可能。大数据计算领域具有数据规模大，数据维度高，数据种类多三个典型特征，内存计算技术逐渐开始发挥重要作用。SPARK，HANA 等大数据软件已经得到工业界的广泛关注。相比于磁盘，DRAM 在带宽和延迟上均有几个数量级的优势，能极大地提升本地数据存储性能，然而，DRAM 也存在集成度低等问题，单节点内存最大只能达到几百 GB，无法满足大型应用的存储需求。

分布式系统具有易于拓展以及能够充分利用计算资源的特点，因此构建分布式内存系统是一种有效途径。与 DRAM 进行计算有所不同，分布式系统是整合多台计算资源，因此分布式系统的性能依赖于网络带宽以及延迟。近年来，远程内存直接访问（remote direct memory access，RDMA）作为一种新兴的跨网数据传输技术逐渐从高性能计算走进大数据领域，并开始受到广泛关注。RDMA 技术能够在远端 CPU 不参与的情况下，绕过内核直接访问远端内存，实现零拷贝的数据传输。近年来网络带宽的不断发展，高速网卡的出现，使得 RDMA 的运算速度能够与 DRAM 持平。

1.2 RDMA原语：

然而，简单地将现有的分布式存储系统中的网络模块替换为 RDMA 通信模式，而不优化上层软件逻辑的策略，并不能充分发挥 RDMA 网络的硬件优势。在高性能通信方面，RDMA 提供多种操作原语以及通信模式，对于不同的场景和需求，需要作出不同的选择。RDMA 通信原语有 SEND/RECV、READ、WRITE 四种，（如图 1）

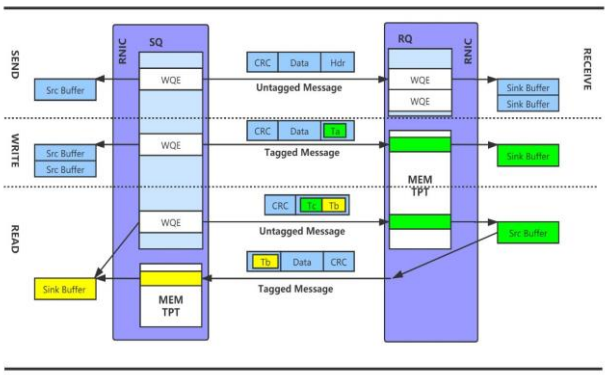

图 1 RDMA 操作原语

READ/WRITE 是单端的，完全不需要 CPU 的参与，而 SEND/RECV 是双端的，需要一端发起 SEND 请求，而另一端发起 RECEIVE 请求，才能完成一次完整的读写操作。双向操作具有传统的消息传递语义。SEND 操作向调用 RECEIVE 的远程应用程序发送消息。单面操作允许主机在远程主机上读取或写入内存（在预先注册的区域中）。当前使用者们针对使用单面操作还是双面操作进行了大量辩论。单面操作速度更快、CPU 效率更高，但仅限于简单的读写操作。双向消息传递，因为它允许在两端进行处理，所以即使通信操作本身较慢，整体上也可能会有更快的效果。

