(19)国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 118192883 A
(43)申请公布日 2024.06.14

(21)申请号 202410144629.7
(22)申请日 2024.02.01
(71)申请人 中国人民解放军国防科技大学
地址 410073 湖南省长沙市开福区砚瓦池正街47号
(72)发明人 谢徐超 袁远 黎铁军 乔星涵 宋振龙 魏登萍 张根 邢建英 李志星 周桐庆
(74)专利代理机构 湖南兆弘专利事务所(普通合伙) 43008
专利代理师 谭武艺
(51)Int.Cl.
G06F 3/06(2006.01)
G06F 15/173(2006.01)
G06F 13/28(2006.01)
G06F 13/42(2006.01)
H04L 67/133(2022.01)

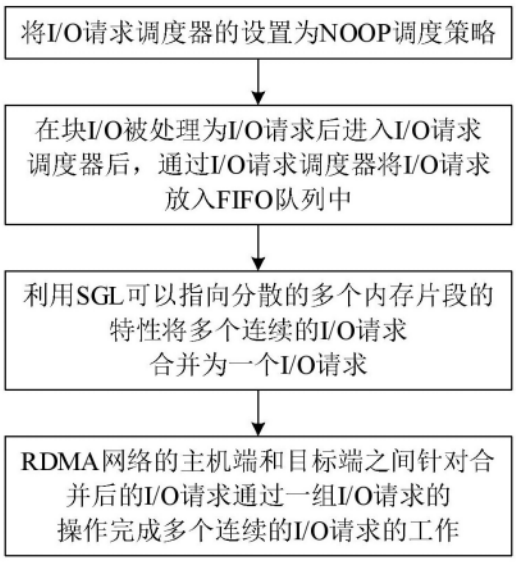
权利要求书2页 说明书8页 附图6页

(54)发明名称

一种面向远程存储访问的请求合并和调度方法及装置

(57)摘要

本发明公开了一种面向远程存储访问的请求合并和调度方法及装置,本发明包括利用SGL可以指向分散的多个内存片段的特性将多个连续的I/O请求合并为一个I/O请求,使得合并后的I/O请求中包含所述多个连续的I/O请求的散列聚集元素SGE,且RDMA网络的主机端和目标端之间针对合并后的I/O请求通过一组I/O请求的操作完成多个连续的I/O请求的工作。本发明旨在更好地基于NVMe over RDMA网络存储的特性对I/O请求进行调度,在通过计时器保证时效性的同时将多个I/O请求的SGL进行合并,以有效减少双边操作的数量、释放CPU算力并充分发挥NVMeoF远程存储的性能。



CN 118192883 A

1.一种面向远程存储访问的请求合并和调度方法,其特征在于,包括将I/O请求调度器的设置为NOOP调度策略,在块I/O被处理为I/O请求后进入I/O请求调度器后,通过I/O请求调度器将I/O请求放入FIFO队列中,并利用SGL可以指向分散的多个内存片段的特性将多个连续的I/O请求合并为一个I/O请求,使得合并后的I/O请求中包含所述多个连续的I/O请求的散列聚集元素SGE,且RDMA网络的主机端和目标端之间针对合并后的I/O请求通过一组I/O请求的操作完成多个连续的I/O请求的工作。

2.根据权利要求1所述的面向远程存储访问的请求合并和调度方法,其特征在于,所述将多个连续的I/O请求合并为一个I/O请求是指将多个连续的写请求合并为一个写请求或者多个连续的读请求合并为一个读请求。

3.根据权利要求2所述的面向远程存储访问的请求合并和调度方法,其特征在于,所述将多个连续的写请求合并为一个写请求是指将主机端和目标端之间本来需要2N次RDMA_SEND双边操作和N次RDMA_READ单边操作才能完成的N个连续的写请求合并为一个写请求,合并后的写请求中包含所述多个连续的写请求的散列聚集元素SGE,且RDMA网络的主机端和目标端之间通过合并后的写请求进行2次RDMA_SEND双边操作和1次RDMA_READ单边操作完成。

4.根据权利要求3所述的面向远程存储访问的请求合并和调度方法,其特征在于,所述主机端和目标端之间通过合并后的写请求进行2次RDMA_SEND双边操作和1次RDMA_READ单边操作完成包括:主机端向目标端发送RDMA_SEND命令,目标端在收到RDMA_SEND命令后向主机端发送RDMA_READ命令,使得主机端向目标端返回N个连续的写请求的数据,目标端在收到N个连续的写请求的数据后向主机端发送RDMA_SEND命令从而完成合并后的写请求的写操作。

5.根据权利要求2所述的面向远程存储访问的请求合并和调度方法,其特征在于,所述将多个连续的读请求合并为一个读请求是指将主机端和目标端之间本来需要2N次RDMA_SEND双边操作和N次RDMA_WRITE单边操作才能完成的N个连续的读请求合并为一个读请求,合并后的读请求中包含所述多个连续的读请求的散列聚集元素SGE,且RDMA网络的主机端和目标端之间通过合并后的读请求进行2次RDMA_SEND双边操作和1次RDMA_WRITE单边操作完成。

6.根据权利要求5所述的面向远程存储访问的请求合并和调度方法,其特征在于,所述主机端和目标端之间通过合并后的读请求进行2次RDMA_SEND双边操作和1次RDMA_WRITE单边操作完成包括:主机端向目标端发送RDMA_SEND命令,目标端在收到RDMA_SEND命令后向主机端发送RDMA_WRITE命令,使得主机端向目标端返回N个连续的读请求的数据,目标端在收到N个连续的读请求的数据后向主机端发送RDMA_SEND命令从而完成合并后的读请求的读操作。

7.根据权利要求1所述的面向远程存储访问的请求合并和调度方法,其特征在于,所述将多个连续的I/O请求合并为一个I/O请求时,包括基于传输能力的合并规则控制合并的I/O请求的数量,且每合并一个I/O请求,则计算合并后的I/O请求包含的散列聚集元素SGE数量,若合并后的I/O请求包含的散列聚集元素SGE数量等于依据NVMeoF远程存储网络的传输能力设置的阈值,则停止合并新的合并的I/O请求,如需继续合并I/O请求,则从下一个需要合并的I/O请求开始继续下一轮合并。

8.根据权利要求1所述的面向远程存储访问的请求合并和调度方法,其特征在于,所述将多个连续的I/O请求合并为一个I/O请求时,包括基于时延的合并规则控制合并的I/O请求的数量,且在第一个I/O请求进入RDMA驱动中的Nvme_rdma_queue_rq函数后启动计时器,且每合并一个I/O请求之前,则判断计时器是否超过,若未超时则允许该I/O请求进行合并,否则不允许该I/O请求进行合并。

9.一种面向远程存储访问的请求合并和调度装置,包括相互连接的微处理器和存储器,其特征在于,所述微处理器被编程或配置以执行权利要求1~8中任意一项所述面向远程存储访问的请求合并和调度方法。

10.一种计算机可读存储介质,所述计算机可读存储介质中存储有计算机程序,其特征在于,所述计算机程序用于被微处理器编程或配置以执行权利要求1~8中任意一项所述面向远程存储访问的请求合并和调度方法。

