Journal of Computer Applications
计算机应用, 2020, 40(2): 316 - 320
ISSN 1001-9081
CODEN JYIIDU
2020-02-10
http://www.joca.cn

文章编号:1001-9081(2020)02-0316-05
DOI:10.11772/j.issn.1001-9081.2019091635

基于远程直接内存访问的高性能键值存储系统

王　成[1],叶保留[1*],梅　峰[2],卢文达[2]
(1. 计算机软件新技术国家重点实验室(南京大学),南京 210023;　2. 国网浙江省电力有限公司,杭州 310007)
(*通信作者电子邮箱yebl@nju.edu.cn)

摘　要:随着数据与系统规模的不断扩大,网络传输成为了键值存储系统的性能瓶颈。同时,远程直接内存访问(RDMA)技术能够支持高带宽和低时延的数据传输,为键值存储系统设计提供了新的思路。结合高性能网络中的RDMA技术,设计并实现了高性能、低CPU负载的键值存储系统Chequer;结合RDMA原语的特性,重新设计了键值存储系统的基本操作工作流程;并设计了基于线性探测的共享hash表,解决客户端缓存失效的问题以及提高hash命中率来减少客户端的读取轮数,进一步提高了系统的性能。在小规模集群上实现了Chequer系统,并通过实验验证了其性能。

关键词:远程直接内存访问;哈希表;键值存储;高性能网络
中图分类号:TP311　　**文献标志码**:A

High performance key-value storage system based on remote direct memory access

WANG Cheng[1], YE Baoliu[1*], MEI Feng[2], LU Wenda[2]
(1. *State Key Laboratory for Novel Software Technology (Nanjing University), Nanjing Jiangsu* 210023, *China*;
2. *State Grid Zhejiang Electric Power Company Limited, Hangzhou Zhejiang* 310007, *China*)

Abstract: With the continuous increment of data and system size, network communication becomes a performance bottleneck of key-value storage systems. Meanwhile, Remote Direct Memory Access (RDMA) technique can support high bandwidth, low latency data transmission, which provides a new idea for designing key-value storage systems. Based on RDMA technique in the high performance network, a key-value storage system named Chequer with high performance and low CPU overhead was designed and implemented. By combining the characteristics of RDMA primitives, the basic operation workflow of key-value storage system was redesigned. And a linear probing based shared hash table was designed to reduce the number of client reading rounds by solving the problem of client cache invalidation as well as increasing the hash hit rate, which can further improve the performance of the system. The Chequer system was implemented on the small-scale cluster, and its performance was demonstrated by experiments.

Key words: Remote Direct Memory Access (RDMA); hash table; key-value storage; high performance network

0　引言

远程直接内存访问(Remote Direct Memory Access, RDMA)[1]技术允许通过网络把数据直接传入到远程系统的存储区或者从远程系统存储区快速读取数据,消除了外部存储器的内容复制和系统的上下文切换的开销,能够实现高带宽和低时延的数据传输,为解决分布式键值存储系统数据传输瓶颈提供了解决思路。一种常用的方法是采用基于RDMA技术支持的InfiniBand[2]高性能网络作为分布式键值存储系统的网络环境,并通过采用IPoIB(Internet Protocol over InfiniBand)模式,可以不需要对分布式键值存储系统进行修改就将其运行在InfiniBand高性能网络上。然而,这种方式由于模拟以太网的数据传输,内核仍然需要处理数据包,将数据拷贝到应用中,无法充分利用RDMA的性能优势。因此,从更高效率的角度出发,应充分结合RDMA技术自身的特点,去构建基于RDMA的分布式键值存储系统。

