# 基于RDMA高速网络的高性能分布式系统

**魏星达，陈榕，陈海波**
上海交通大学并行与分布式系统研究所，上海 200240

**摘要**
高速的RDMA网络设备已经被广泛部署在现代数据中心。RDMA可以从两方面加速分布式系统：首先可以提供一种快速的消息处理机制，其次RDMA提供了新的硬件原语。这极大地提升了处理器的利用率以及对RDMA的使用率，但是需要重新设计系统。介绍了RDMA的研究进展，概述了近年来利用RDMA加速分布式系统的工作，包括基于RDMA重新设计的系统以及如何更好地利用RDMA的设计，并给出了未来的研究方向。

**关键词**
分布式系统；键值存储系统；图处理系统；联机事务处理系统；远程过程调用

中图分类号：TP316 文献标识码：A doi: 10.11959/j.issn.2096-0271.2018036

# *Optimizing distributed systems with remote direct memory access*

**WEI Xingda, CHEN Rong, CHEN Haibo**
Institute of Parallel and Distributed Systems, Shanghai Jiao Tong University, Shanghai 200240, China

***Abstract***
Fast network devices with RDMA support have been price-compatible with traditional network primitives such as Ethernet, and it's now widely deployed in modern data centers. RDMA can be used in two ways. Firstly, it can optimize the messaging primitive in distributed applications. The second way is to redesign the applications with RDMA's one-sided features. One-sided features provide high CPU utilizations and high network performance, but the system should be redesigned. The research progress of RDMA was introduced. An overview on the research efforts on using RDMA for distributed systems was presented. The works on how to use RDMA to redesign systems and the works on how to better leverage RDMA were included. The future research directions were also put forward.

***Key words***
distributed system, key-value stores system, graph processing system, OLTP system, remote procedure call

## 1 引言

对于分布式系统而言，如何加速网络通信一直以来都是一个非常重要的问题。例如此前的研究[1]指出，将一个单机键值存储系统应用到基于客户机—服务器（client-server）模式的分布式环境中，即便使用了批量处理（batching）等优化技术，仍然会造成大幅的性能下降。分布式系统依赖网络通信完成节点间的协作，因此通信开销很大程度上决定了应用程序的整体性能。传统的网络协议栈（如TCP/IP）并不是针对高性能应用场景设计的，因此难以提供高效的通信支持，系统调用和内存复制等操作都会带来巨大的性能开销。

远程直接内存访问（remote direct memory access，RDMA）技术是一种最早应用于高性能计算领域的网络通信协议，当前已在数据中心逐渐普及[2-3]。RDMA允许用户程序绕过操作系统内核，直接和网卡交互进行网络通信，从而提供高带宽和极小时延。此外，RDMA还提供了one-sided原语（one-sided primitive），即网卡可以在没有远端节点帮助的情况下，由网卡直接发起和完成对远程内存的读写请求，在提升CPU利用率的同时，为分布式系统的设计提供了更多的可能。从系统软件设计的角度而言，可以直接将RDMA视为一种更快的网络，并通过模拟TCP/IP的方式（即IBoIP模式）直接加速现有应用。然而这样无法完全利用RDMA提供的性能优势。近几年来，学术界以及工业界提出了一系列基于RDMA的分布式系统[4-11]，探索了如何通过对现有系统的再设计充分发挥出RDMA的硬件性能，实现数量级的性能提升。下面首先对RDMA技术进行简要介绍，并进一步从RDMA优化技术、远程过程调用实现、分布式键值存储系统、分布式事务处理系统等方面介绍当前领域的研究进展以及笔者在这些领域中的一些工作。

## 2 RDMA概述

RDMA网络协议允许用户程序绕过操作系统内核直接进行网络通信。这样既避免了用户空间到系统空间的复制开销，也可以省去进入内核处理的开销，极大地降低了网络时延，并且提高了吞吐量。此外，RDMA技术还提供了新的网络原语，网卡可以绕过处理器直接处理对服务器内存的读写请求。网卡提供了对远程内存的读（read）、写（write）和原子（atomics）操作，可以极大地提升远程服务器的CPU使用效率。**图1**展示了使用不同网络架构读取服务器端内存中数据的操作流程。**图1（a）**展示了如何利用RDMA的one-sided特性来减少服务器端的开销，服务器的网卡使用直接内存存取（direct memory access，DMA）读取用户需要的内存数据，并返回给用户。**图1（b）**展示了如何利用RDMA的消息原语（send/recv）来加速数据传递过程。客户端可以直接发送请求给网卡，网卡使用direct请求读回，并发送给远端服务器。这样节省了用户程序进入内核以及内存复制的开销。**图1（c）**展示了使用传统网络消息机制（TCP/IP）处理读写请求的过程。可以看到，用户程序首先进入操作系统内核，将请求复制到内核的缓冲区，内核再给网卡发送请求。服务端接收到用户请求后，读取内存内容，再以相同的步骤将读取的内存发回给客户端。