一种面向远程存储访问的请求合并和调度方法及装置

技术领域

[0001] 本发明涉及网络存储技术领域,具体涉及一种面向远程存储访问的请求合并和调度方法及装置。

背景技术

[0002] NVMe(Non-Volatile Memory Express)是一种针对PCIe SSD重新定义的主机控制器接口规范。它是为了充分利用固态存储的高速性能而开发的,通过优化命令队列和减少I/O操作的延迟来提供比传统的SATA和SAS接口更高的性能。NVMe的一个显著优势是它的低延迟和高IOPS(每秒进行读写操作的次数)性能。最多可支持64K 由SQ(提交队列)与CQ(完成队列)组成的队列组(Queue Pair, QP),每个队列的最大深度为64K。在驱动层,NVMe与Linux内核块层的blk-mq(linux块设备驱动)机制协同设计,多核CPU的多线程I/O访问不会对SSD请求队列产生锁竞争问题,并避免了单一请求队列造成的性能瓶颈。NVMe还优化了驱动器与系统之间的通信方式。它减少了驱动器操作所需的CPU周期数量,这意味着更高的系统效率和更低的功耗。此外,NVMe协议还包括对电源管理的支持,这使得它在节能和延长设备寿命方面更加出色。随着固态驱动器技术的发展和价格的下降,NVMe已成为高性能存储解决方案的首选,广泛应用于各种环境中,从个人计算机到数据中心。它特别适合需要高速数据访问的应用,如大数据分析、高性能计算、实时数据处理和游戏。随着技术的不断进步,NVMe将继续推动存储技术的发展,提供更快、更可靠的数据存储和访问解决方案。

[0003] NVMe over Fabrics(NVMeoF)是一种网络协议,旨在将NVMe技术的高性能和低延迟特性扩展到更广泛的网络环境中。它允许通过网络连接的设备以接近本地性能的方式访问远程NVMe存储设备。这种技术的核心在于将NVMe命令和数据封装在网络传输协议中,然后在网络上进行传输,从而实现远程存储访问。NVMeoF利用现有的网络技术,如光纤通道、以太网和InfiniBand,通过它们来传输NVMe命令和数据。特别是当使用支持远程直接内存访问(RDMA)的网络技术时,NVMeoF能够实现极低的延迟和高吞吐量,这对于性能敏感的应用非常重要。RDMA允许数据直接在服务器和存储设备之间的内存中移动,减少了CPU干预和额外的内存复制操作,从而提高了效率。NVMeoF的引入为数据中心和云基础设施提供了显著的性能提升。它使得数据中心可以更有效地利用NVMe存储设备,提供更快的数据访问速度,更好的扩展性,以及更高的资源利用率。此外,NVMeoF也支持更灵活的存储架构,允许构建大规模、分布式的存储系统,同时保持对存储的快速访问。随着数据中心和企业对于高速、高效存储解决方案的需求不断增长,NVMeoF成为了一个关键技术,帮助他们应对大数据、实时分析和高性能计算等挑战。其在提高存储网络性能和效率方面的能力,使其成为现代数据中心存储架构的重要组成部分。

[0004] RDMA(Remote Direct Memory Access)是一种高效的网络通信技术,允许一台计算机直接访问另一台计算机上的内存,无需通过远程系统的CPU或操作系统,该技术显著减少了网络通信过程中的CPU负载和系统延迟,提高了数据传输的效率。在传统的网络通信中,数据必须经过操作系统的多个层次,涉及到CPU的参与来处理数据包、执行协议栈操作

和在系统之间复制数据。这个过程不仅增加了延迟,还占用了宝贵的CPU资源。相比之下,RDMA允许网络适配器直接读取或写入远程主机的内存,而无需远程主机的CPU介入。这是通过在参与的主机上预先设置好的内存区域(注册内存区域)来实现的。这些内存区域的地址和访问权限被共享给网络中的其他主机。当一个RDMA传输发生时,数据直接从发送方的内存移动到接收方的内存,绕过了操作系统的传统网络堆栈。RDMA的这种直接内存访问机制带来了几个显著优势。首先,它大大减少了数据传输过程中的延迟,因为它消除了操作系统中网络堆栈的处理时间。其次,由于CPU不再参与实际的数据传输,因此可以显著减少CPU的使用率,从而提高系统的整体性能和效率。此外,由于减少了数据在主机内部的复制操作,所以还能提高数据传输的吞吐量。由于这些特点,RDMA在高性能计算、大数据处理、云计算以及那些对网络性能有高要求的场景中尤其受到青睐。它在现代数据中心和企业级存储解决方案中发挥着关键作用,特别是在需要处理大量数据且对延迟和吞吐量有严格要求的应用中。

[0005] 在RDMA中,散列聚集列表(Scatter Gather List,简称SGL)是一种关键的数据结构,用于处理分散在内存中的多个非连续数据块的高效传输。由于RDMA的核心特性是能够在网络参与者之间直接进行内存到内存的数据传输,因此SGL在RDMA操作中扮演着重要角色。在RDMA通信过程中,数据可能不会总是存储在一个连续的内存块中。SGL允许RDMA操作指定一个由多个内存片段组成的列表,每个片段有自己的地址和长度。通过SGL,RDMA能够一次性地从这些分散的内存位置读取数据或向它们写入数据,而无需事先将数据复制到一个连续的内存区域。图1解释了SGL和WR链表的对应关系,其中ibv_send_wr为RDMA的工作请求项,wr_id为该请求的编号,sg_list*为指向分散聚合表(SGL)的指针,num_sge为该SGL中包含的分散聚合项(SGE)数量,next*为指向下一个工作请求项的指针,ibv_sge为由分散聚合项构成的数组,addr为其地址,length表示该项指向的数据段的长度,N1Bytes～N3Bytes为分散在内存中的多个数据段。从图1可以看到,WR链表的每一个结点都包含了一个SGL,SGL是一个数组,包含一个或多个散列聚集元素SGE,散列聚集元素SGE指向需要发送数据的换成缓存Buffer。

[0006] 在RDMA网络中,使用SGL进行数据传输的过程如图2所示。该SGL数组包含了3个SGE,长度分别为N1, N2, N3字节。从图2可以看到,这3个缓存buffer并不连续,它们分散在内存中的各个地方。RDMA硬件读取到SGL后,进行聚合操作,于是在RDMA硬件的Wire上看到的就是N3+N2+N1个连续的字节。这种方法的优势在于显著提高了数据传输的效率和灵活性。它减少了数据复制和内存重组的需求,从而减轻了CPU的负担,并降低了延迟。对于高性能计算和大规模数据处理应用来说,这是极其重要的,因为它们通常涉及到大量的、分散的数据集。总的来说,SGL在RDMA中的应用强化了RDMA的核心优势,即提供高效率、低延迟的直接内存访问,这对于现代高性能网络通信和数据中心技术尤其关键。

[0007] RDMA_READ 和 RDMA_WRITE 在RDMA通信中提供了高效的数据读写能力。RDMA_READ 用于从远程内存读取数据,而 RDMA_WRITE 则用于向远程内存写入数据。这两种操作都通过绕过传统的网络堆栈和操作系统内核,直接在内存之间进行数据传输,大大减少了延迟和CPU开销,从而提高了数据传输的效率。这在需要快速数据交换的高性能计算和数据中心环境中尤其重要。

[0008] 如图3所示,RDMA_WRITE操作允许Target端直接向Host端的内存写入数据。RDMA_

WRITE操作和RDMA_SEND操作应使用同一个RDMA队列对(QP),以确保主机Host和目标Target之间通信的一致性和有序性,其详细流程如下:1、Host端通过RDMA_SEND操作向控制器传输命令封包。该封包包含或指向用于数据传输所需的SGL(Scatter-Gather List)。2、Target端使用RDMA_WRITE操作将数据传输到Host端。每个RDMA_WRITE都与一个键控的Host端内存缓冲区(SGL条目)和一个或多个本地内存缓冲区关联。这确保了数据直接从Target端的内存传输到Host端内存的指定位置。3、传输完成后,Target使用RDMA_SEND或RDMA_SEND_INVALIDATE操作将响应封包发送回主机,后者可使内存键失效。