1.3 RDMA困境：

这里，我们以 Pilaf，一个早期基于 RDMA 的键值存储为例来说明双端操作和单端操作分别的优劣。Pilaf 中使用 RDMA 操作访问键值如图 2，

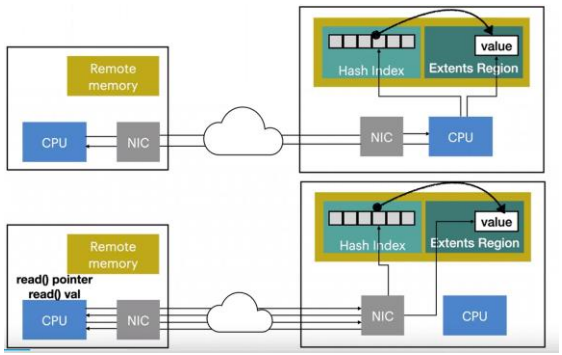

图 2 Pilaf 访问哈希表

Pilaf 将指向键值对象的指针存储在哈希表中，实际数据存储在另外的存储结构中，这两种结构都通过 RDMA 公开。在单端操作下，客户端第一次远程读取哈希表可以获取到存储区对应的指针，第二次使用指针来执行对数据块存储的远程读取。这不需要服务器端 CPU 参与，但需要两次网络往返。使用双面操作构建的传统键值存储实现只需要一次往返，但每次操作都涉及 CPU。通过此例说明了双端操作需要 CPU 的额外参与，而单端操作则是需要额外的网络往返。因此当前 RDMA 如何在分布式系统下得到充分的应用就需要考虑一个关键的问题，那就是到底是使用单面的 RDMA 操作即一次简单的读写带来的效率更高还是单次的 RPC 操作会带来更高的网络效率。

本文研究的内容包括了 RDMA 在分布式系统当中几个典型的使用，以及如何在分布式系统中发挥 RDMA 的优势。关于调整 RDMA 以使得 RDMA 在分布式系统中充分发挥作用这一点，当前存在一个共识，调整应用程序使其能在 RDMA 上运行比较复杂且代价高昂。这种复杂性的根源在于 RDMA 接口本身。RDMA 以接口的形式提供了 READ,WRITE,RECEIVE,SEND 四种操作，然而很少有分布式应用程序能够以简单的 READ 和 WRITE 操作的形式轻易地表达其逻辑。因此，许多 RDMA 应用被迫添加额外的操

作，引入了额外的网络往返，从而牺牲了 RDMA 本应该带来的延迟上的收益。其他人使用混合设计，这些设计要求应用程序 CPU 参与某些操作或者在某些情况下，仅仅使用 RDMA 来实现更快的消息传递协议，然而这就否定了 CPU 旁路的好处。

关于 RDMA 在分布式系统中的应用，一个最经常被提到的问题也就是到底 RDMA 提供的 4 种操作方式哪一种的性能会更好。Pilaf，FaRM 等研究认为 RDMA 的单边操会有更少的内存使用，因此采用 RDMA 单边操作。FaSST 认为在设计分布式数据存储时，单边 RDMA 的灵活性有限，会降低性能并增加软件复杂性；其次，在现代 RDMA 网络上很少看到丢包。因此使用双边操作来实现。而 PRISM 研究认为，当前仅使用单边或仅使用双边都是一种无奈的选择，PRISM 认为为了充分发挥网络加速的潜力，建立低延迟系统，必须超越基本的 RDMA 接口。最初设计用于支持并行超级计算应用程序的 RDMA 接口无法满足当今分布式系统的需求。因此提出了一种基于 RDMA 接口的拓展。

RDMA 绕过内核实现数据零拷贝，并借助硬件执行数据包的派送和解析，从而提供了高带宽、低延迟的通信特性。但是，将 RDMA 技术应用到分布式 系统中时，也存在原语漏用、冗余拷贝、协议低效等问题急需解决。后续本文探讨的主要内容也就是 RDMA 在分布式系统中常见的使用以及当前针对 RDMA 不兼容分布式系统所提出的一些解决思路。

2 原理和优势：

2.1 RDMA工作原理:

RDMA 允许本地 CPU 绕过操作系统，直接读写远端节点内存，该过程无需远端 CPU 的参与。以远程写操作为例，本地 CPU 直接以 MMIO (memory mapped IO)的方式向网卡发起远程写命令，并传递相应参数，包括待写入数据块基地址、远 端内存地址、写入数据块大小、远端注册内存密钥等信息；本地网卡收到命令之后，立即根据本地数据块 基地址将数据块从主存以 DMA READ 的方式读取到网卡缓存，并发送到远端；远端网卡接收到数据块 之后，以 DMA WRITE 的方式直接将数据写入内存对应地址。此过程中，RDMA 无需像传统以太网一样穿越内核中的多层网络协议栈，因此实现了跨节点数据传输过程中的数据零拷贝。