**表1**总结了RDMA支持的3种操作。send/recv操作支持传统消息通信的方法：一台机器可以用send操作给另一台机

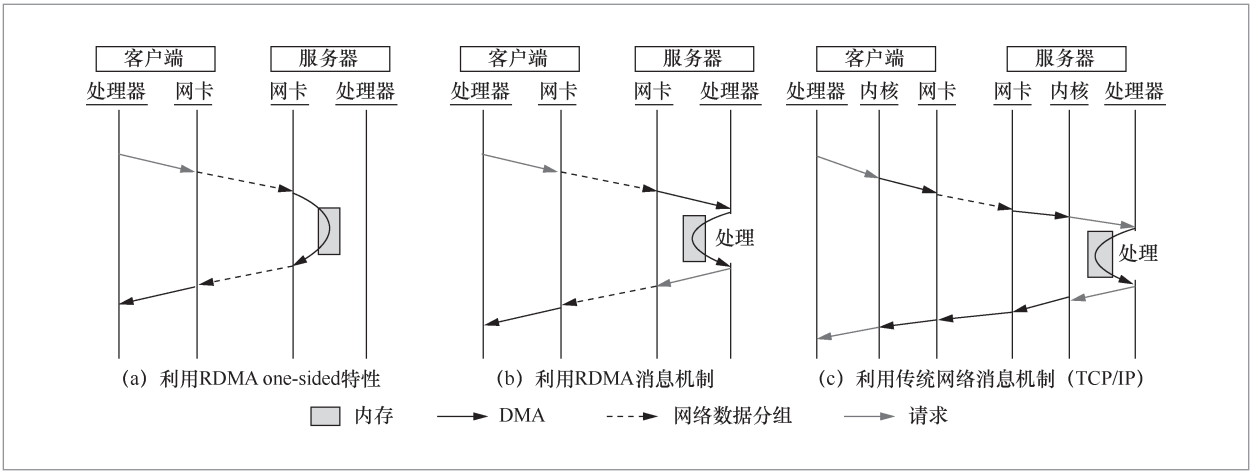


图1 RDMA与传统网络的对比

器发送消息，同时它可以用recv操作接收其他机器发送给它的消息。一台机器可以利用write请求对其他机器的内存进行写操作，使用read请求进行读操作。atomics操作包括原子比较再替换（compare and swap，CAS）以及读再增加（fetch and add，FAD）操作。这里笔者区分了read/atomics操作和send/write操作，这是由于send/write操作不需要返回远端服务器端的状态，而read/atomics操作需要将远端服务器的内存内容返回（CAS操作会将比较的值返回给用户程序）。

RDMA使用预先建立好的连接对（queue pair，QP）完成网络通信。一个连接拥有两个队列（queue）：一个发送队列和一个接收队列。发送队列记录着需要网卡发送的请求，接收队列记录其他网卡的请求以及发送队列中请求完成的事件。RDMA支持创建多种不同种类的连接，每种连接支持不同的RDMA请求种类，**表1**总结了不同种类的QP以及其所支持的操作类型。可靠连接（reliable connection，RC）支持RDMA的所有操作类型。不可靠连接（unreliable connection，UC）只支持send/recv操作和write操作，而不可靠数据报（unreliable datagram，UD）只支持send/recv操作。RC的QP有可靠性的保证，请求的完成状态会被返回给用户，因此RC的QP是最为常用的RDMA连接。然而，在特定场景中其他QP可以用来提升系统的性能。

表1 RDMA的连接模式以及支持的操作

| 连接模式/请求 | send/recv | one-sided write | one-sided read/atomics |
|---|---|---|---|
| RC | √ | √ | √ |
| UC | √ | √ | × |
| UD | √ | × | × |

当前流行的RDMA实现主要有3种：InfiniBand、RoCE和iWARP。InfiniBand是针对RDMA特性设计的，较早出现在高性能计算领域，并且近几年开始在数据中心得到普及[4-5]。RoCE和iWARP可以在传统以太网环境中实现RDMA的特性。

## 3 RDMA研究进展

目前学术界对RDMA在分布式场景中的应用主要涉及两个方面：一是如何更加

高效地配置和使用RDMA技术本身；二是如何利用RDMA特性加速现有系统。

### 3.1 RDMA优化技术

现有的RDMA优化工作主要集中在如何优化网卡资源的使用上。现有网卡一般使用DMA对内存数据进行操作。除了使用DMA对数据进行读取以外，网卡还需要使用DMA读取元数据，比如QP的信息以及和用户态内存相关的信息，一般被称为内存注册（memory registration，MR）。这些元数据的DMA操作会显著地影响RDMA的使用性能，网卡会使用自带的内存来缓存QP和MR的信息。然而网卡的内存大小比较有限，通常无法缓存所有的元数据。因此用户程序需要仔细地管理QP和MR创建的数量和使用方式。

MR记录了用户态虚拟内存地址到物理内存的页表。为了减少网卡需要注册的页表的数量，FaRM[4]使用了2 GB的物理页对RDMA进行注册，从而显著地减少了网卡需要存储的虚拟内存翻译项（page translation entry）。当需要注册大量内存时，该方法能够显著提升系统的性能。