[0009] 如图4所示,RDMA_READ操作允许Target端直接从Host端的内存读出数据。RDMA_READ操作和RDMA_SEND操作应使用同一个RDMA队列对(QP),以确保Host和Target之间通信的一致性和有序性,其详细流程如下:1、Host端通过RDMA_SEND操作向控制器传输命令封包。该封包包含或指向用于数据传输所需的SGL(Scatter-Gather List),此封包可能直接包含数据,这些数据通过封包内SGL中的偏移地址引用。2、对于驻留在Host端的命令数据,Target端使用RDMA_READ操作将数据从Host端内存传输到Target端。每个RDMA_READ都与一个键控的远程主机内存缓冲区(SGL条目)和一个或多个本地内存缓冲区关联。3、传输完成后,Target使用RDMA_SEND或RDMA_SEND_INVALIDATE操作将响应封包发送回主机,后者可使内存键失效。在上述过程中可以看到在进行RDMA_READ和RDMA_WRITE操作之前需要通过RDMA_SEND操作发送相应的命令和响应,在RDMA网络中,SEND/RECV 操作和 READ/WRITE 操作都用于在网络上进行数据传输,但它们在操作方式和开销上存在显著差异:END/RECV 是一种双边操作。发送方使用 SEND 操作发送数据,接收方必须预先准备一个 RECV 队列来接收这些数据。这种操作涉及发送方和接收方的CPU。接收方的CPU需要处理接收队列,管理接收缓冲区。数据首先被复制到发送方的内存中,然后通过网络发送到接收方的内存。由于涉及CPU处理和内存复制,SEND/RECV 操作相比于 READ/WRITE 通常有更高的延迟和较低的吞吐量。READ/WRITE是一种单边操作,允许直接从一个主机的内存向另一个主机的内存读取或写入数据,无需接收方CPU的介入。这些操作直接在内存之间进行数据传输,减少了额外的内存复制步骤,提高了效率。READ/WRITE 操作通常提供更低的延迟和更高的吞吐量,特别是在大量数据传输时更为显著。

[0010] I/O请求调度是操作系统中对存储设备的读写请求进行管理和优化的过程。其关键在于决定哪些I/O操作首先被执行,以及如何有效地执行它们,以提高存储设备的性能和整体系统的效率。在处理I/O请求时,调度器需要考虑诸如请求的优先级、数据的物理位置以及存储设备的性能特性等因素。在I/O请求调度中,操作系统会根据一定的算法或策略来组织和排序待处理的I/O请求。例如,它可能会对请求进行排序,以减少机械硬盘上读写头的移动,或者合并多个相邻的小请求以减少对固态硬盘的操作次数。这种优化旨在减少对存储设备的访问延迟,提高数据吞吐量,以及在多任务环境中公平地分配资源。但是,现有的I/O请求调度策略主要基于本地存储器进行优化,没有充分考虑在网络存储的场景下网络开销的影响,无法有效提升其性能。

[0011] Host端的I/O流程如图5所示,bio(Block I/O,块I/O)被处理成request(I/O请求)进入I/O请求调度器,I/O请求调度器根据一定的算法或策略来组织和排序待处理的I/O请求,并下发给RDMA驱动中的Nvme_rdma_queue_rq函数进行处理。Nvme_rdma_queue_rq函数负责根据I/O 请求构建相应的 RDMA 命令,并确保这些命令能够被正确地发送和处理,以

完成高效的数据存取操作。由于现有的I/O请求调度策略主要基于本地存储器进行优化,没有充分考虑在网络存储的场景下网络开销的影响,无法有效提升其性能。

发明内容

[0012] 本发明要解决的技术问题:针对现有技术的上述问题,提供一种面向远程存储访问的请求合并和调度方法及装置,本发明旨在更好地基于NVMe over RDMA网络存储的特性对I/O请求进行调度,在通过计时器保证时效性的同时将多个I/O请求的SGL进行合并,以有效减少双边操作的数量、释放CPU算力并充分发挥NVMeoF远程存储的性能。

[0013] 为了解决上述技术问题,本发明采用的技术方案为:

一种面向远程存储访问的请求合并和调度方法,包括将I/O请求调度器的设置为NOOP调度策略,在块I/O被处理为I/O请求后进入I/O请求调度器后,通过I/O请求调度器将I/O请求放入FIFO队列中,并利用SGL可以指向分散的多个内存片段的特性将多个连续的I/O请求合并为一个I/O请求,使得合并后的I/O请求中包含所述多个连续的I/O请求的散列聚集元素SGE,且RDMA网络的主机端和目标端之间针对合并后的I/O请求通过一组I/O请求的操作完成多个连续的I/O请求的工作。

[0014] 可选地,所述将多个连续的I/O请求合并为一个I/O请求是指将多个连续的写请求合并为一个写请求或者多个连续的读请求合并为一个读请求。

[0015] 可选地,所述将多个连续的写请求合并为一个写请求是指将主机端和目标端之间本来需要2N次RDMA_SEND双边操作和N次RDMA_READ单边操作才能完成的N个连续的写请求合并为一个写请求,合并后的写请求中包含所述多个连续的写请求的散列聚集元素SGE,且RDMA网络的主机端和目标端之间通过合并后的写请求进行2次RDMA_SEND双边操作和1次RDMA_READ单边操作完成。

[0016] 可选地,所述主机端和目标端之间通过合并后的写请求进行2次RDMA_SEND双边操作和1次RDMA_READ单边操作完成包括:主机端向目标端发送RDMA_SEND命令,目标端在收到RDMA_SEND命令后向主机端发送RDMA_READ命令,使得主机端向目标端返回N个连续的写请求的数据,目标端在收到N个连续的写请求的数据后向主机端发送RDMA_SEND命令从而完成合并后的写请求的写操作。

[0017] 可选地,所述将多个连续的读请求合并为一个读请求是指将主机端和目标端之间本来需要2N次RDMA_SEND双边操作和N次RDMA_WRITE单边操作才能完成的N个连续的读请求合并为一个读请求,合并后的读请求中包含所述多个连续的读请求的散列聚集元素SGE,且RDMA网络的主机端和目标端之间通过合并后的读请求进行2次RDMA_SEND双边操作和1次RDMA_WRITE单边操作完成。

[0018] 可选地,所述主机端和目标端之间通过合并后的读请求进行2次RDMA_SEND双边操作和1次RDMA_WRITE单边操作完成包括:主机端向目标端发送RDMA_SEND命令,目标端在收到RDMA_SEND命令后向主机端发送RDMA_WRITE命令,使得主机端向目标端返回N个连续的读请求的数据,目标端在收到N个连续的读请求的数据后向主机端发送RDMA_SEND命令从而完成合并后的读请求的读操作。

[0019] 可选地,所述将多个连续的I/O请求合并为一个I/O请求时,包括基于传输能力的合并规则控制合并的I/O请求的数量,且每合并一个I/O请求,则计算合并后的I/O请求包含

的散列聚集元素SGE数量,若合并后的I/O请求包含的散列聚集元素SGE数量等于依据NVMeoF远程存储网络的传输能力设置的阈值,则停止合并新的合并的I/O请求,如需继续合并I/O请求,则从下一个需要合并的I/O请求开始继续下一轮合并。