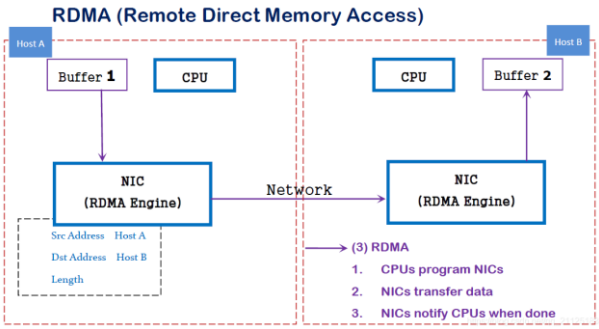

图 3 RDMA 工作原理

2.2 RDMA 的工作过程：

1) 当一个应用执行 RDMA 读或写请求时，不执行任何数据复制.在不需要任何内核内存参与的条件下，RDMA 请求从运行在用户空间中的应用中发送到本地 NIC(网卡)。

2) NIC 读取缓冲的内容，并通过网络传送到远程 NIC。

3) 在网络上传输的 RDMA 信息包含目标虚拟地址、内存钥匙和数据本身.请求既可以完全在用户空间中处理(通过轮询用户级完成排列)，又或者在应用一直睡眠到请求完成时的情况下通过系统中断处理.RDMA 操作使应用可以从一个远程应用的内存中读数据或向这个内存写数据。

4) 目标 NIC 确认内存钥匙，直接将数据写入应用缓存中.用于操作的远程虚拟内存地址包含在 RDMA 信息中。

2.3 RDMA的优势：

这里我们主要以 RDMA 与传统的 TCP/IP 模式下数据拷贝为例进行对比，如图所示。

传统的 TCP/IP 技术在数据包处理过程中，要经过操作系统及其他软件层，需要占用大量的服务器资源和内存总线带宽，数据在系统内存、处理器缓存和网络控制器缓存之间来回进行复制移动，给服务器的 CPU 和内存造成了沉重负担。尤其是网络带宽、处理器速度与内存带宽三者的严重"不匹配性"，更加剧了网络延迟效应。

RDMA 是一种新的直接内存访问技术，RDMA 让计算机可以直接存取其他计算机的内存，而不需要经过处理器的处理。RDMA 将数据从一个系统快速移动到远程系统的内存中，而不对操作系统造成任何影响。在实现上，RDMA 实际上是一种智能网卡与软件架构充分优化的远端内存直接高速访问技术，通过将 RDMA 协议固化于硬件(即网卡)上，以及支持 Zero-copy 和 Kernel bypass 这两种途径来达到其高性能的远程直接数据存取的目标。使用 RDMA 的优势如下：

1) 零拷贝(Zero-copy) - 应用程序能够直接执行数据传输，在不涉及到网络软件栈的情况下。数据能够被直接发送到缓冲区或者能够直接从缓冲区里接收，而不需要被复制到网络层。

2) 内核旁路(Kernel bypass) - 应用程序可以直接在用户态执行数据传输，不需要在内核态与用户态之间做上下文切换。

3) 不需要 CPU 干预(No CPU involvement) - 应用程序可以访问远程主机内存而不消耗远程主机中的任何 CPU。远程主机内存能够被读取而不需要远程主机上的进程（或 CPU)参与。远程主机的 CPU 的缓存(cache)不会被访问的内存内容所填充。

4) 消息基于事务(Message based transactions) - 数据被处理为离散消息而不是流，消除了应用程序将流切割为不同消息/事务的需求。

5) 支持分散/聚合条目(Scatter/gather entries support) - RDMA 原生态支持分散/聚合。也就是说，读取多个内存缓冲区然后作为一个流发出去或者接收一个流然后写入到多个内存缓冲区里去。