RDMA技术提供了send/receive双边原语和read/write单边原语。Jia等[3]使用RDMA的双边操作加速分布式深度学习系统TensorFlow获得了6倍性能的提升;Li等[4]也使用RDMA加速了另外一个分布式深度学习系统MXNet。与双边原语相比,read/write单边原语不仅可以达到很大的带宽吞吐量,而且很少占用CPU的使用,能够极大地降低服务器CPU的负载,因此在进行系统设计时会着重考虑使用单边原语read/write。然而,由于read/write的远端机器无感知的特性,需要对传统键值系统put和get等基本操作进行重新设计。针对get操作设计,Pliaf[5]等系统中,单次get操作可能会需要过多的read轮数,影响了系统的整体性能;实际上,由于hash冲突的解决方式不同,往往网络read轮数也就不同,所以如何设计高效的hash表是解决这个问题的关键。针对put操作设计,FARM[6]、Herd[7]等系统大多采用多个轮询线程来监听数据的到来。这

收稿日期:2019-07-31;**修回日期**:2019-09-25;**录用日期**:2019-09-29。
基金项目:国家重点研发计划项目(2018YFB1004704);国家自然科学基金资助项目(61832005);国家电网公司科技项目(52110418001M)。
作者简介:王成(1994—),男,江苏泗洪人,硕士研究生,主要研究方向:高性能网络;　叶保留(1976—),男,江苏如东人,教授,博士,CCF杰出会员,主要研究方向:分布式计算、边缘计算;　梅峰(1977—),男,浙江湖州人,高级工程师,硕士,主要研究方向:电力信息系统、大数据;卢文达(1989—),男,吉林松原人,助理工程师,硕士,主要研究方向:数据挖掘、云计算。

虽然有利于性能的提高,但极大地耗费了CPU和内存资源。Nessie[8]系统仅仅是针对大的value提出了完全由客户端驱动的设计模式,使用了RDMA的原子锁机制,并不适用于时延敏感的小尺寸value。Craftscached[9]系统将通信区中读区域和写区域进行分离,从而避免数据写入时导致的数据损坏,但是这种方式会花费很多时间解决数据同步问题。除此之外,多数系统设计时总是无法避免数据的拷贝操作,对于数据拷贝也需要消耗服务器的CPU资源,当value的数据尺寸过大时,拷贝会严重影响系统的性能。

基于上述分析,本文设计并实现了基于RDMA的高性能、低CPU负载的键值存储系统Chequer,该系统具备如下几个重要特征:首先,为了降低服务器端负载,使用read原语设计了键值系统的get操作,实现了服务器的CPU旁路;其次,为了减少数据的拷贝操作,同时兼顾性能与工作负载,为get操作提供了两种传输模式;最后,设计了基于线性探测的共享hash表,解决客户端与服务器端读写竞争问题与客户端缓存失效问题,以及提高hash命中率,减少客户端的read操作轮数。在小规模集群上实现了Chequer系统,并采用基准测试工具对其性能进行了实验验证。

1　Chequer整体架构

本文结合RDMA原语的特性,设计了基于RDMA的键值存储系统Chequer[10],其总体架构如图1所示,主要由客户端和服务器端组成,且数据都要经过RDMA网络进行传输。

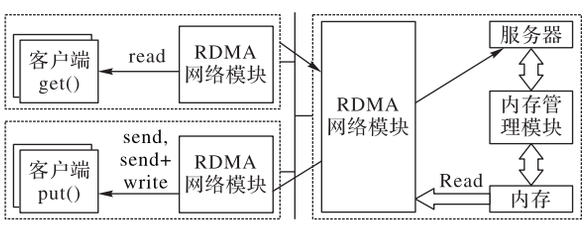

图1　Chequer整体架构

Fig. 1　Overall architecture of Chequer

在Chequer系统中,客户端的主要任务是负责发起put(key,value)、get(key)、delete(key)、contains(key)等键值存储的基本操作,而服务器端就是负责处理客户端发起的基本操作请求。在Chequer系统客户端发起的get操作中,客户端可以使用read原语直接从服务器端的内存里读取数据;而在put操作中,客户端可以在直接使用send原语和使用send+write原语这两种模式间进行选择。