[0020] 可选地,所述将多个连续的I/O请求合并为一个I/O请求时,包括基于时延的合并规则控制合并的I/O请求的数量,且在第一个I/O请求进入RDMA驱动中的Nvme_rdma_queue_rq函数后启动计时器,且每合并一个I/O请求之前,则判断计时器是否超过,若未超时则允许该I/O请求进行合并,否则不允许该I/O请求进行合并。

[0021] 此外,本发明还提供一种面向远程存储访问的请求合并和调度装置,包括相互连接的微处理器和存储器,所述微处理器被编程或配置以执行所述面向远程存储访问的请求合并和调度方法。

[0022] 此外,本发明还提供一种计算机可读存储介质,所述计算机可读存储介质中存储有计算机程序,所述计算机程序用于被微处理器编程或配置以执行所述面向远程存储访问的请求合并和调度方法。

[0023] 和现有技术相比,本发明主要具有下述优点:本发明充分考虑将I/O请求调度与RDMA传输流程协同设计,通过基于NVMe over RDMA网络存储的特性对I/O请求进行调度,在通过计时器保证时效性的同时将多个I/O请求的SGL进行合并,以有效减少双边操作的数量、释放CPU算力并充分发挥NVMeoF远程存储的性能。

附图说明

[0024] 图1为现有技术中SGL组织方式的示意图。

[0025] 图2为现有技术中SGL传输过程的示意图。

[0026] 图3为现有技术中RDMA_WRITE过程的交互示意图。

[0027] 图4为现有技术中RDMA_READ过程的交互示意图。

[0028] 图5为现有技术中RDMA网络的主机端的I/O流程示意图。

[0029] 图6为本发明实施例方法的基本流程示意图。

[0030] 图7为现有技术中RDMA网络的写请求的处理过程。

[0031] 图8为本发明实施例方法中RDMA网络的写请求的处理过程。

[0032] 图9为现有技术中RDMA网络的读请求的处理过程。

[0033] 图10为本发明实施例方法中RDMA网络的读请求的处理过程。

[0034] 图11为本发明实施例方法中由SGE数目触发的发送过程示意图。

[0035] 图12为本发明实施例方法中由计时器触发的发送过程示意图。

具体实施方式

[0036] 如图6所示,本实施例面向远程存储访问的请求合并和调度方法包括将I/O请求调度器的设置为NOOP调度策略,在块I/O被处理为I/O请求后进入I/O请求调度器后,通过I/O请求调度器将I/O请求放入FIFO队列中,并利用SGL可以指向分散的多个内存片段的特性将多个连续的I/O请求合并为一个I/O请求,使得合并后的I/O请求中包含所述多个连续的I/O请求的散列聚集元素SGE,且RDMA网络的主机端和目标端之间针对合并后的I/O请求通过一组I/O请求的操作完成多个连续的I/O请求的工作。

[0037] 本实施例中将多个连续的I/O请求合并为一个I/O请求是指将多个连续的写请求合并为一个写请求或者多个连续的读请求合并为一个读请求。

[0038] 图7为现有技术中RDMA网络的写请求的处理过程。对于Host端的多个写请求,位于NVMe over RDMA驱动层的queue_rq根据上层请求依次准备数据,构建请求命令,通过RDMA_SEND操作将该命令发送至Target端。接收到命令后,对于驻留在Host端的命令数据,Target端使用RDMA_READ操作将数据从Host端内存传输到Target端。传输完成后,Target使用RDMA_SEND或RDMA_SEND_INVALIDATE操作将响应封包发送回主机,后者可使内存键失效。以图7中所示场景为例,对于3个写请求的传输处理过程,对于每个请求都需要进行2次RDMA_SEND双边操作和1次RDMA_READ单边操作,处理连续的3个写请求共需要进行6次双边操作和3次单边操作。

[0039] 图8为本实施例方法中RDMA网络的写请求的处理过程。如图8所示,本实施例中将多个连续的写请求合并为一个写请求是指将主机端和目标端之间本来需要2N次RDMA_SEND双边操作和N次RDMA_READ单边操作才能完成的N个连续的写请求合并为一个写请求,合并后的写请求中包含所述多个连续的写请求的散列聚集元素SGE,且RDMA网络的主机端和目标端之间通过合并后的写请求进行2次RDMA_SEND双边操作和1次RDMA_READ单边操作完成。图8中所示在nvme_rdma_queue_rq函数中,利用SGL可以指向分散在内存中的多个非连续数据块的特性,多个连续的写请求被合并为一个写请求,处理该写请求仅需进行2次RDMA_SEND双边操作和1次RDMA_READ单边操作,从而提升了写性能。

[0040] 参见图8,主机端和目标端之间通过合并后的写请求进行2次RDMA_SEND双边操作和1次RDMA_READ单边操作完成包括:主机端向目标端发送RDMA_SEND命令,目标端在收到RDMA_SEND命令后向主机端发送RDMA_READ命令,使得主机端向目标端返回N个连续的写请求的数据,目标端在收到N个连续的写请求的数据后向主机端发送RDMA_SEND命令从而完成合并后的写请求的写操作。

[0041] 图9为现有技术中RDMA网络的读请求的处理过程。对于Host端的多个读请求,位于NVMe over RDMA驱动层的queue_rq根据上层请求依次准备内存区域,构建请求命令,通过RDMA_SEND操作将该命令发送至Target端。接收到命令后,Target端使用RDMA_WRITE操作将数数据直接从Target端的内存传输到Host端内存的指定位置。传输完成后,Target端使用RDMA_SEND操作或RDMA_SEND_INVALIDATE操作将响应封包发送回主机,后者可使内存键失效。以图9中所示场景为例,对于3个读请求的传输处理过程,对于每个请求都需要进行2次RDMA_SEND双边操作和1次RDMA_WRITE单边操作,处理连续的3个写请求共需要进行6次双边操作和3次单边操作。

[0042] 图10为本发明方法中RDMA网络的读请求的处理过程。如图10所示,本实施例中将多个连续的读请求合并为一个读请求是指将主机端和目标端之间本来需要2N次RDMA_SEND双边操作和N次RDMA_WRITE单边操作才能完成的N个连续的读请求合并为一个读请求,合并后的读请求中包含所述多个连续的读请求的散列聚集元素SGE,且RDMA网络的主机端和目标端之间通过合并后的读请求进行2次RDMA_SEND双边操作和1次RDMA_WRITE单边操作完成。图10所示在nvme_rdma_queue_rq函数中,利用SGL可以指向分散在内存中的多个非连续数据块的特性,多个连续的读请求被合并为一个读请求,处理该读请求仅需进行2次RDMA_SEND双边操作和1次RDMA_WRITE单边操作,从而提升了读性能。

[0043] 参见图10,主机端和目标端之间通过合并后的读请求进行2次RDMA_SEND双边操作和1次RDMA_WRITE单边操作完成包括:主机端向目标端发送RDMA_SEND命令,目标端在收到RDMA_SEND命令后向主机端发送RDMA_WRITE命令,使得主机端向目标端返回N个连续的读请求的数据,目标端在收到N个连续的读请求的数据后向主机端发送RDMA_SEND命令从而完成合并后的读请求的读操作。