在具体的远程内存读写中，RDMA 操作用于读写操作的远程虚拟内存地址包含在 RDMA 消息中传送，远程应用程序要做的只是在其本地网卡中注册相应的内存缓冲区。远程节点的 CPU 除在连接建立、注册调用等之外，在整个 RDMA 数据传输过程中并不提供服务，因此没有带来任何负载。总之 RDMA 是一种低延迟，高吞吐，高效率以及占用较低 CPU 资源的一种通信方式。

3 研究现状：

本章中我们主要会介绍RDMA在当前分布式系统中几种比较常见的应用，以及针对当前 RDMA 不兼容分布式系统所作出的一些改进。

3.1 RDMA在key value中的应用：

在传统存储系统中，数据的组织和索引由服务端本地执行．一般地，客户端读取或更新服务端数据时，首先向服务端发送 RPC(remote procedure call)请求，服务端接收到请求 后，迭代式地查询树状或散列组织的数据，然后将查询或更新结果返回给客户端。Key-value 系统采用平铺式的数据存储管理模式，仅提供类似 GET,PUT 等接口，系统结构简单。同时，RDMA 可以直接访问远端内存数据，这使得分布式键值存储系统中的数据索引 模式发生改变。结合 RDMA 和 key-value 存储的分布式键值存储系统被广泛研究。

Pilaf 是纽约大学于 2013 年提出的一个内存级分布式键值存储系统，借助 RDMA 原语实现了极 高的系统性能，同时有效降低了服务端 CPU 开销。Pilaf 在处理 GET 请求时，利用 RDMA 内存语义低延迟的特性，通过客户端发起多次 READ 请求完成键值查询，将数据索引任务由服务端转移到客户端 Pilaf 的键值对通过散列表索引，散列表和键值对统一存放在内存区域中，散列表中的各表项存放键值对的内存地址，用于索引真实的键值对。存放上述内容的内存空间在系统初始化时分配，并注册到网卡，使得客户端可远程直接访问，客户端在接入时获取服务端注册内存的虚拟地址。客户端在执行 GET 操作时，首先计算出 Key 相应散列值，并根据散列值确定对应表项在散列表中的偏移；然后执行 RDMA read 读取散列表在该偏移下的内容，如果表项包含 1 个有效地址，则根据该地址执行第 2 次 Read 操作获取键值内容。如果获取的键值对匹配，则成功返回；如果散列表项为空，或者键值对不匹配，则线性查找下 1 个表项，直到查询成功。Pilaf 借助多次 read 请求将 GET 处理逻辑转移到客户端，这种远端索引方式一定程度降低了服务端的处理开销。同时，依靠 RDMA 的硬件优势，系统处理请求的能力相比于传统方式提升显著。

HERD[2]认为在 InfiniBand 和 RoCE 中即使使用不可靠的传输，也不会由于缓冲区溢出而丢失数据包。数据包丢失的原因包括线路上的位错误和硬件故障，这种情况极为罕见。因此 HERD 客户端使用 RDMA 不可靠连接写入将请求发送到服务器的内存。这块内存通过 shmget 申请，为了防止客户端之间的竞争，每个客户端在每个核心中都有一个固定的槽位。服务器的 CPU 轮询其内存中是否有传入请求。接收到新请求后，它将在其本地 KV 存储引擎中执行 GET 或 PUT 操作，并将响应发送回客户端。对于服务端而言，使用 WRITE 发送数据使用大量的队列，因而会限制连接的可扩展性，每个发送的请求（发送的请求都需要保存在发送队列中）都可能导致 RNIC 缓存未命中。因此服务端的响应是作为 SEND 消息通过数据报连接发送，虽然有较多的 WRITE 的 QP(Queue Pairs)，但都是未激活的，无需保存在缓存中。