以下将主要介绍Chequer系统中的两个重要模块:RDMA网络模块和内存管理模块。

1.1　RDMA网络模块

为了让远程服务器及时收到消息,并且使远程服务器CPU负载降低,在配合Chequer键值系统上层的get操作和put操作的前提下,网络模块实现了read的读取数据操作,以及send和send+write的发送数据操作。并且,由于RDMA底层网络传输模式非常复杂,传输接口均被封装成了类似于TCP/IP socket的接口以掩盖其复杂性。此外,又因为RDMA使能网卡(RDMA-enabled NIC, RNIC)的片上内存有限,当服务器端连接过多时,网络性能会受到影响,降低了网络传输质量。要解决这个问题,可以采用减少队列的创建和使用队列复用的方式。在共享接收队列的方式中,它允许工作请求被多个队列对(Queue Pair, QP)来共享接收。在事件驱动的模式中,当传入的消息来到任何一个与共享接收队列关联的QP,就会使用共享接收队列里的工作队列元素(Work Queue Element, WQE)来接收传入的数据。

1.2　内存管理模块

内存管理模块用来负责服务器端的内存管理。为了使服务器申请和释放内存块的速度得到提高,且尽量避免服务端成为性能瓶颈,系统实现了基于Buddy内存管理算法的二级缓存的内存池[11]。如图2所示,服务器首先向第一层快速缓存层申请缓冲区并释放缓冲区;经过内存池的二级缓存混合缓存层后,Buddy内存管理层直接向操作系统申请大块内存,且在RDMA网卡中注册这些内存,之后使用Buddy算法管理内存。Buddy算法使得Memory Manager对大块连续内存进行管理,它会调用mmap接口向操作系统申请大块的连续内存,并接收来自上层申请内存和释放连续内存的请求,且可以将大块的缓冲区切割成合适大小的缓冲区反馈给上层。

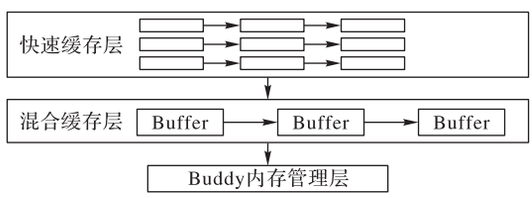

图2　内存池缓存模块

Fig. 2　Module of memory pool caching

2　Chequer系统关键技术设计

由于RDMA具有高带宽、低时延以及轻CPU负荷的特点,它通常用于加速一些现有的键值存储系统。然而,仅仅使用RDMA的send/receive原语来改写应用的网络层并不能充分发挥RDMA的优势。本章主要结合RDMA的read/write原语,对键值存储系统中的get和put操作流程进行重新设计。其中,为了减少数据的拷贝,针对put操作,系统提供了两种传输模式来提升操作的性能,而get操作根本不需要服务器端的参与。此外,为了减少每次get操作所需的平均read轮数,使get操作得到进一步优化,系统还设计了高效的共享hash表。

2.1　Put操作设计

键值存储系统中包含两个基本的修改类型操作:put(key,value)和delete(key),这两种类型的操作需要修改数据。在大多数现有的基于RDMA设计的键值存储系统中,针对put操作的设计通常会采用多个轮询线程的方式来监听数据的到达。这种多个轮询线程的方式虽然有利于性能的提高,但另一方面却极大地耗费了CPU的资源,因为即使没有客户端请求到来,仍然会有多个轮询线程在工作,占用了CPU资源。

除此之外,无论是基于send/receive消息模式的设计,还是带有轮询监听的write设计,服务器都无法避免从通信区到应用的内存数据拷贝。如果需要拷贝的数据量较大,不仅会造成很大的操作时延,还会造成CPU资源的消耗,严重影响系统的性能。因此,为了减少系统资源的消耗,可适当地减少数据的拷贝。例如,如果客户端已经确切地知道value需要存储在服务器端内存的位置,那么客户端可以直接使用RDMA的write操作将value写到客户端。