对于RC和UC模式的QP来说，一个客户端线程需要使用不同的QP与不同服务器进行链接。这样当服务器数量和客户端线程数量上升的时候，通常一台机器会建立许多QP，从而造成网卡缓存的缺失。QP的操作是线程安全的，因此FaRM[4]让线程之间共享QP，以减少需要创建的QP的总数量。由于共享QP需要额外的同步开销[8]，FaRM利用群组来减少同步开销。

当一台机器运行多个RDMA应用时，应用需要建立单独的MR，而运行多个应用时会注册大量MR，造成网卡资源紧张。LITE[12]利用一个内核层管理RDMA内存注册。所有的应用进入内核发送RDMA请求，内核负责监测应用RDMA请求的读写权限。这样一台机器只需要一个MR即可注册所有的内存，极大地减少了在网卡上建立的MR的数量。

此外，应用需要使用内存映射I/O（memory-mapped I/O，MMIO）给网卡发送请求，而MMIO通常有比较大的开销。卡耐基梅隆大学的研究人员提出，多个RDMA请求可以被批量提交给网卡，这样只需要一个MMIO就能发送多个请求给网卡，其他请求的内容网卡可以使用DMA自行读取[7]。doorbell batching技术能够更好地利用网卡和处理器间的PCIe带宽，同时减少了发送端的处理器开销，因此具有更好的性能。需要注意的是，只有一个QP的请求才能够以doorbell batching的方式发送给网卡。

### 3.2 远程过程调用实现

远程过程调用（remote procedure call，RPC）[6]是分布式计算中最基础的机器间的交互方式。一台机器可以像调用一个本地函数一样调用远端机器的函数。高效实现RPC的核心是实现一个消息传输机制：调用方（客户端）需要将调用的函数以及参数发送给处理方（服务器端），而服务器端需要通过消息传输机制将函数执行的结果返回给调用方。

RDMA的QP直接提供send/recv接口实现RPC通信，然而send/recv操作在网卡端需要复杂的逻辑进行处理，通常性能没有直接使用RDMA one-sided操作好[4]。最近学术界和工业界提出了一系列基于RDMA的对RPC通信的优化。

FaRM[4]使用one-sided RDMA write发送消息，服务器通过轮询内存内容来接收消息。FaRM利用RDMA提供的缓存一致性和顺序写的特性保证服务器能完整

地收到消息。使用FaRM消息机制的吞吐量相比传统网络（TCP/IP），最高能有一个数量级的提升。Herd[11]优化了RPC在非对称环境下的性能。非对称环境指的是一个服务器需要和多个客户端进行交互的环境。Herd在客户端使用one-sided RDMA write发送消息，这是因为one-sided操作的接收端（in-bound）性能最好。Herd使用基于UD的send/recv操作从服务器端向客户端回复，这是由于UD的QP的连接数比RC的QP少，从而具有更好的可扩展性和发送端（out-bound）性能。FaSST[8]进一步将基于UD的send/recv操作应用到一个对称的场景中，利用doorbell batching对UD提升比较大的特性，FaSST RPC对发送方具有更好的性能。这对于一些发送消息不大的场景（如事务处理场景）非常有效。基于one-sided RDMA write的消息存在接收方的轮询开销随着机器数增加而增加的问题，这是由于每台发送方在接收方这里都需要有一个独立的缓存来存储消息。Octopus[9]提出的RPC实现了使用RDMA one-sided write-with-IMM操作来发送消息，通过在接收方产生一个write完成事件，不同服务器的写请求事件可以被接收方一次性地获取，从而避免了不必要的轮询。Su M等人[10]提出了一种新的思路——远程获取范式（remote fetching paradigm，RFP）来实现消息传输，服务器将请求的回复缓存在本地内存中，客户端使用one-sided RDMA read来读取消息。这样尽可能地利用了RDMA的性能，因为RDMA out-bound的性能比in-bound的性能差。

### 3.3 分布式键值存储系统优化

内存键值存储（key-value store）是分布式系统的一个重要组件。它可以作为一个单独数据库，也可以作为一个缓存层来加速基于磁盘的存储系统[13]。键值数据库被广泛部署在大规模网络服务商，如Amazon[14]和Facebook[15-16]等。键值数据库一般提供两种操作：Get(key)操作是指给定一个数据的key(键)，返回数据的值；Put(key,value)操作将key对应的数据值更新为value。