3.2 RDMA在事务处理中的应用：

DrTM+R[3]利用 HTM（hardware transaction memory，事务性内存）的强原子性和 RDMA 的强一致性，以保持严格的可串行性和高性能。HTM 用来代替/优化使用锁的并发控制，在硬件层面上实现写入内存事务的特性：数据的写入都是先写到 cache 中，保证了隔离性；同时修改了 cache 一致性协议，可以在 cache 中检测出数据的冲突，检测到冲突之后事务终止，从而可以保证事务的原子性。RDMA 操作与目标 CPU 中的内存操作具有很强的一致性，与内存操作保持高速缓存一致（数据先写入 L3 缓存）。因此，当与 HTM 的强一致性相结合时，RDMA 操作可以无条件中止目标机器中发生冲突的 HTM 事务。

FaRM 是在 RDMA 上的事务处理中的现有技术的代表性示例。在 FaRM 中，数据存储在可通过 RDMA 访问的散列表中，并且每个对象都与版本号和锁关联。在事务执行期间，客户

端通过使用单面 RDMA READ 操作访问服务器内存来执行读取，并在提交阶段之前本地缓冲写入。对于 FaRM 的键值存储变体，每个访问可能需要两个 READ，如在 Pilaf 中。提交过程是一个需要 CPU 参与的三阶段协议。客户端首先锁定写入集中的所有对象。完成后，它们将重新读取读集中的所有对象，以验证它们是否未被同时修改。最后，它们更新已写入的对象，并解锁它们。第二阶段可以使用 READ 来实现；另外两个需要 RPC。FaRM 从 RDMA 的低延迟传输中受益，但它仍然需要大量的服务器 CPU 参与。虽然读取是使用单面 RDMA 操作来实现的，但提交阶段的更复杂逻辑并不会是简单的单面操作。

3.3 RDMA在文件系统中的应用：

RDMA 更多地被应用到键值存储和分布式事务系统，而分布式文件系统由于结构复杂，很难充分发挥出 RDMA 的硬件特性。目前，也有部分分布式文件系统开始尝试支持 RDMA 网 络，以提供更高的性能。但是，这些系统大多采用了模块化的软件设计，将网络传输、文件存储和控制逻辑严格分离，而在引入 RDMA 网络的时候，仅仅采用了简单的网络通讯模块替换，而不是选择重构文件系统的内部逻辑，因此取得的效果甚微。

清华大学于2017年提出的分布式持久性内存文件系统 Octopus，通过紧密结合 RDMA 特性，重新设计了文件系统软件逻辑。各个节点将数据存储区注册到内存，并共享到集群使之可被远程直接访问，进而构建持久性共享内存，而元数据区域则由服务节点进行本地管理。Octopus 通过引入持久性共享内存以降低数据冗余拷贝，进而提供接近硬件的读写带宽；引入客户端主动式数据传输来重新均摊客户端和服务端之间的网络负载；引入自识别远程过程调用协议以提供低延迟元数据访问性能。

3.4 RDMA双边操作的使用：

上述所提到的大多采用的是 RDMA 中的单边操作，即仅使用到了 RDMA 当中 READ 和 WRITE 操作。FaSST 认为在设计分布式数据存储时，单边 RDMA 的灵活性有限，会降低性能并增加软件复杂性；其次，在现代 RDMA 网络上很少看到丢包。从而，FaSST 使用基于 UD 的双边消息原语来避开单边内存原语进行快速 RPC，这种方法提供了更好的性能，可伸缩性和简单性，而无需使用软件中昂贵的可靠性机制。