然而，为了达到数据的无拷贝目的，客户端必须得到value在服务器端的存储位置，因此，客户端需要额外与服务器进行一轮通信。但是这种策略对于小尺寸的value来说，若以两轮通信为代价，其最终需要花费的时间就变成了原来的2倍；然而随着value的逐渐增大，第一轮客户端与服务器的通信时间也就可以变得越来越忽略不计。所以，针对小尺寸的value，为了不影响其性能，本节将put操作设计为混合模式传输。

由于客户端发起存储key-value的请求中，value的大小往往不同，大小范围变化可从1 B到1 MB。因此，为了提高系统的操作性能，在设计put操作时，本节设计了put操作的两种传输模式供客户端选择。put操作的整体设计流程如图3所示。

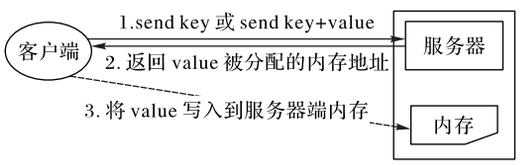

图3 put操作流程

Fig. 3 Operation process of put

put操作的第一种传输模式和传统的put操作模式相同。客户端将key和value直接封装成一条消息，并通过RDMA的send原语发送给服务器端；服务器端在接收到消息请求后，会将key和value拷贝到相应的hash表和存储区。

在put操作的第二种传输模式设计中，采用了RDMA的write原语。为了避免服务器端使用轮询线程监听消息的到来，减少CPU资源的浪费，客户端与服务器间需要额外的一轮通信。第二种put操作的运行过程为：客户端首先通过RDMA的send原语发送key和value的长度给服务器端；服务器在接收到消息后，会分配存储value的内存块，并将该内存块地址通过send原语发送返回给客户端；最后，客户端收到value的内存地址，再通过RDMA的write原语将value直接写入到服务器的内存中。

第一种put操作模式的特点是所有的put操作请求都是由服务器端处理的。由于服务器端根本不需要处理get操作，且实际应用场景中的工作负载基本上是读密集型的，因此可以根据需求适当分配一点工作给服务器端。在第二种put操作模式中，使用到了write原语，虽然它避免了服务器对value的拷贝操作，但是需要客户端与服务器端进行两轮通信，这种策略对小尺寸的value并不友好。因此，客户端需要将两种模式结合使用：当value的长度小于一定阈值时，可采用第一种put操作模式；而当value的长度大于或等于一定阈值时，采用第二种put操作模式。通过实验发现，当待发送的value数据小于1 KB时，第一种put传输模式所需要的时间较少；而当value的大小超过1 KB时，第二种put操作传输模式更优。然而，value长度的阈值设置不是固定的，软件运行环境（比如操作系统不同）及内存、CPU等硬件环境都会对阈值的设置产生影响，所以value长度的阈值介于512 B到2 KB之间。Chequer系统设置1 KB作为value长度的分界点，从而根据实际value的大小情况来选择不同的传输模式。

2.2 get操作设计

Chequer键值存储系统中包含两个基本的查找类型操作get(key)和contains(key)。传统的get类型操作都是在客户端与服务器端交互模式下完成的，但服务器的性能总是有上限的，如果一个服务器为多个客户端提供服务，而客户端向服务器的请求又过于频繁，那么服务器就很容易成为系统的性能瓶颈。

RDMA的read原语能够使远端节点CPU旁路并读取数据，完全不需要任何服务器端CPU的参与。如果Chequer键值系统的客户端已经得知数据存储在何处，那么就可以直接采用read操作进行数据的读取。这种方式不仅节省了服务器端CPU资源的使用，同时还提高了客户端的并发读取效率。