键值数据库的操作都是简单的读写操作，因此可以有效地利用RDMA。Pliaf[17]使用one-sided读操作来实现Get操作，从而达到高性能和高处理器利用率。直接使用read操作有一个问题，即一个Get/Put的查找通常需要进行多次内存读操作，造成多次网络通信。为了减少查询的网络通信次数，Pliaf利用Cuckoo Hash表[18]作为其底层的索引实现。Pliaf的Put操作发送给服务器处理，同时使用自验证（self-verifying）数据结构来确保one-sided读操作读回的数据在有并发的Put操作的情况下是一致的。FaRM-KV[4]使用了一个优化过的Hopscotch[19] Hash表来同时达到比较好的存储利用率和单词查找需要的读操作。Herd[11]利用Herd-RPC优化键值存储系统。通过优化，RPC可以拥有与read操作相似或者比read操作更好的性能，并且RPC有更少的网络传输次数。因此一个为键值存储优化过的RPC比基于多次one-sided读操作的查询操作有更好的性能。Cell[20]提供了一种可以用one-sided读操作实现Get的B树方案，从而使键值存储系统支持范围搜索（range query）。虽然B树的查找需要多个RDMA read，使用RDMA read在服务器负载比较高的情况下仍然能够获得性能优势。

### 3.4 分布式事务处理系统优化

许多电子商务系统依赖在线事务处

理(on-line transaction processing, OLTP)作为后端支持。在分布式事务处理中,事务需要给多台机器发送消息进行数据同步,同时服务器需要一直处理事务请求。如何高效地处理事务中的网络请求对系统性能的影响非常重要。

FaRM[5]利用一个混合的处理方式来执行事务,如数据读取和验证步骤使用RDMA read操作执行,其余的操作使用FaRM RPC[4]实现。FaSST[8]使用一个针对RDMA优化过的RPC库来支持事务处理。Zamanian E等人[21]提出了一个新的基于RDMA的执行架构——NAM(network-attached-memory),在NAM架构中客户端在大部分情况下只用RDMA和服务器交互。他们在NAM架构中优化了快照隔离(snapshot isolation, SI)协议[22],从而使得基于SI的分布式事务处理系统能够借助RDMA扩展到大规模集群中。

### 3.5 其他利用RDMA的系统

除了键值存储系统和事务处理系统外,还有许多根据RDMA特性设计的系统。GraM[23]利用一个对多核和RDMA感知的通信栈来优化分布式图分析系统。Octopus[9]是一个基于RDMA的分布式文件系统,利用RDMA来加速文件系统的文件读取修改操作,同时使用基于RDMA优化的RPC方法来管理文件系统的元数据。DrTM+B[24]是一个高效的在线数据迁移系统,利用RDMA read进行数据迁移,由于read操作绕过了处理器,数据迁移的过程基本不影响正常请求的处理。最后,还有一系列使用RDMA技术来加速一致性协议实现的工作[25-26]。APUS[26]使用基于RDMA的一致性(consensus)协议将用户的请求进行备份,在不修改服务器程序的情况下提供高效的可用性。

## 4 RDMA的应用

### 4.1 分布式键值存储系统: DrTM-KV

DrTM-KV[27]是一个基于one-sided RDMA操作的高效键值存储系统,能够为分布式事务处理系统提供数据存储支持。在事务处理系统中,处理器的计算资源被大量地用在请求处理上,因此提高服务器处理器的利用率非常重要。

#### 4.1.1 Cluster hashing

DrTM-KV使用Hash表来索引键值数据库,数据会根据数据键(key)来Hash到对应的内存地址,快速找到数据所在的位置。DrTM-KV只使用RDMA read和write操作实现Get和Put操作,这样就减少了处理器对数据查询的开销。这在事务处理的场景中非常重要,因为事务处理场景是一个处理器密集的应用场景。DrTM-KV的设计考虑了减少单次Get和Put操作所需要的内存读写次数,以减少对应的网络通信次数。传统的基于链表的Hash结构使用链表来存储具有相同Hash的数据值,以解决Hash冲突(Hash collision)问题。这样一次查找操作可能需要多次内存访问才能够找到所需的数据值。这种情况对于基于RDMA的实现不是很友好,因为多次访问对应了多次RDMA read请求。即使RDMA比传统网络快一个数量级,one-sided请求仍然比内存访问慢一个数量级[4]。这使得发送多次one-sided请求的效率低于发送一个网络消息让服务器帮忙处理的效率[11]。此外,为了减少每次Get以及Put操作所需的one-sided RDMA操作次数,DrTM-KV使用聚集(clustering)来减

少有Hash冲突的情况下所需的one sided操作次数。整个Hash表基于传统的链式Hash表的结构。为了支持使用RDMA的查询，DrTM-KV将RDMA注册的区域分为两个部分：索引结构和数据部分。每个索引部分的bucket记录了数据的键和其对应的值（在数据部分的地址）。

在DrTM-KV中，一个Get/Put操作首先通过Hash函数找到key所在的索引部分的bucket。如果在bucket中找到了key，那么可以直接从bucket中的地址读取数据的值。否则，需要遍历下一个bucket中的数据。为了减少每个查询bucket链的次数，DrTM-KV将多个键聚集到一个bucket中，这样有效地减少了链表的长度。将数据和索引分开会造成读取索引内容的不一致问题，在查找完数据地址到数据读取内容期间，数据的地址可能会被修改。这将导致查找时读取的数据地址和实际数据的位置不对应。DrTM-KV使用一个校验机制使得Get/Put操作可以检测出数据不一致的情况。数据会记录一个额外的校验码，这个校验码也会存储在索引的bucket中。为了节省空间，bucket中只存储校验码的14个bit，通常情况下这样足够检测出不一致的情况[4]。在插入或删除数据时，bucket和数据的校验码会被更新。这样DrTM-KV通过检测校验码是否一致来检测数据是否被删除。