FaSST RPC 的主要功能包括与协程的集成，以有效地隐藏网络延迟，并进行优化，在启动网络 I/O 之后产生一个协程，从而允许其他协程在 RPC 运行期间进行有用的工作。 为了节省 CPU 周期，减少写成切换的开销，FASST 使用批处理请求和响应：将 CPU 必须振铃（ring）的 NIC 门铃的数量从 b 减少到 1，允许 RPC 层合并发送到同一目标计算机的消息。当主程序轮询 NIC 中的新数据包时，它通常收到一批请求数据包，之后为每个请求调用请求处理程序，并组装一批响应数据包，使用一个门铃（doorbell）发送这些响应。

eRPC[8]是基于 FASST[7]的工作，加入了可靠性和拥塞以及流量控制，能够在有损网络提供于 FaSST 一致的性能。为了减少线程之间的通信开销，eRPC 请求的处理函数由用户指定在分发线程或工作线程中执行请求处理。防止发送过多的数据而接收端的接收队列中没有足够的 WQE，实现了基于 credit 的流量控制。文中提出 session 的概念，每个 session 代表了端到端的连接，每个 session 在初始情况下有固定数量的 credits，每次发送都会消耗一个 credits，每收到一个确认（可能是单独的 ACK 或者包含在数据包中）就增加一个 credits。当 credits 不足时则等待。对于乱序的数据包，eRPC 会以丢包的情况看待。在丢包时，客户端会直接采取 go-back-N 的模式回滚，并会收回之前用于传输的 credit。最后基于 TIMELY 的拥塞控制，减少拥塞带来的数据包丢失，通过 RTT 的时间来评估网络的状态。在完全不拥塞的情况下，低于 Timely 的最低发送速率，不会更新发送速率；对于不拥塞的 session，发送的数据会直接传输，不会经过速率控制器；为每个数据包统计信息将会带来较大的消耗，通过对 RX 和 TX 的批量采样可以有效降低开销。

3.5 对于NIC进行调整以使其能适应分布式系统：

在云环境下，大多数的应用程序通常会在 I/O,数据传输以及数据存储方面产生巨大的成本和复制开销。因为大多数的情况下，这些分布式的计算都是与 NIC 没有联系的，因此就可以考虑到修改 NIC 来减少这些数据移动的开销。基于此，StRoM 是一种基于 FPGA 可编程的 NIC，支持卸载应用级内核。这些内核可用于直接从 NIC 执行内存访问操作，如遍历远程数据结构，以及在发送端或接收端对 RDMA 数据流进行过滤或聚合。StRoM 完全绕过 CPU，并扩展 RDMA 的语义，以支持多步骤数据访问操作和 RDMA 流的网络处理。我们通过四种不同内核扩展单边 RDMA 命令，展示了 StRoM 的多功能性和潜力：

1) 通过指针追踪遍历远程数据结构
2) 一致检索远程数据块
3) 通过将传入数据划分到不同内存区域或 CPU

核在 NIC 上进行数据洗牌

4) 数据流的基数估计

3.6 对于RDMA接口的扩展：PRISM

为了充分发挥网络加速的潜力，建立低延迟系统，必须超越基本的 RDMA 接口。最初设计用于支持并行超级计算应用程序的 RDMA 接口无法满足当今分布式系统的需求。我们展示了通过用一些额外的原语扩展接口，就可以完全使用远程操作来实现复杂的分布式应用程序，如复制存储。

我们在本文中的目标是识别 RDMA 接口的一组通用（即非应用程序特定）扩展，这允许分布式系统更好地利用 RDMA 硬件的低延迟和 CPU 卸载能力。我们的建议是 PRISM 接口。PRISM 扩展了 RDMA 读/写接口，增加了四个原语：间接、分配、增强的比较和交换以及操作链接。

3.6.1 间接作业：

许多 RDMA 应用程序需要遍历远程数据结构。这些结构用于许多目的：提供索引、支持可变长度数据等。目前，跟随指针需要额外的往返行程。PRISM 允许 READ、WRITE 和比较交换（CAS）操作采用间接参数。相反，这些操作的目标地址可以解释为指向实际目标的指针的地址。此外，可以从服务器端内存位置而不是从 RDMA 请求本身读取 WRITE 或 CAS 操作的数据。PRISM 中的间接操作重复使用现有的 RDMA 安全机制，这些机制确保远程客户端只能在已被授予访问权限的主机内存的区域上操作。要使用间接操作访问内存区域，客户端必须包含主机在首次向 NIC 注册区域时生成的 rkey。如果目标地址或目标地址指向的位置位于具有不同 rkey 的内存区域中（或根本未注册），则主机将拒绝该操作。