[0044] 当请求到达NVMe over RDMA驱动中的queue_rq时,根据RDMA网络情况(例如RDMA网络允许的一个请求内可携带的最大SGE数量,和网络中路由器交换机允许的数据包大小等)对其进行调度,利用SGL可以指向分散的多个内存片段的特性,将命令封包内包含的SGL进行合并,在控制延迟在合理范围内的前提下,降低双边操作的次数,以减少对计算资源的占用,并在一定程度上提升吞吐量。在具体的传输处理过程中,如何进行请求的合并也是调度方法的关键部分。若合并后的SGE数量超过RDMA网络允许的SGE数目或合并后的请求大小超过了RDMA网络规定的命令封装大小会导致错误。若为了合并请求进行的等待时间过长则会导致不可接受的延迟增加。因此,本实施例从基于传输能力和基于时延两个方面设计了请求的合并规则:一方面,本实施例将多个连续的I/O请求合并为一个I/O请求时,包括基于传输能力的合并规则控制合并的I/O请求的数量,且每合并一个I/O请求,则计算合并后的I/O请求包含的散列聚集元素SGE数量,若合并后的I/O请求包含的散列聚集元素SGE数量等于依据NVMeoF远程存储网络的传输能力设置的阈值,则停止合并新的合并的I/O请求,如需继续合并I/O请求,则从下一个需要合并的I/O请求开始继续下一轮合并。该阈值可为RDMA网络允许的一个请求内可携带的最大SGE数量和网络中路由器交换机允许的数据包大小两者中的较小值;另一方面,本实施例将多个连续的I/O请求合并为一个I/O请求时,包括基于时延的合并规则控制合并的I/O请求的数量,且在第一个I/O请求进入RDMA驱动中的Nvme_rdma_queue_rq函数后启动计时器,且每合并一个I/O请求之前,则判断计时器是否超过,若未超时则允许该I/O请求进行合并,否则不允许该I/O请求进行合并。

[0045] 图11以读请求的发送过程为例说明了由SGE数目触发的发送过程。该示例中假设RDMA网络允许一个请求内的最大SGE数量为3,且该网络内所有的交换机路由器均支持此大小的数据包,因此将合并SGE数量的阈值设为3。在Host端的nvme_rdma_queue_rq函数中为每待发送的请求设置SGE计数器和计时器。SGE计数器统计该请求内SGL包含的SGE数量,在新到来的请求包含的SGE被合并至该请求时,更新SGE计数器的值。计时器则用来限定该请求进入队列后的最大等待时间。在图10中,第一个读请求进入队列时计时器启动,SGE计数器值更新。在计时器超时前,SGE计数器达到设定阈值,触发发送机制,该请求被发送。图12以读请求的发送过程为例说明了计时器触发的发送过程,该请求的计时器在该请求到达nvme_rdma_queue_rq函数时启动,由于第二个请求包含的SGE被合并之后直至计时器超时没有后续请求到来。虽然SGE计数器未达到设定阈值,但计时器超时触发发送机制,该请求被发送。计时器的阈值一般根据具体应用环境能接受的时延进行设置,对于时延敏感的应用环境可以设置的低一些以保证时延,对于时延不敏感的应用环境可以设置的高一些以保证将尽可能多的SGE进行合并发送。

[0046] 综上所述,本实施例面向远程存储访问的请求合并和调度方法中将I/O调度器设置为NOOP,该调度器将IO请求放入到一个FIFO队列中,然后逐个执行这些IO请求。该调度器不会重新组织IO请求顺序,对于一些在磁盘上连续的IO请求,会适当做一些合并。而且当请

求到达NVMe over RDMA驱动中的nvme_rdma_queue_rq函数时,本实施例面向远程存储访问的请求合并和调度方法根据RDMA网络情况(RDMA网络允许的一个请求内可携带的最大SGE数量,和网络中路由器交换机允许的数据包大小)对其进行调度,利用SGL可以指向分散的多个内存片段的特性,将命令封包内包含的SGL进行合并,在控制延迟在合理范围内的前提下,降低双边操作的次数,以减少对计算资源的占用,并在一定程度上提升吞吐量。

[0047] 此外,本实施例还提供一种面向远程存储访问的请求合并和调度装置,包括相互连接的微处理器和存储器,所述微处理器被编程或配置以执行所述面向远程存储访问的请求合并和调度方法。

[0048] 此外,本实施例还提供一种计算机可读存储介质,所述计算机可读存储介质中存储有计算机程序,所述计算机程序用于被微处理器编程或配置以执行所述面向远程存储访问的请求合并和调度方法。

[0049] 本领域内的技术人员应明白,本申请的实施例可提供为方法、系统、或计算机程序产品。因此,本申请可采用完全硬件实施例、完全软件实施例、或结合软件和硬件方面的实施例的形式。而且,本申请可采用在一个或多个其中包含有计算机可用程序代码的计算机可读存储介质(包括但不限于磁盘存储器、CD-ROM、光学存储器等)上实施的计算机程序产品的形式。本申请是参照根据本申请实施例的方法、设备(系统)、和计算机程序产品的流程图和/或方框图来描述的。应理解可由计算机程序指令实现流程图和/或方框图中的每一流程和/或方框、以及流程图和/或方框图中的流程和/或方框的结合。可提供这些计算机程序指令到通用计算机、专用计算机、嵌入式处理机或其他可编程数据处理设备的处理器以产生一个机器,使得通过计算机或其他可编程数据处理设备的处理器执行的指令产生用于实现在流程图一个流程或多个流程和/或方框图一个方框或多个方框中指定的功能的装置。这些计算机程序指令也可存储在能引导计算机或其他可编程数据处理设备以特定方式工作的计算机可读存储器中,使得存储在该计算机可读存储器中的指令产生包括指令装置的制造品,该指令装置实现在流程图一个流程或多个流程和/或方框图一个方框或多个方框中指定的功能。这些计算机程序指令也可装载到计算机或其他可编程数据处理设备上,使得在计算机或其他可编程设备上执行一系列操作步骤以产生计算机实现的处理,从而在计算机或其他可编程设备上执行的指令提供用于实现在流程图一个流程或多个流程和/或方框图一个方框或多个方框中指定的功能的步骤。

[0050] 以上所述仅是本发明的优选实施方式,本发明的保护范围并不仅局限于上述实施例,凡属于本发明思路下的技术方案均属于本发明的保护范围。应当指出,对于本技术领域的普通技术人员来说,在不脱离本发明原理前提下的若干改进和润饰,这些改进和润饰也应视为本发明的保护范围。