当使用RDMA的read原语设计get操作时，客户端与服务器端的交互模式随之需要改变。图4为get操作的整体设计流程。在get操作流程中，客户端操作主要有三个步骤：1)客户端首先通过read操作获取对应的key在服务器端hash表上的映射记录。2)客户端查询与key对应的value数据是否缓存在本地缓存中：如果value值被缓存，它就会被直接返回给客户端；如果value值没有被缓存，就执行下一个步骤。3)客户端根据第1)步获取的value的存储地址，再次使用read操作从服务器端读取value值。

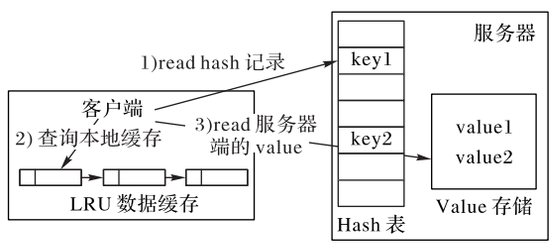

图4 get操作流程

Fig. 4 Operation process of get

get操作完全不会涉及到远端服务器的CPU。为了允许客户端发起RDMA的read操作，服务器端必须公开其hash表的数据结构。服务器端有两块内存区域：第一块内存区域是固定大小的hash表；另外一块内存区域用来存储value，其中value的长度是可变的。服务器端在启动时会将这两块内存区域注册到网卡上，而客户端在与服务器端建立连接时，会得到服务器端相关配置信息和这两块内存的注册信息。之后，客户端就可以在这两块内存区域上执行任意的RDMA操作。

由于服务器端不涉及到get操作，所以get操作完全是由客户端发起的。而完全由客户端发起get操作存在一定的时间代价，客户端至少需要花费两轮的时间进行读取操作，这样才能获取到key对应的value值。但是与传统的TCP/IP网络相比，即使该策略采用了两轮读取操作，这种设计模式在性能上仍然有很大的优势。此外，客户端的并发访问速度也得到了进一步提高，使得服务器的整体性能得到了极大的提升。为了进一步提高get操作的性能，又在中间的过程里添加了本地缓存。如果在本地缓存中能找到对应的value值，则不需要进行远端的value值读取，此时仅仅只需要进行一轮读取就可以获得相应的数据。

在get操作流程中实现的缓存方法是最近最少使用(Least Recently Used，LRU)算法，该算法对访问热点数据非常高效。一般的LRU算法是通过数组数据结构实现的，但是通过数组实现的LRU算法，其时间复杂度并不是非常高效。无论是查询操作还是删除操作，通过数组实现的LRU算法的

时间复杂度都是$O(n)$的,其中n为数组长度。为了提高客户端缓存查询的效率,本文设计了hashmap与双向链表相结合的LRU结构。Hashmap用于保证查询的时间复杂度为$O(1)$,而双向链表保证了元素添加与删除的时间复杂度为$O(1)$。

如图5所示,在LRU结构中,hashmap记录了key与双向链表的映射关系。例如,指针p1记录在key1中,它指向双向链表中记录了key1和value1的某个节点。Hashmap是基于哈希表的map接口实现,其查询时间效率为$O(1)$。由于hashmap是在客户端本地实现,所以可以在本地计算解决各种计算与hash冲突,使得即使存在hash冲突也不会太过影响查询效率。而双向链表用来缓存客户端前面读取的key-value键值对。其中,prev指针用来指向前一个节点,next指针用来指向下一个节点。在双向链表中还包括了一个头指针和一个尾指针,通过使用双向链表,客户端可以在$O(1)$的时间复杂度内删除节点或者插入新的节点。

