# 基于 RDMA 的分布式系统研究进展

本文系中国人民大学在读硕士生李婧瑶所著，本篇也是 OceanBase 学术系列稿件第三篇。

「**李婧瑶：**中国人民大学信息学院在读硕士生，硕士期间在中国人民大学数据库与智能信息检索实验室从事新硬件 RDMA 的相关算法研究，在并发控制算法优化和异构网络下的高可用系统设计方向取得了一定成果，将继续在分布式事务处理领域努力探索。」

在分布式系统被广泛需要且性能瓶颈明显的背景下，由于远程直接数据存取（Remote Direct Memory Access, 简称 RDMA）提供了高带宽、低延时、以及内核旁路等特性，其相关研究引起了学术界和工业界的广泛关注。为统筹理解、寻求启发，本文提出了 RDMA 相关论文分类框架，并以近年来部分经典工作为例，从所属分类、发表机构两个角度概述了现有工作，并总结了分布式系统可能的未来方向。


## 引言

由于单机系统在计算能力和存储容量上的限制，特别是扩展能力的不足，已经不能满足互联网各类应用场景对海量数据分析和处理的要求，分布式系统得到了产业和学术届的重视。特别是分布式数据库系统，由于支持分布式事务，在银行、电商等关键任务领域得到了广泛应用。然而，现有的研究结果[1]表明，基于传统 TCP/IP协议以及 Share-Nothing 架构的分布式事务处理技术，受制于事务调度器的 CPU 低利用率、事务调度器与存取节点的网络高延迟，分布式事务的性能瓶颈明显。

基于无限带宽（InfiniBand, 简称 IB）高速网络的远程直接数据存取（RDMA）技术为解决这一问题提供了新思路。RDMA 能在更有效利用 CPU 的同时降低通信延迟，针对性缓解分布式数据库系统中的现存问题。

RDMA 利用预编程的网卡（RDMA NIC, 简称 RNIC）和成对创建的队列对（Queue Pair, 简称 QP）使如下优势成为可能：

(1) 绕开 TCP/IP 协议栈。不同节点间数据的网络传输不再需要层层封装和解封；

(2) 绕开操作系统。数据传输在用户态完成，免去用户态和内核态的上下文切换开销;

(3) 绕开远端 CPU。远端仅作为存储节点被访问，访问过程不需要远端 CPU 参与。

具体地，RDMA 提供单边、双边两类原语：单边原语包括读（Read），写（Write），原子操作（如Compare And Swap, 简称 CAS; Fetch And Add, 简称 FAA）；双边原语包括发送，接收（Send/Receive）。应用通过调用单边或双边原语，享有部分或全部 RDMA 带来的性能优势。

基于上述原因，近年来出发点、侧重点各不相同，实现方法迥异的有关分布式数据库系统中的 RDMA 的研究与日俱增。本文整理并归类了近年来的部分经典工作，旨在提炼现有的工作内容，并希望能为未来方向提供可能的灵感。


**分类概述**

为便于读者理解，本章首先简要描述 RDMA 的分类基础，然后从所属分类和发表机构两个角度，以时间线为索引，罗列并概述相关论文。总的来说，RDMA 相关工作主要可以分为以下四大类：

一、设计基础

探寻 RDMA 的最优使用方法。主题包括引入 RDMA 后的分布式系统架构探讨（包括共享内存，存算分离等），单/双边原语的选择，原语使用方法优化（e.g.Doorbell Batching，简称 DB）等。

二、网络相关算法重构

利用 RDMA 对网络传输的改善来提高系统性能。主题包括重写分布式事务的并发控制算法（包括可串行化的两阶段锁、乐观并发控制，和快照隔离级别等），重写分布式副本同步算法以实现高可用等。