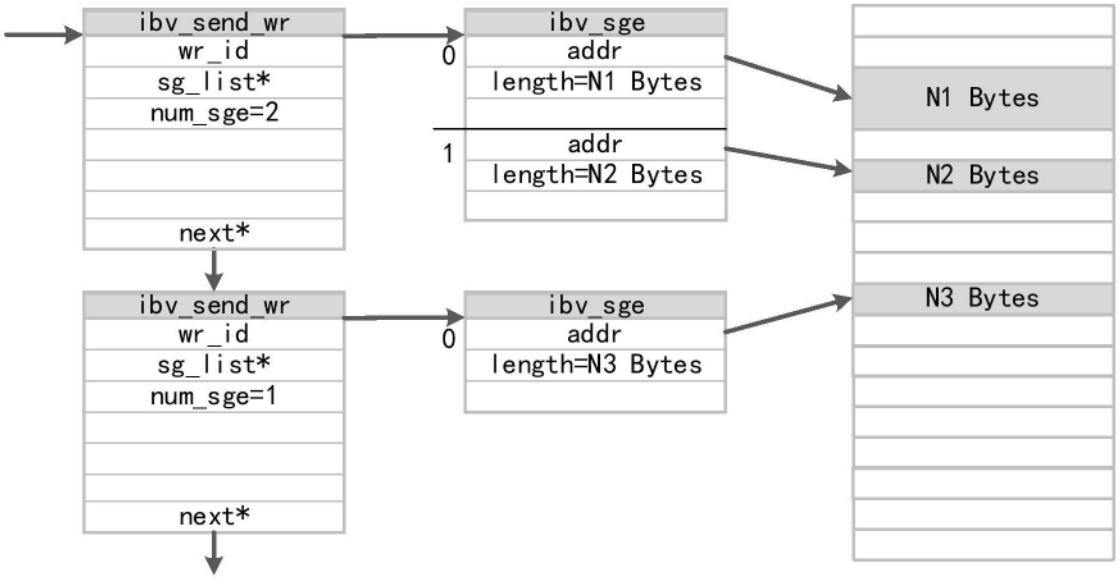

ibv_send_wr
wr_id
sg_list*
num_sge=2
next*
ibv_send_wr
wr_id
sg_list*
num_sge=1
next*
ibv_sge
0
addr
length=N1 Bytes
1
addr
length=N2 Bytes
ibv_sge
0
addr
length=N3 Bytes
N1 Bytes
N2 Bytes
N3 Bytes

图1

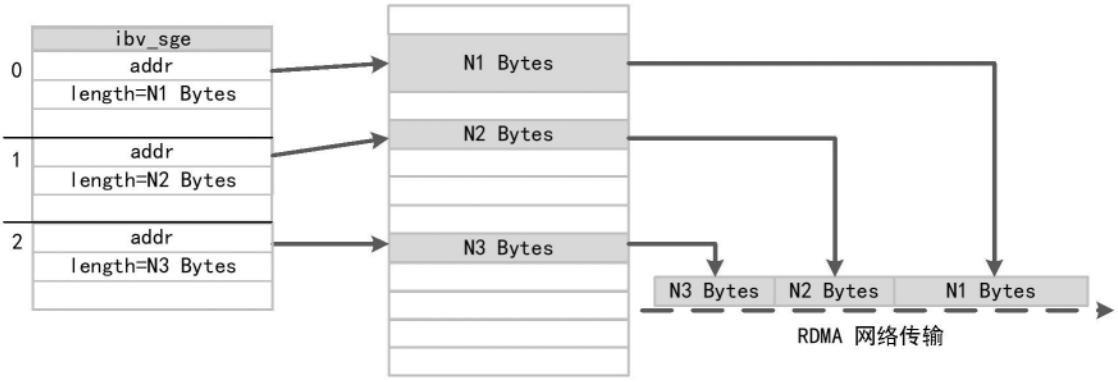

ibv_sge
0
addr
length=N1 Bytes
1
addr
length=N2 Bytes
2
addr
length=N3 Bytes
N1 Bytes
N2 Bytes
N3 Bytes
N3 Bytes
N2 Bytes
N1 Bytes
RDMA 网络传输

图2

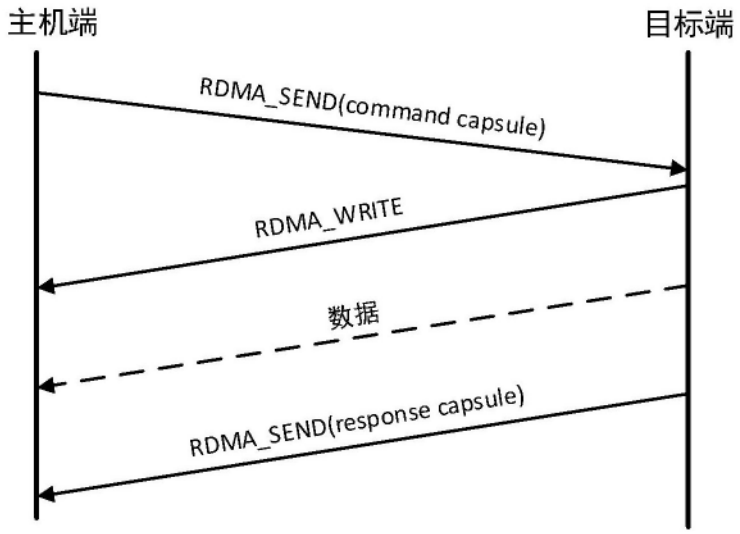

主机端
目标端
RDMA_SEND(command capsule)
RDMA_WRITE
数据
RDMA_SEND(response capsule)

图3

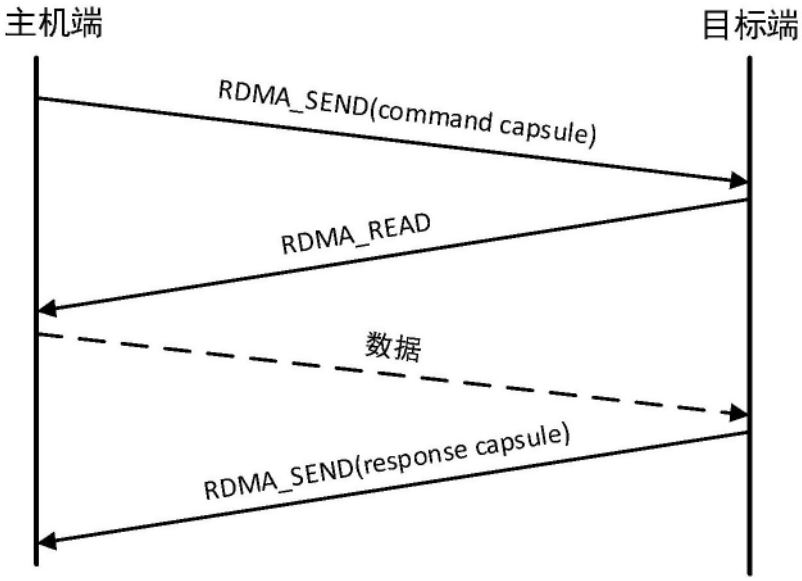

主机端
目标端
RDMA_SEND(command capsule)
RDMA_READ
数据
RDMA_SEND(response capsule)