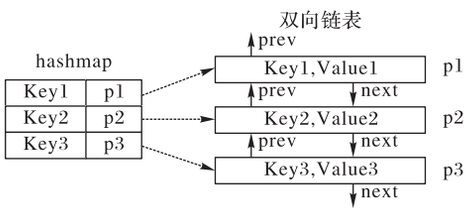

图5　LRU结构图

Fig. 5　Structure diagram of LRU

2.3　共享hash表设计

当客户端发起get操作时,它使用RDMA的read原语读取hash表中的每条记录。而所有的put操作,不论是两种put操作模式的哪种模式,都要在服务器端进行处理,并修改hash表。因此,当服务器端的CPU写本地内存时,客户端可能正在读取这段内存,这种情况就可能会导致客户端读取的信息混乱,这就是所谓的读-写竞争问题。

客户端采用了两轮操作来读取value数据。在第一轮中,首先根据key的信息获取value的内存地址,而在第二轮中才根据内存地址开始读取value。为了提高读取效率,客户端在本地缓存第二轮读取的value,然而客户端不知道它缓存的value数据是否在服务器端已经被修改。为了确定客户端缓存的key-value对是否无效,可在hash表的每一条记录中添加一个时间戳,当服务器端接收到来自的客户端put操作或者delete操作请求时,为对应的key生成时间戳并存储在hash记录中。而当客户端进行get操作时,在第一轮读取到key相关信息以后,客户端会在本地缓存中查找对应的key,如果在本地查询到的key对应时间戳与第一轮读取到的时间戳相同,那么本地缓存就不是无效的。否则客户端需要进行第二轮的read操作来获取最新的value,并及时更新本地LRU缓存中key的时间戳和value值。

图6显示了设计的hash记录数据结构。其中,in_use表示当前的记录是否为空,key是实际的键,value pointer指向存储value的内存地址,checksum1是value的校验和,time stamp是时间戳,checksum2就是整个记录的校验和。其中,指针指向存有value值的内存地址,value字符串的长度在前面,实际的value值在后面;客户端使用checksum1和checksum2来确定read读取的数据是否存在读-写竞争。如果发生checksum不一致,表明服务器端正在修改内存中的数据,此时客户端会反复尝试通过read操作获取数据,直到获取的数据正确为止。

hash表的命中率往往决定了系统的读取性能。这是因为随着命中率的提高,get的读取轮数也会随之降低。最理想情况下,get操作在单轮就可以读取到正确信息。为了实现这一点,就必须避免hash冲突,然而在现实中hash冲突总是会发生。在Herd系统[7]中,每种操作虽然只需要一轮,但是CPU会采用轮询的方式时刻监听消息的到来,降低了CPU的效率,而需要实现的目标是系统在保持高性能的同时降低CPU的负载。

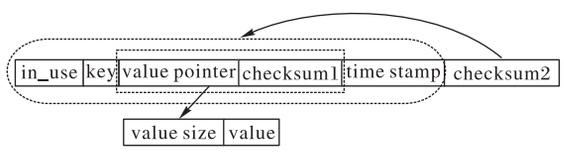

图6　hash记录结构设计

Fig. 6　Design of hash record structure

经过比较,最后选择了线性探测hash方式。这是因为线性探测与结合RDMA的read特性得到了进一步的优化。在线性探测hash中,将固定数量的连续的桶定义为块。结合RDMA的read能够读取连续内存的能力,在一轮中就读取到这个块。当第一个桶不满足且存在hash冲突时,线性探测会查询当前块中第一个桶的下一个桶。如果在当前块所有的桶中都没有查询到,就读取另外一个块。正是通过这种预取的方式,可以使得hash表的命中率大大提高。

3　系统实现与实验分析

实现了Chequer系统的C语言版本。为了减少频繁的内存注册操作,服务器需要提前预先注册一大块内存,而内存管理模块会负责该块内存的申请与释放。Hash表中的校验和采用CRC64循环冗余校验的方式,速度快,几乎不消耗CPU资源。最后,通过异步的方式将所有的put和delete操作的日志同步到本地磁盘。