#### 4.1.2 基于数据位置的缓存

经过Cluster hashing的优化，DrTM-KV能够有效地减少Get/Put操作需要查找索引的RDMA read次数。然而在最优情况下，DrTM-KV仍然需要2次RDMA read操作将数据的值读回：一次查找索引，一次读取数据的值。为了能够实现一次RDMA操作便能将数据读取回来，DrTM-KV在每台机器上使用了一个缓存来缓存数据键对应的值的地址。这样，当数据的地址被缓存在本地时，本地机器可以直接使用一次RDMA read操作将数据读回。需要注意的是，DrTM-KV并不缓存数据的值，而只缓存数据的地址，这样不需要进行数据值的同步。否则，每次数据的更新都需要将其他机器缓存的值更新，这在分布式环境下具有非常大的开销。

DrTM-KV的缓存在数据被插入或者删除时才需要更新。为了检测数据被删除或插入的情况，DrTM-KV的缓存也借助校验码来检测机器缓存的位置是否失效。缓存中同样存储着数据的校验码，当从缓存中的位置读回的数据校验码和缓存中的校验码不一致时，缓存被认为无效。这时应用需要重新利用RDMA遍历索引把数据读回。

通常不需要很大的缓存就能够记录很多数据的位置。首先，地址的大小通常比数据的大小小很多；其次，在真实应用场景中数据的访问模式都是不均匀分布的，也就是说某一小部分数据被访问的频率会比其他数据高很多。这样经常被访问的数据可以被缓存起来。DrTM-KV利用传统的缓存驱逐（eviction）机制（如LRU）来控制缓存的大小。

### 4.2 分布式图数据查询系统：Wukong

Wukong[28]是一个高效的分布式资源描述框架（resource description framework，RDF）图查询系统。RDF是一种描述网络资源的框架，被广泛用来描述网络中的数据集，例如Google公司的knowledge graph[29]。**图2**展示了Wukong的架构。RDF的数据集被划分后，部署在多台机器中，其中每台机器的每个核都共享这个图。一个用户请求需要访问多台

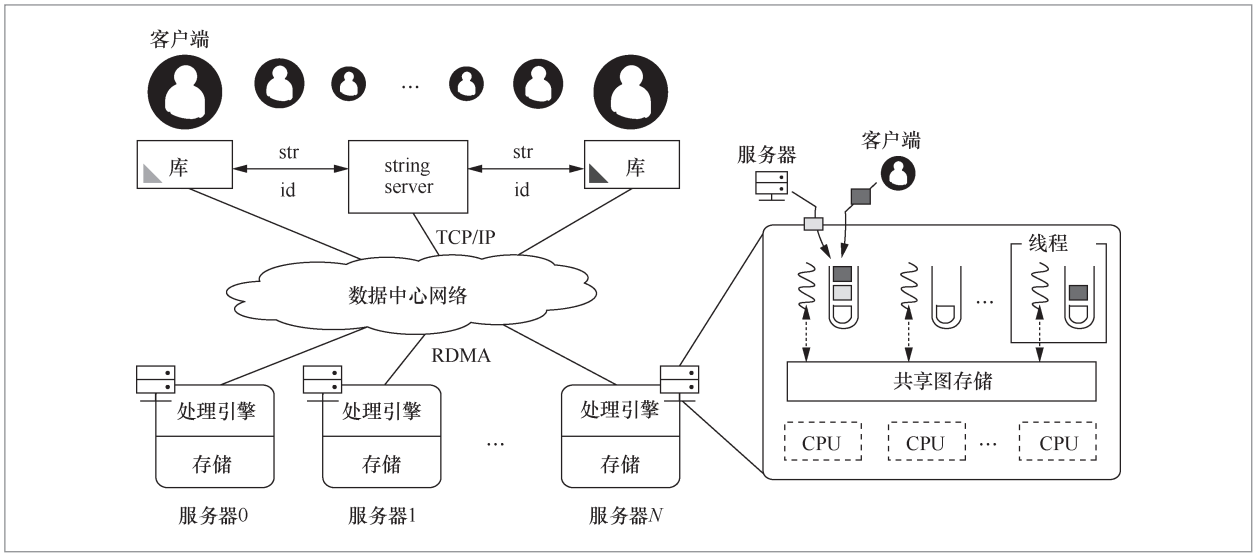


图 2 Wukong 的系统架构

机器的数据。Wukong针对RDMA的特性进行了一系列的优化来加速在RDF上的查询。首先，Wukong利用一个对RDMA和RDF友好的数据库来存储RDF数据，这样RDF图能通过RDMA高效地进行查询。其次，Wukong利用RDMA优化图查询的规划，使用全历史剪枝（full-history pruning）的方法优化查询的步骤。最后，Wukong利用RDMA扩展了传统分布式执行的fork-join模式，提出了in-place模式，并采用in-place和fork-join混合的方法来优化执行。