图4

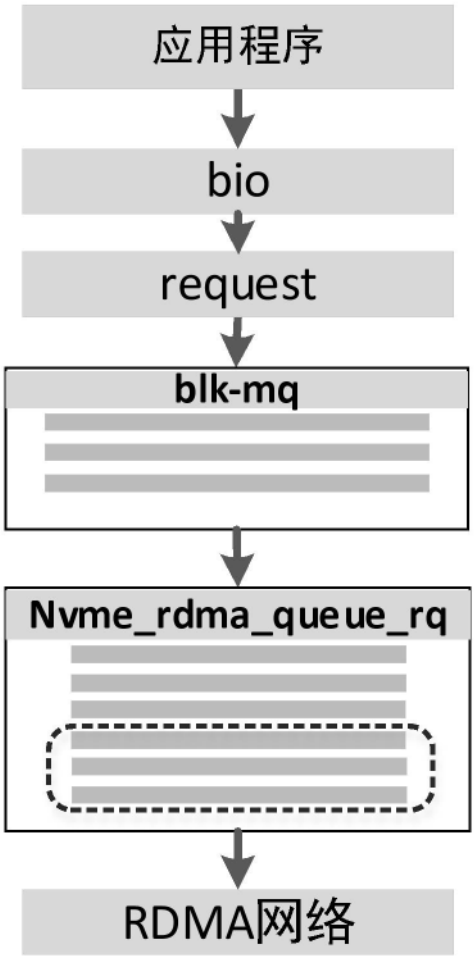


图5

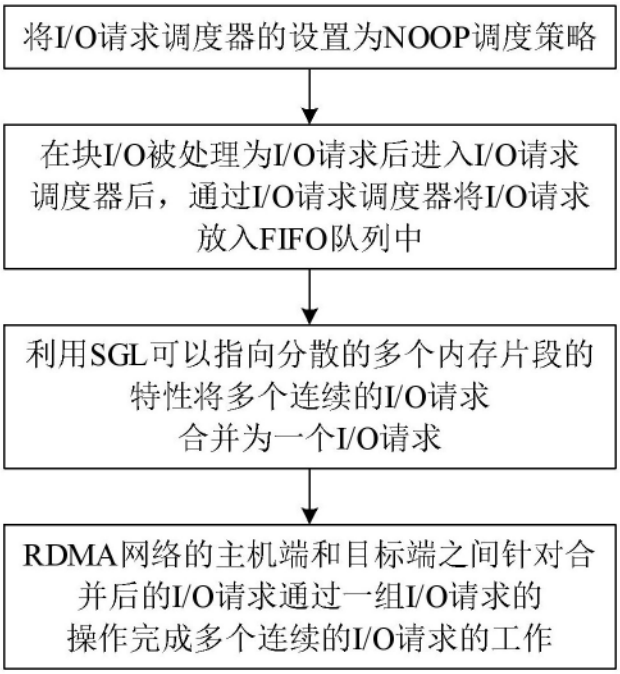

将I/O请求调度器的设置为NOOP调度策略
在块I/O被处理为I/O请求后进入I/O请求调度器后，通过I/O请求调度器将I/O请求放入FIFO队列中
利用SGL可以指向分散的多个内存片段的特性将多个连续的I/O请求合并为一个I/O请求
RDMA网络的主机端和目标端之间针对合并后的I/O请求通过一组I/O请求的操作完成多个连续的I/O请求的工作

图6

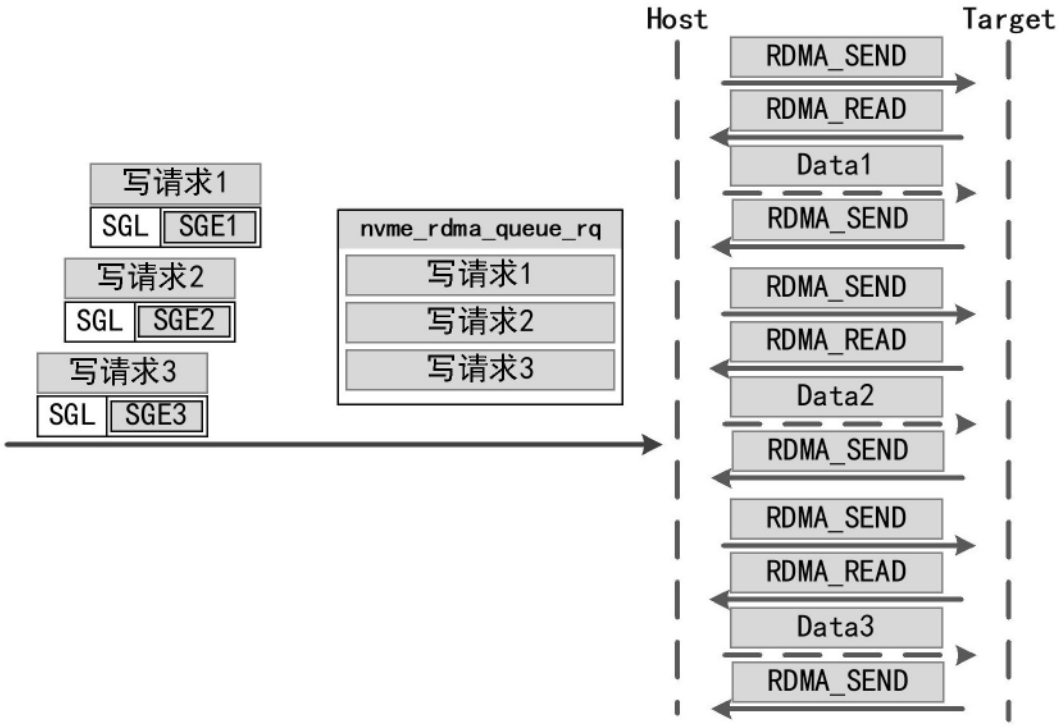

Host
Target
写请求1
SGL
SGE1
写请求2
SGL
SGE2
写请求3
SGL
SGE3
nvme_rdma_queue_rq
写请求1
写请求2
写请求3
RDMA_SEND
RDMA_READ
Data1
RDMA_SEND
RDMA_SEND
RDMA_READ
Data2
RDMA_SEND
RDMA_SEND
RDMA_READ
Data3
RDMA_SEND

图7

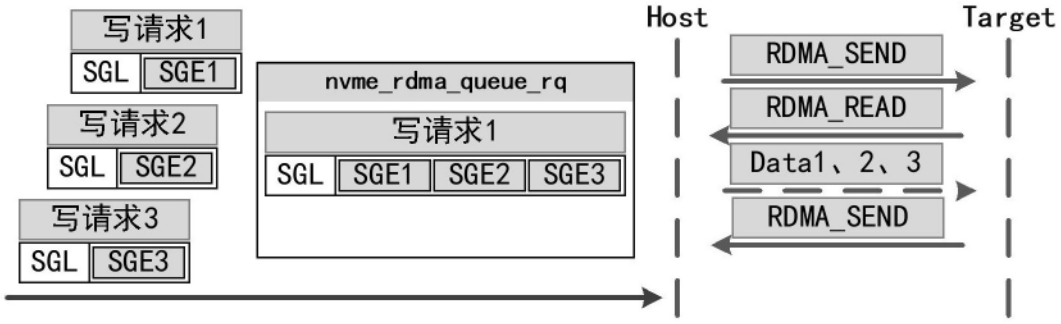

写请求1
SGL
SGE1
写请求2
SGL
SGE2
写请求3
SGL
SGE3
nvme_rdma_queue_rq
写请求1
SGL
SGE1
SGE2
SGE3
Host
Target
RDMA_SEND
RDMA_READ
Data1、2、3
RDMA_SEND