有关集群配置的详细信息参见表1。在实验中,搭建了3个配备了RDMA网卡的节点,每台机器的配置信息完全相同,并且这3台机器都通过交换机互连。实验中的测试数据集由YCSB(Yahoo!　Cloud Serving Benchmark)[12]基准测试工具生成。YCSB可以根据实际的工作负载构造出各种可变长度的键值对。此外,对于key的访问也可以实现长尾zipf分布。在所有的实验中,value的长度大小从16 B到256 KB不等,并且应用测试使用了两种混合类型的工作负载。一种是10%的puts操作加90%的gets操作,另外一种是50%的puts操作加50%的gets操作。

表1　集群节点的配置信息

Tab. 1　Configuration information of nodes in cluster

软硬件配置	参数
CPU	Intel Xeon CPU E5-2620 v2 @ 2. 10 GHz
硬盘	2 TB
内存	64 GB
网卡	Mellanox ConnectX-5
操作系统	CentOS7. 1
GCC Version	7. 2. 0
OFED Version	4. 3-1. 0. 1

为了便于性能比较,实现了一个简易版本的Chequer-

message，它仅仅使用了 RDMA 的 send/receive 模式来传输数据。Chequer-message 的客户端使用 send 操作将键值系统的 put 请求或者 get 请求打包发送给服务器端，服务器处理后，会再次通过 send 模式将最终结果发送给客户端。

图 7 展示了 10% 的 puts 操作加 90% 的 gets 操作工作负载下的实验结果，图 8 展示了 50% 的 puts 操作加 50% 的 gets 操作工作负载下的实验结果。对比图 7 和图 8 可以发现，Chequer 不论在何种工作负载、何种 value 大小下，都获得了很高的性能。在 value 为 64 B 的情况下，Chequer 的吞吐量是 Chequer-message 的 2 倍多；而在 value 大小为 64 KB 的情况下，Chequer 的吞吐量是 Chequer-message 的 1.3 倍多。这是因为随着 value 的增大，系统开销主要花费在网络传输上。

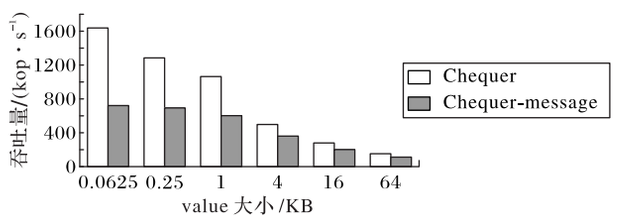

图 7 90% gets + 10% puts 工作负载下的服务器吞吐量

Fig. 7 Throughput of server under workload of 90% gets + 10% puts

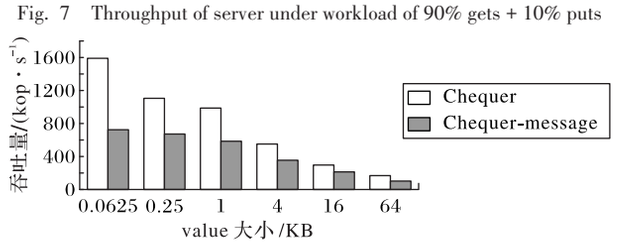

图 8 50% gets + 50% puts 工作负载下的服务器吞吐量

Fig. 8 Throughput of server under workolad of 50% gets + 50% puts

4 结语

随着处理器性能的逐渐增强，分布式系统遇到了网络瓶颈，RDMA 技术可以带来很好的网络性能优化。本文设计了一种基于 RDMA 的键值存储系统 Chequer，具有高性能、低 CPU 负载、可灵活扩展的特性。基于 RDMA 提供的原语，本文重新设计了键值存储系统的 put 操作和 get 操作流程，以达到高性能、低负载的特性，并结合 read 特性设计了基于线性探测的共享 hash 表，从而进一步优化了 get 操作性能。与采用传统设计方式实现的 RDMA 键值存储系统相比，在 value 为 64 B 时 Chequer 有着 2 倍以上性能的提升，而在 value 为 64 KB 时 Chequer 有着 1.3 倍以上性能的提升。与现有工作基于 RDMA 的键值存储系统相比，Chequer 会使用更少的 CPU 资源。

本文在进行 Chequer 系统设计时，主要考虑了系统性能与利用率，未来还将考虑结合多副本备份机制进一步提升其可用性[13-14]。

参考文献（References）

[1] RECIO R, METZLER B, CULLEY P, et al. A remote direct memory access protocol specification: RFC5040[S]. Fremont, CA: Internet Engineering Task Force, 2007-10.