三、非网络相关算法重构

考虑 RDMA 对网络瓶颈转移，重新设计非网络相关算法。主题包括数据分区算法的重写等。

四、多种新硬件结合

结合 RDMA 和其它新硬件，如非易失存储器（Non-Volatile Memory, 简称NVM）和硬件事务内存（Hardware Transactional Memory，简称 HTM）。多种新硬件结合的主题又可细分为设计基础(4.1)、网络相关算法重构(4.2)、非网络相关算法重构(4.3)三类。

| 分类 | 文章（简称. 署名机构. 发表年[参考文献]） | |
|---|---|---|
| 1 | HERD. CMU, Intel Labs. 2014[2]Guidelines. CMU, Intel Labs. 2016[3]FaSST. CMU, Intel Labs. 2016[4]Custom. Microsoft Research. 2016[7]NAM. Brown University. 2016[14]TreeIdx. TU Darmstadt, Brown University, MIT. 2019[16]GAM. National University of Singapore, University of California at Santa Barbara, Zhejiang University. 2018[19] | |
| 2 | FaRMv1. Microsoft Research. 2014[5]FaRMv1'. Microsoft Research. 2015[6]FaRMv2. Microsoft Research. 2019[8]DrTM+H. Shanghai Jiao Tong University. 2018[12]NAM-DB. Brown University, Oracle Labs. 2017[15]Active-Memory. Brown University, MIT. 2019[17]DSLR. University of Michigan, Ann Arbor. 2018[20]TCores. ETH Zurich. 2019[21] | |
| 3 | Chiller. Brown University, MIT, TU Darmstadt. 2020[18] | |
| 4 | 4.1 | RDMA-NVM-hints. Shanghai Jiao Tong University. 2021[13] |
| 4.2 | DBX. Fudan University, Shanghai Jiao Tong University, New York University. 2014[9]DrTM. Shanghai Jiao Tong University. 2015[10]DrTM+R. Shanghai Jiao Tong University. 2016[11] | |
| 4.3 | ... | |

表1.RDMA相关研究工作分类

基于以上四大类，高校和业界近年来发表了诸多文章。表 1 给出了本文所涉及工作的分类情况，图 1 给出了它们对应的时间线。下文将以文章署名机构为切入点，以发表时间为线索，概述相关论文。考虑到同一机构前后发表的文章往往有逻辑递进等依赖关系，期望能给读者以启发。

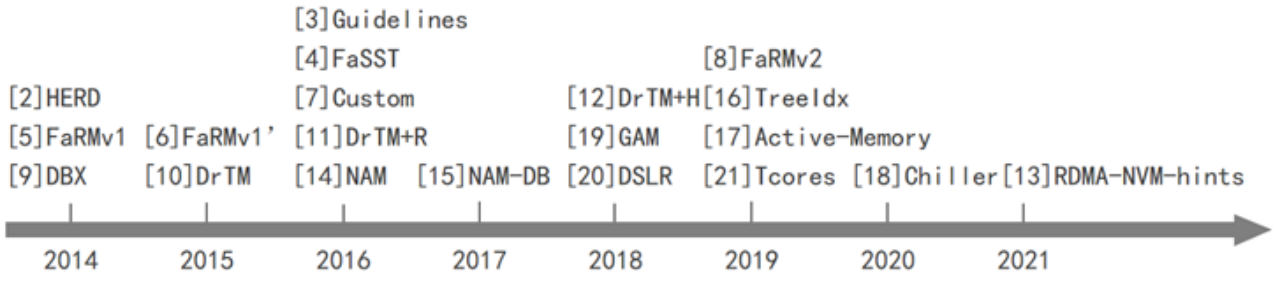


图1.RDMA 相关研究工作发表时间线

卡内基梅隆大学和英特尔研究院从多个角度研究了 RDMA 的设计基础。他们首先尝试证明键值系统中双边比单边的表现更好[2]，并为此提出了 RDMA 下的键值系统——HERD。HERD 中，客户端通过不可靠连接（Unreliable Connected, 简称UC）的单边写将消息写入服务端, 服务端通过不可靠数据报（Unreliable Datagram, 简称 UD）的双边 Send 回消息。实验证明，在不对称场景（all-to-one，即多个客户端向一个服务端发送 RPC）中，能达到和使用单边原语相似的性能。然而，在分布式事务常用的对称场景（all-to-all，即每个节点都可能发送请求和回复）中 HERD 呈现了较差的可扩展性，此时单边比双边的性能高 4 倍。为进一步优化使用，他们在另一工作[3]中列出了使用 RDMA 的一些技巧点（如减少内存映射 I/O 等），并针对每点提供了一些优化方案（如 DB 等）。最后，基于 HERD[2]和优化方案[3]，双边被进一步扩展到了对称场景[4]，达到了比使用单边原语更高的性能。