#### 4.2.1 RDMA友好的数据存储

图查询系统一般使用键值数据库来存储RDF图。Wukong使用DrTM-KV[27]作为其底层的键值存储系统，这样Wukong可以直接利用DrTM-KV的许多针对RDMA的优化，例如cluster Hashing和基于位置的缓存。

传统的RDF存储使用图顶点的ID作为数据的键，然而这会造成很多冗余的数据传输：因为一个顶点会有很多条边，这些边都会被划分到这个顶点的数据中。这样直接利用顶点ID查询图会读取很多不需要的边。在数据量很大时，RDMA的性能会急剧下降，因为这时网络的带宽会成为制约因素。Wukong采用更细粒度的划分方式，将图顶点的ID和边的方向作为数据库的键。这样一次查询只需要返回需要的点和边，极大地减少了单个查询所需要的数据传输量。

#### 4.2.2 全历史剪枝方法

一个对RDF的查询通常会检索到很大一部分图数据，因此适当地对要检索的图的空间进行剪枝会极大地提升查询的效率。传统的方法通常使用单步剪枝法，然而这样带来的并发性并不高，下一步查询需要等到当前查询的结果完成之后才能进行。Wukong基于RDMA的一个非常有趣的观察是：RDMA网络read操作发送1 byte和2 KB时，请求完成的时延几乎相同。Wukong可以发送更多的消息用来方便剪枝操作，却不会带来网络性能的明显下降。因此Wukong使用全历史剪枝技术，

在跨机器查询时，将查询的历史完整地发送给对方机器。这样，在每台机器执行查询的时候，所有任务可以依照历史信息进行独立剪枝，通过异步执行来提高查询的并发性。

#### 4.2.3 RDMA友好的执行模式

在传统分布式计算中，当查询涉及多台机器时，服务器利用一个fork-join的模式来计算：服务器发送请求给所有涉及的机器进行计算（fork），然后再汇总结果（join）。fork-join模式是一种自然的分布式计算方式，然而fork-join计算的时延与参与处理机器的负载情况非常相关。此外，总时延是由参与处理的最慢的机器决定的，容易造成较长的尾时延（long tail latency）[30]。长尾效应在实际应用场景中对系统的性能影响非常大，有研究表明渲染一个用户页面通常可能会存在几千个子请求[17]。

Wukong利用RDMA来减少长尾效应对图查询处理时延的影响，在可能的情况下，Wukong会利用RDMA read操作绕过处理器进行图查询。这被称为in-place执行模式。当查询可以直接用RDMA one-sided操作处理时，这个请求的时延基本恒定，不会被服务器处理器的负载所影响。同时one-sided操作相比等价的消息处理，具有更低的时延。虽然使用in-place模式可以完全绕过处理器，但是它只能处理比较简单的读数据请求，无法处理复杂的请求，比如聚合（join）等。因此，Wukong采取的是一种混合的模式，即当请求在一台机器上涉及的数据比较多时，Wukong采用fork-join模式来执行，否则，采用in-phace模式执行。Wukong同时使用RDMA write操作来加速fork-join模式下的消息传输。

## 5 未来方向

随着硬件技术的发展，最新的RDMA网卡已经可以达到200 GB的带宽，并且one-sided操作的时延可以低至600 ns。同时，RDMA网卡开始支持更多的one-sided操作。Mellanox的Connect-X5网卡提供了新的NVMeF（NVMe over fabrics）特性，使得RDMA one-sided操作可以直接操作固态存储设备。这让RDMA可以直接用来加速分布式的日志系统。同时学术界提出了一系列针对RDMA one-sided操作的新特性[31-32]。例如，Sabres[31]使用了原子的one-sided read操作，这个操作可以用来加速基于one-sided read的键值存储系统。这些更加高效的硬件给新的分布式系统设计带来新的挑战，因为网络将不再是性能的瓶颈。

除了利用RDMA网卡来加速现有分布式系统以外，现有系统开始使用多种硬件结合来进一步地获取性能提升。由于RDMA支持更加通用的网络操作，one-sided操作只支持简单的读写语义，这样直接用来实现系统（如键值存储）会带来不必要的开销。KV-direct[33]利用可编程网卡将键值存储数据库操作直接下陷到网卡中，在KV-direct中键值数据库的性能只被网络或者PCIe限制。随着摩尔定律增长变得缓慢，未来的分布式系统将更加依赖于定制化的硬件来获得新的性能提升。

## 6 结束语

本文介绍了如何利用新型快速网络RDMA来优化典型的分布式系统。首先介绍了如何利用RDMA来优化消息传输机

制，从而对分布式系统进行通用的优化，并进一步介绍了在多个领域基于RDMA技术的优化工作，包括键值存储系统和事务处理系统等，展示了如何基于RDMA特性来进行系统的重新设计。

## 参考文献:


[1] FAN B, ANDERSEN D G, KAMINSKY M. MemC3: compact and concurrent MemCache with dumber caching and smarter hashing[C]//The 10th USENIX Conference on Networked Systems Design and Implementation, April 2-5, 2013, San Francisco, USA. Berkeley: USENIX Association, 2013: 371-384.