[2] BUYYA R, CORTES T, JIN H. An introduction to the InfiniBand architecture[M]// High Performance Mass Storage and Parallel I/O: Technologies and Applications. Piscataway: IEEE, 2002: 616-632.

[3] JIA C, LIU J, JIN X, et al. Improving the performance of distributed TensorFlow with RDMA[J]. International Journal of Parallel Programming, 2018, 46(4):674-685.

[4] LI M, WEN K, LIN H, et al. Improving the performance of distributed MXNet with RDMA[J]. International Journal of Parallel Programming, 2019, 47(3):467-480.

[5] MITCHELL C, GENG Y, LI J. Using one-sided RDMA reads to build a fast, CPU-efficient key-value store[C]// Proceedings of the 2013 USENIX Annual Technical Conference. Berkeley, CA: USENIX Association, 2013: 103-114.

[6] DRAGOJEVIĆ A, NARAYANAN D, HODSON O, et al. FaRM: fast remote memory[C]// Proceedings of the 11th USENIX Symposium on Networked Systems Design and Implementation. Berkeley, CA: USENIX Association, 2014: 401-414.

[7] KALIA A, KAMINSKY M, ANDERSEN D G. Using RDMA efficiently for key-value services[J]. ACM SIGCOMM Computer Communication Review, 2015, 44(4): 295-306.

[8] CASSELL B, SZEPESI T, WONG B, et al. Nessie: a decoupled, client-driven key-value store using RDMA[J]. IEEE Transactions on Parallel and Distributed Systems, 2017, 28(12): 3537-3552.

[9] CHEN W, YU S, WANG Z. Fast in-memory key-value cache system with RDMA[J]. Journal of Circuits, Systems and Computers, 2018, 28(5): No. 19500749-.

[10] 王成. 基于 RDMA 的键值存储系统性能优化[D]. 南京:南京大学, 2019:56-70.（WANG C. Performance optimization of key value storage system based on RDMA[D]. Nanjing: Nanjing University, 2019: 56-70.）

[11] ZHENG L, KURODA S I, LIU H, et al. Buddy algorithm optimization in Linux memory management[J]. Applied Mechanics and Materials, 2013, 423/426:2746-2750.

[12] COOPER B F, SILBERSTEIN A, TAM E, et al. Benchmarking cloud serving systems with YCSB[C]// Proceedings of the 1st ACM Symposium on Cloud Computing. New York: ACM, 2010: 143-154.

[13] CHEN H, ZHANG H, DONG M, et al. Efficient and available in-memory KV-store with hybrid erasure coding and replication[J]. ACM Transactions on Storage, 2017, 13(3): No. 25.

[14] YIU M M T, CHAN H H W, LEE P P C. Erasure coding for small objects in in-memory KV storage[C]// Proceedings of the 10th ACM International Systems and Storage Conference. New York: ACM, 2017: No. 14.

This work is partially supported by National Key Technology Research and Development Program (2018YFB1004704), the National Natural Science Foundation of China (61832005), the Science and Technology Project of State Grid Corporation of China (52110418001M).

WANG Cheng, born in 1994, M. S. candidate. His research interests include high-performance network.

YE Baoliu, born in 1976, Ph. D., professor. His research interests include distributed computing, edge computing.

MEI Feng, born in 1977, M. S., senior engineer. His research interests include power information system, big data.

LU Wenda, born in 1989, M. S., assistant engineer. His research interests include data mining, cloud computing.