微软研究院利用 RDMA 对网络传输的改善，重构并逐步完善了相关算法。他们首先给出了用 RDMA 重写分布式事务两阶段提交和乐观并发控制算法的初步方案——FaRMv1[5]。如图2，FaRMv1 首先设计了一个消息传递原语：通过利用RDMA单边写写入环形缓冲区和本地线程读环形缓冲区，实现了不同节点间的快速通信。其次，针对共享内存区的本地读写和远程 RDMA 读写之间的并发冲突，通过在每个缓存行增设版本号实现了“无锁 RDMA 读”：仅当锁 L 为 0 且各行的版本号一致（Vc1，Vc2和Vobj的低阶位相符）时，认为读到了一个一致的正确状态。

为了在支持高可用的同时保留高性能，后续工作[6]在 FaRMv1 的基础上提供了故障恢复策略并优化了副本协议。至此，FaRMv1 仅为提交了的事务提供严格可串行化。考虑到为所有事务——包括提交和回滚的事务，提供了严格可串行化的必要性，FaRMv2[8] 利用 RDMA 单边周期性地协调全局时钟并根据时间戳排序事务。除了直接利用 RDMA 对网络传输的改善来提高系统性能的网络相关算法重构，微软研究院在另一工作[7]中探讨了数据库系统设计基础，即 RDMA 共享内存架构的实现。通过提出一个使用轻量级文件 API 公开远程服务器上的可用内存，并在本机内存不足时利用 RDMA 访问集群中的远程可用内存，而非使用本机磁盘的算法，实现了明显性能提升。

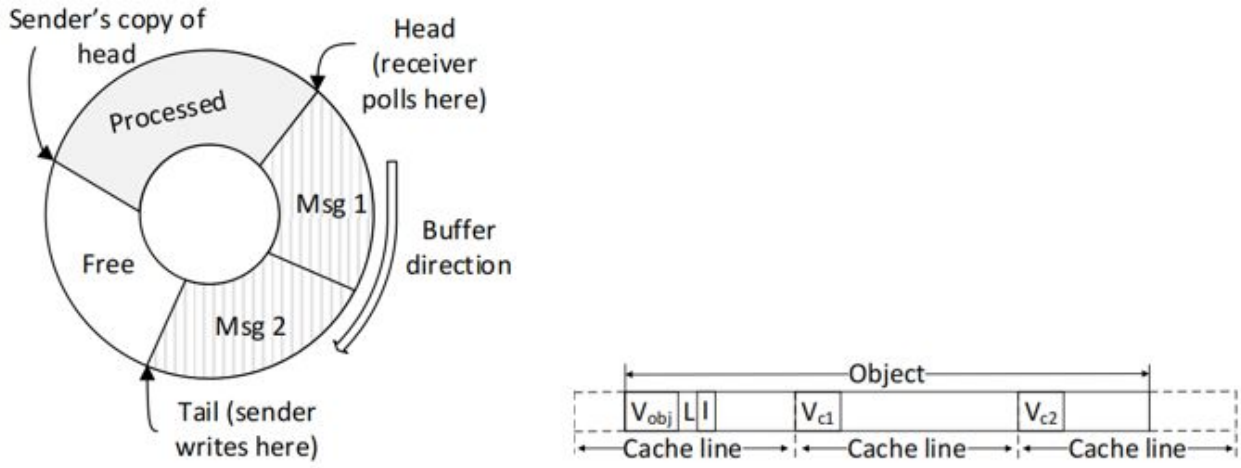


图2.FaRMv1的快速消息传递缓冲区设计及无锁RDMA读实现

在上海交通大学和复旦、纽约大学共同发表的研究中，他们首先设计了加入 HTM 和 RDMA 的事务系统[9]。此后，上海交通大学发表了多篇将 RDMA 与多种新硬件结合的文章。如图 3，DrTM[10]利用 HTM 本地并发控制和 RDMA 远程并发控制之间的强一致性，实现了可串行化的两阶段锁协议。如图 4，DrTM+R[11]通过乐观和加锁混合并发控制策略改进了 DrTM 需要预知读写集的缺点，并进一步优化了副本机制以实现高可用。DrTM+H[12]详细探讨了乐观并发控制算法的最优 RDMA 实现。通过将乐观并发控制算法细分为如图 5 所示的执行、验证、日志、提交四个