[2] GUO C, WU H, DENG Z, et al. RDMA over commodity ethernet at scale[C]//The 2016 ACM SIG-COMM Conference, August 22-26, 2016, Florianopolis, Brazil. New York: ACM Press, 2016: 202-215.

[3] TSAI S Y, ZHANG Y. Lite kernel rdma support for datacenter applications[C]//The 26th ACM Symposium on Operating Systems Principles, October 28-31, 2017, Shanghai, China. [S.l.:s.n.], 2017: 306-324.

[4] NARAYANAN D, HODSON O, CASTRO M, et al. FaRM: fast remote memory[C]//The 11th USENIX Conference on Networked Systems Design and Implementation, April 2-4, 2014, Seattle, USA. Berkeley: USENIX Association, 2014: 401-414.

[5] DRAGOJEVIĆ A, NARAYANAN D, NIGHTINGALE E B, et al. No compromises: distributed transactions with consistency, availability, and performance[C]//The 25th Symposium on Operating Systems Principles, October 4-7, 2015, Monterey, USA. New York: ACM Press, 2015: 54-70.

[6] NEWMARCH J. Remote procedure call[D]. Pittsburgh: Carnegie Mellon University, 1981.

[7] KALIA A, KAMINSKY M, ANDERSEN D G. Design guidelines for high performance RDMA systems[C]//2016 USENIX Annual Technical Conference, June 22-24, Denver, USA. Berkeley: USENIX Association, 2016: 437-450.

[8] KALIA A, KAMINSKY M, ANDERSEN D G. FaSST: fast, scalable and simple distributed transactions with two-sided (RDMA) datagram RPCs[C]//The 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), November 2-4, 2016, Savannah, USA. Berkeley: USENIX Association, 2016: 185-201.

[9] LU Y, SHU J, CHEN Y, et al. Octopus: an RDMA-enabled distributed persistent memory file system[C]//2017 USENIX Annual Technical Conference (USENIX ATC 17), July 12-14, 2017, Santa Clara, USA. Berkeley: USENIX Association, 2017: 773-785.

[10] SU M, ZHANG M, CHEN K, et al. RFP: when RPC is faster than server-bypass with RDMA[C]//The 12th European Conference on Computer Systems, April 23-26, 2017, Belgrade, Serbia. New York: ACM Press, 2017: 1-15.

[11] KALIA A, KAMINSKY M, ANDERSEN D G. Using RDMA efficiently for key-value services[C]//The 2014 ACM Conference on SIGCOMM, August 17-22, 2014, Chicago, USA. New York: ACM Press, 2014, 44(4): 295-306.

[12] TSAI S Y, ZHANG Y. Lite kernel rdma support for datacenter applications[C]//The 26th ACM Symposium on Operating Systems Principles, October 28-31, 2017, Shanghai, China. New York: ACM Press, 2017: 306-324.

[13] BRONSON N, AMSDEN Z, CABRERA G, et al. TAO: Facebook's distributed data store for the social graph[C]//The 2013 USENIX Conference on Annual Technical Conference, June 26-28, 2013, San Jose,

USA. Berkeley: USENIX Association, 2013: 49-60.

[14] DECANDIA G, HASTORUN D, JAMPANI M, et al. Dynamo: amazon's highly available key-value store[J]. ACM SIGOPS Operating Systems Review, 2007, 41(6): 205-220.

[15] ATIKOGLU B, XU Y H, FRACHTENBERG E, et al. Workload analysis of a large-scale key-value store[J]. ACM SIGMETRICS Performance Evaluation Review, 2012, 40(1): 53-64.

[16] NISHTALA R, FUGAL H, GRIMM S, et al. Scaling memcache at Facebook[C]//The 10th USENIX Conference on Networked Systems Design and Implementation (NSDI'13), April 2-5, 2013, Lombard, USA. Berkeley: USENIX Association, 2013: 385-398.

[17] MITCHELL C, GENG Y, LI J. Using one-sided RDMA reads to build a fast, CPU-efficient key-value store[C]// The 2013 USENIX Annual Technical Conference, June 26-28, 2013, San Jose, USA. Berkeley: USENIX Association, 2013: 103-114.

[18] PAGH R, RODLER F F. Cuckoo hashing[J]. Journal of Algorithms, 2004, 51(2): 122-144.

[19] HERLIHY M, SHAVIT N, TZAFRIR M. Hopscotch Hashing[C]//The 22nd International Symposium on Distributed Computing, September 22-24, 2008, Arcachon, France. Berkeley: USENIX Association, 2008: 350-364.

[20] MONTGOMERY C M K, NELSON L, SEN S, et al. Balancing CPU and network in the cell distributed B-Tree store[C]//2016 USENIX Annual Technical Conference, June 22-24, 2016, Denver, USA. Berkeley: USENIX Association, 2016.