3.6.2 内存分配：

使用现有的 RDMA 接口修改数据结构尤其具有挑战性：对象必须写入固定的、预先分配的内存区域，这使得难以处理可变大小的对象或异地更新。PRISM 的内存分配原语中，PRISM 分配一个缓冲区并返回一个指向其位置的指针。服务器端进程将缓冲区队列与 NIC 进行注册（RDMA 术语为“发布”）。当 NIC 收到来自远程主机的 ALLOCATE 请求时，它会从此可用列表中弹出缓冲区，将提供的数据写入缓冲区，并以地址响应。PRISM 客户端可以分配缓冲区，写入缓冲区，然后在另一个数据结构中安装指向缓冲区的指针（通过 CAS）。

3.6.3 增强的比较与交换：

原子比较交换（CAS）是一种用于并行系统中数据更新的经典原语。RDMA 标准已经提供了原子 CAS 操作，但它受到高度限制：它进行一次相等比较，然后交换 64 位值。这不足以实现性能良好的应用程序。实际上，目前很少有应用程序使用 RDMA 原子操作，除了实现锁。虽然可以使用锁来实现任意复杂的原子操作，但这需要多次昂贵的往返并增加争用。PRISM 的 CAS 操作相对于其他 PRISM 操作是原子的。与现有的 RDMA 原子一样，它们在并发 CPU 操作方面不是有保证的原子。

3.6.4 操作链：

分布式应用程序通常需要执行一系列数据待定操作来读取或更新远程数据结构。例如，他们可能希望分配缓冲区，写入缓冲区，然后更新指针指向缓冲区。当前，每个操作必须返回到客户端，然后才能发出下一个操作 PRISM 提供了一个链接机制，允许这样的复合操作在一次往返中执行。

3.7 其他研究：

Cell 实现了一个 B 树，它需要更多的往返来执行读取（尽管缓存可能是有效的）。XStore 将树替换为学习的索引结构，以使用更少的 RDMA 读取进行搜索，但仍然需要间接搜索。PRISM 的间接原语可以帮助许多这些系统。在事务系统中，FaRM 使用 RDMA 读取数据，但使用 RPC 提交更新。DrTM 根据单侧操作构建基于锁的协议。PRISM 通过启用单面 OCC 协议来拓宽设计空间。

RDMA 扩展。实现者偶尔会为 RDMA 模型定义新的扩展。Mellanox 的扩展原子 API 允许 CAS 操作对更大操作数的部分与一个 8 字节值进行比较和交换。Snap 的软件 RDMA 堆栈支持间接操作以及模式搜索原语，这些在 Google 中使用，尽管没有公开详细描述。还提出了远内存数据结构的简单原语，包括间接寻址。

StRoM 和 RMC 建议允许应用程序分别在 StRoM 和多核网卡上安装自己的一次性原语。这与我们的目标是添加服务器端处理以避免额外的网络往返，并且确实支持更复杂的服务器端处理。然而，如第 2.3 节中所讨论的，运行自定义应用程序逻辑会带来部署挑战，并且支持一个小的通用原语库提供了更多的实现可能性。PRISM 证明这样的 API 是有用的。

HyperLoop 实现了与 PRISM 不同的链接形式：它表明，通过巧尽心思构建的 RDMA 请求，发送者可以导致接收者向第三台机器发起 RDMA 请求。这可以与 PRISM 结合使用来实现其他通信模式。RedN 进一步采用这种方法，使用自修改 RDMA 链作为图灵机，这是在现有硬件上实现 PRISM 原语的潜在方法。