图8

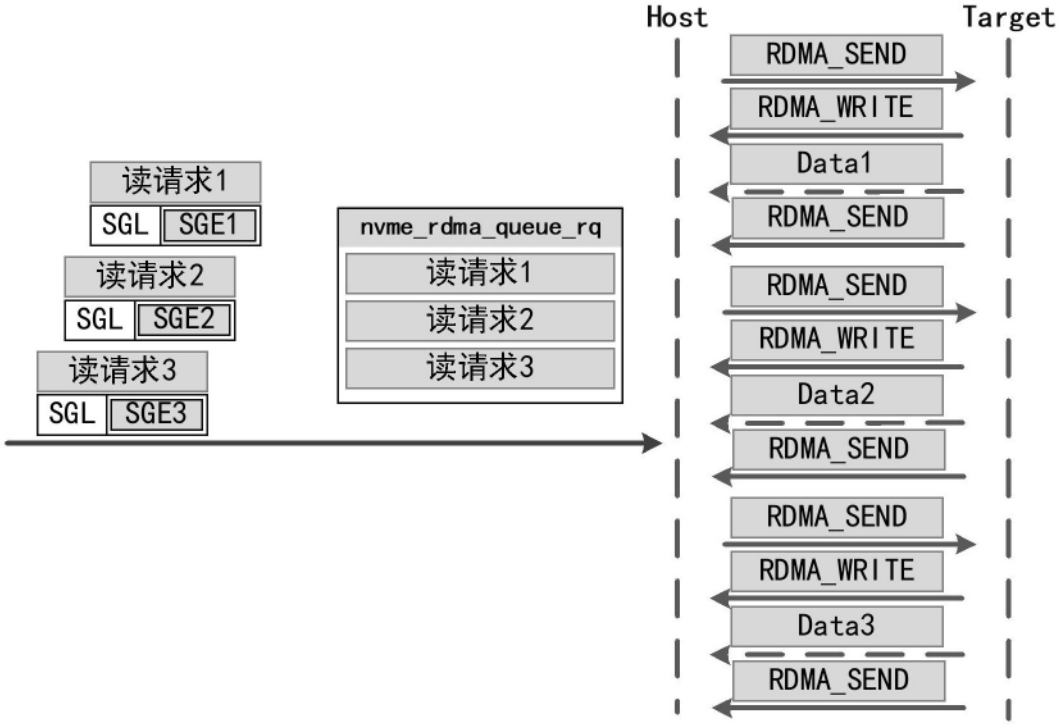

读请求1
SGL
SGE1
读请求2
SGL
SGE2
读请求3
SGL
SGE3
nvme_rdma_queue_rq
读请求1
读请求2
读请求3
Host
Target
RDMA_SEND
RDMA_WRITE
Data1
RDMA_SEND
RDMA_SEND
RDMA_WRITE
Data2
RDMA_SEND
RDMA_SEND
RDMA_WRITE
Data3
RDMA_SEND

图9

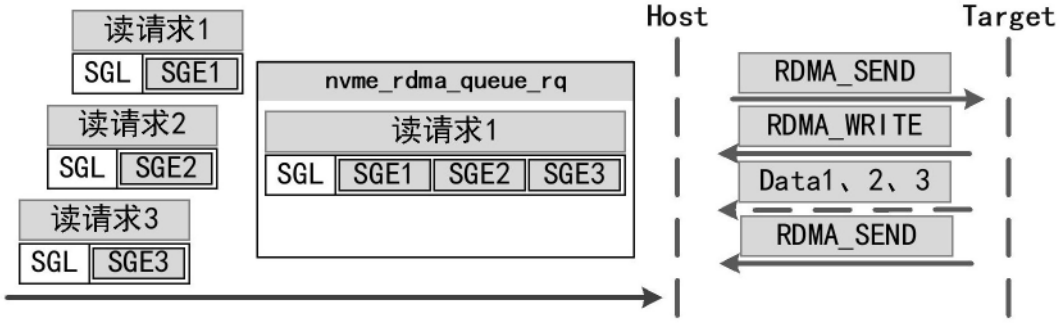

读请求1
SGL
SGE1
读请求2
SGL
SGE2
读请求3
SGL
SGE3
nvme_rdma_queue_rq
读请求1
SGL
SGE1
SGE2
SGE3
Host
Target
RDMA_SEND
RDMA_WRITE
Data1、2、3
RDMA_SEND

图10

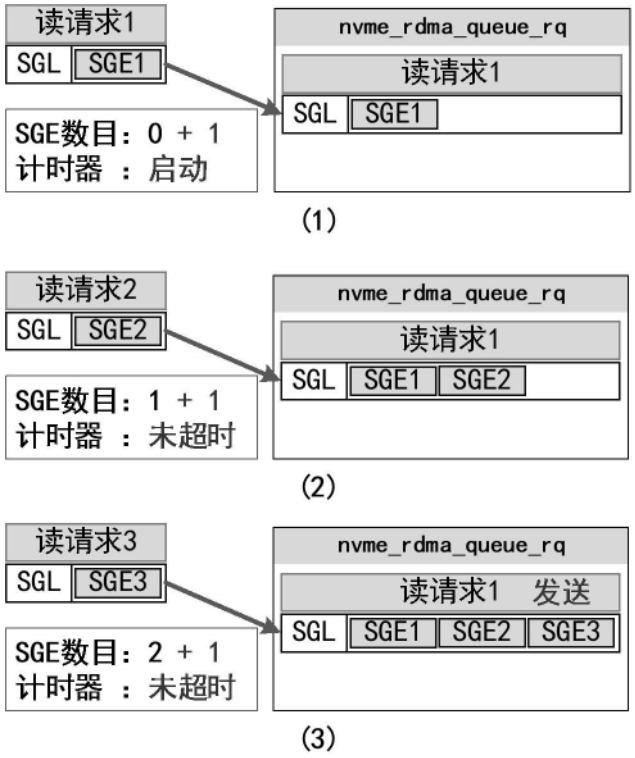

读请求1
SGL
SGE1
SGE数目：0 + 1
计时器 ：启动
nvme_rdma_queue_rq
读请求1
SGL
SGE1
(1)
读请求2
SGL
SGE2
SGE数目：1 + 1
计时器 ：未超时
nvme_rdma_queue_rq
读请求1
SGL
SGE1
SGE2
(2)
读请求3
SGL
SGE3
SGE数目：2 + 1
计时器 ：未超时
nvme_rdma_queue_rq
读请求1 发送
SGL
SGE1
SGE2
SGE3
(3)

图11

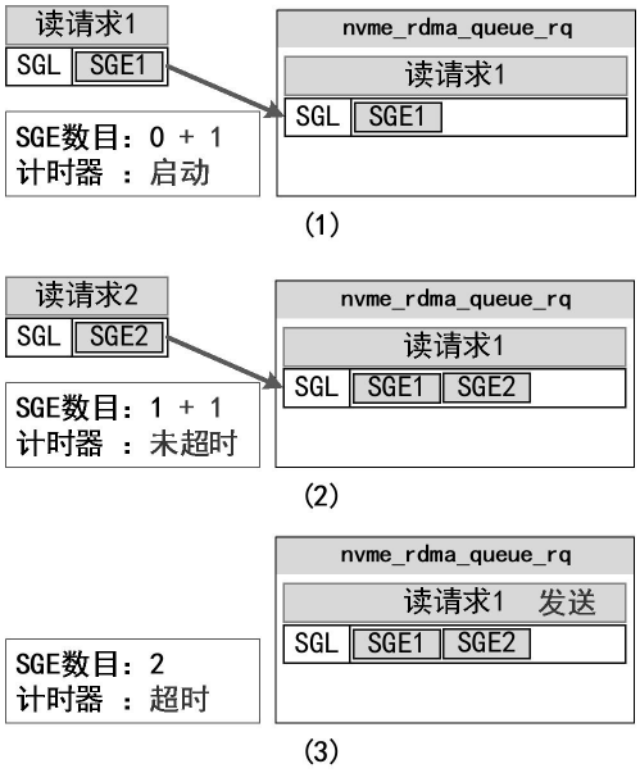

读请求1
SGL
SGE1
SGE数目：0 + 1
计时器 ：启动
nvme_rdma_queue_rq
读请求1
SGL
SGE1
(1)
读请求2
SGL
SGE2
SGE数目：1 + 1
计时器 ：未超时
nvme_rdma_queue_rq
读请求1
SGL
SGE1
SGE2
(2)
SGE数目：2
计时器 ：超时
nvme_rdma_queue_rq
读请求1 发送
SGL
SGE1
SGE2
(3)

图12