[21] ZAMANIAN E, BINNIG C, HARRIS T, et al. The end of a myth: distributed transactions can scale[J]. Proceedings of the VLDB Endowment, 2017, 10(6): 685-696.

[22] BERENSON H, BERNSTEIN P, GRAY J, et al. A critique of ANSI SQL isolation levels[C]//The 1995 ACM SIGMOD International Conference on Management of Data, May 22-25, 1995, San Jose, USA. New York: ACM Press, 1995: 1-10.

[23] WU M, YANG F, XUE J, et al. GraM: Scaling graph computation to the trillions[C]//The 6th ACM Symposium on Cloud Computing (SoCC'15), August 27-29, 2015, Kohala Coast, USA. New York: ACM Press, 2015: 408-421.

[24] WEI X, SHEN S, CHEN R, et al. Replication-driven live reconfiguration for fast distributed transaction processing[C]//2017 USENIX Annual Technical Conference, July 12-14, 2017, Santa Clara, USA. Berkeley: USENIX Association, 2017: 335-347.

[25] POKE M, HOEFLER T. Dare: high-performance state machine replication on RDMA networks[C]//The 24th International Symposium on High-Performance Parallel and Distributed Computing, June 15-19, 2015, Portland, USA. New York: ACM Press, 2015: 107-118.

[26] WANG C, JIANG J, CHEN X, et al. APUS: fast and scalable paxos on RDMA[C]//The 2017 Symposium on Cloud Computing, September 25-27, 2017, Santa Clara, USA. New York: ACM Press, 2017: 94-107.

[27] WEI X, SHI J, CHEN Y, et al. Fast in-memory transaction processing using RDMA and HTM[C]//The 25th Symposium on Operating Systems Principles, October 4-7, 2015, Monterey, USA. New York: ACM Press, 2015: 87-104.

[28] DEAN J, BARROSO L A. The tail at scale[J]. Communications of the ACM, 2013, 56(2): 74-80.

[29] SHI J, YAO Y, CHEN R, et al. Fast and concurrent RDF queries with RDMA-based distributed graph exploration[C]// The 12th USENIX Conference on Operating Systems Design and Implementation, November 2-4, 2016,

Savannah, USA. Berkeley: USENIX Association, 2016: 317-332.
[30] AJOUX P, BRONSON N, KUMAR S, et al. Challenges to adopting stronger consistency at scale[C]//The 15th USENIX Conference on Hot Topics in Operating Systems, May 18-20, 2015, Kartause Ittingen, Switzerland. Berkeley: USENIX Association, 2015.
[31] DAGLIS A, USTIUGOV D, NOVAKOVIC´ S, et al. Sabres: atomic object reads for in-memory rack-scale computing[C]// The 49th Annual IEEE/ACM International Symposium on Microarchitecture, October 15-19, 2016, Taipei, China. Piscataway: IEEE Press, 2016: 1-13.
[32] RAIKIN S, LISS L, SHACHAR A, et al. Remote transactional memory: 20150269116[P]. 2015-09-24.
[33] LI B, RUAN Z, XIAO W, et al. KV-direct: high-performance in-memory key-value store with programmable NIC[C]// The 26th Symposium on Operating Systems Principles, October 28-31, 2017, Shanghai, China. New York: ACM Press, 2017: 137-152.

## 作者简介

**魏星达**(1992-),男,上海交通大学并行与分布式系统研究所硕士生,主要研究方向为分布式系统、数据库事务处理、利用新型硬件的分布式系统。

**陈榕**(1981-),男,博士,上海交通大学并行与分布式系统研究所副教授,ACM/IEEE/CCF会员,主要研究方向为系统软件、并行与分布式系统。在SOSP、OSDI、EuroSys、USEIX ATC等国际著名会议发表多篇学术论文,并获得EuroSys 2015、ACM APSys 2017与IEEE ICPP 2017的最佳论文奖。

**陈海波**(1982-),男,上海交通大学并行与分布式系统研究所教授、博士生导师,主要研究方向为操作系统与并行分布式系统。在操作系统、基于新型硬件的事务处理系统与大数据查询系统等领域做出了引领性工作。入选2014年国家“万人计划”青年拔尖人才计划,获得2011年全国优秀博士学位论文奖、2015年CCF青年科学家奖、2017年CCF NASAC-东软青年软件创新奖与2018年CCF青竹奖。目前担任ACM SIGOPS ChinaSys副主席、ACM APSys指导委员会主席、《ACM Transactions on Storage》编委,ACM CCS 2018系统安全领域主席。在国际著名学术会议与学术期刊上共发表学术论文100余篇,获得ACM EuroSys 2015、ACM APSys 2013/2017与IEEE ICPP 2007的最佳论文奖与IEEE HPCA 2014的最佳论文提名奖。研究工作也获得Google Faculty Research Award、IBM X10 Innovation Award、NetApp Faculty Fellowship与华为创新价值成果奖等企业奖励。

**收稿日期:** 2018-05-10
**基金项目:** 国家自然科学基金资助项目(No.61772335)
**Foundation Item:** The National Natural Science Foundation of China(No.61772335)