4 总结和展望

本文从 RDMA 带来的优势以及分布式系统中 RDMA 的应用及改进阐述了对 RDMA 的研究。详细分析了当前 RDMA 的应用场景，描述了 RDMA 的四种原语以及 RDMA 的单边双边操作，分析了 RDMA 操作所带来的性能上的不同。RDMA 区别于传统的通信网络，因此如何充分利用 RDMA 的性能优势也是本文研究的一个重点。本文着重介绍了一种 RDMA 接口的拓展 PRISM，介绍了 PRISM 的扩展操作。

迈洛斯公司目前已经提供超过 200Gbs 的传输速率的 RDMA 网卡，传输延迟早已突破 1 μ s ，其迅速提升的网络传输性能，正促使 RDMA 技术被逐渐应用到数据中心．针对其扩展性问题，迈洛斯引入动态链接传输机制（dynamic connected transport, DCT）实现动态创建/销毁 QP 链 接，以保证 QP 数量维持稳定;Oracle 提出的 Sonoma 处理器则将网卡和 FPGA 集成到片上，能轻松支持数千个 QP 链路．为增强 RDMA 网卡设备的灵活性，迈洛斯还提供了可编程式 RDMA 网卡，用于协助用户进行应用数据传输加速、数据压缩/解压缩和其他功能性优化，为上层系统设计提供了更强的灵活性。RDMA 具有独特的数据传输模式和极高的数据传输性能，与此同时，其引入的扩展性等问题正在逐步解决与 RDMA 紧耦合的分布式系统软件设计将为大数据处理和存储带来新的机遇。

参 考 文 献

[1] Aleksandar Dragojević, Dushyanth Narayanan, Orion Hodson, and Miguel Castro. 2014. FaRM: Fast Remote Memory. In Proceedings of the 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI '14). USENIX, Seattle, WA, USA.

[2] David Sidler, Zeke Wang, Monica Chiosa, Amit Kulkarni, and Gustavo Alonso. 2020. StRoM: Smart Remote Memory. In Proceedings of the 15th ACM SIGOPS EuroSys (EuroSys '20). ACM, Heraklion, Crete, Greece.

[3] Emmanuel Amaro, Zhihong Luo, Amy Ousterhout, Arvind Krishnamurthy, Aurojit Panda, Sylvia Ratnasamy, and Scott Shenker. 2020. Remote Memory Calls. In Proceedings of the 16th Workshop on Hot Topics in Networks (HotNets '20). ACM, Chicago, IL, USA.

[4] Jiaxin Lin, Kiran Patel, Brent E. Stephens, Anirudh Sivaraman, and Aditya Akella. 2020. PANIC: A High-Performance Programmable NIC for Multi-tenant Networks. In Proceedings of the 14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20). USENIX, Banff, AL, Canada.

[5] Ming Liu, Tianyi Cui, Henry Schuh, Arvind Krishnamurthy, Simon Peter, and Karan Gupta. 2019. Offloading distributed applications onto SmartNICs using iPipe. In Proceedings of ACM SIGCOMM 2019. ACM, Beijing, China.

[6] Matthew Burke, Sowmya Dharanipragada, Shannon Joyner, Adriana Szekeres, Jacob Nelson, Irene Zhang, and Dan R. K. Ports. 2021. PRISM: Rethinking the RDMA Interface for Distributed Systems. In Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles (SOSP '21). Association for Computing Machinery, New York, NY, USA, 228–242.

[7] Daehyeok Kim, Amirsaman Memaripour, Anirudh Badam, Yibo Zhu, Hongqiang Harry Liu, Jitu Padhye, Shachar Raindel, Steven Swanson, Vyas Sekar, and Srinivasan Seshan. 2018. Hyperloop: group-based NIC-offloading to accelerate replicated transactions in multi-tenant storage systems. In Proceedings of ACM SIGCOMM 2018. ACM, Budapest, Hungary.