阶段，并在每个阶段详细对比 CPU 密集型、网络密集型两个数据集的实验结果，分阶段总结了最优的 RDMA 方案。此外，在另一工作[13]中结合对分布式数据库系统 DrTM+H 和分布式文件系统 Octopus 的分析和实验，总结了在分布式事务系统中同时使用 RDMA 和 NVM 的 9 个窍门。

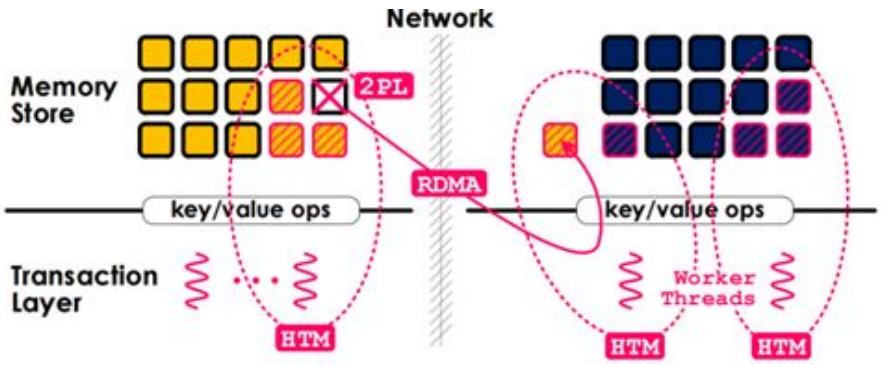


图3.DrTM系统架构图

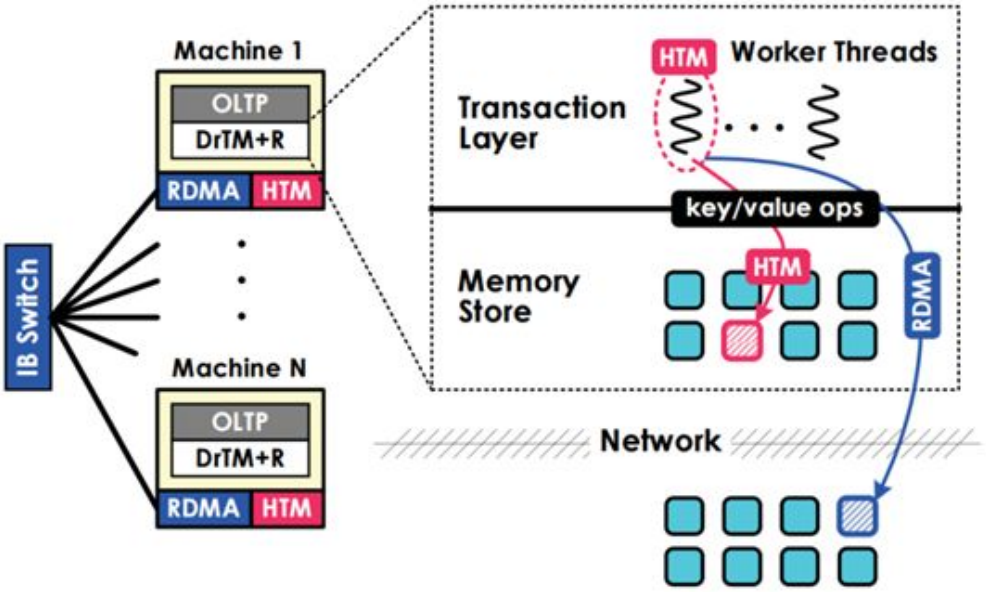


图4.DrTM+R系统架构图

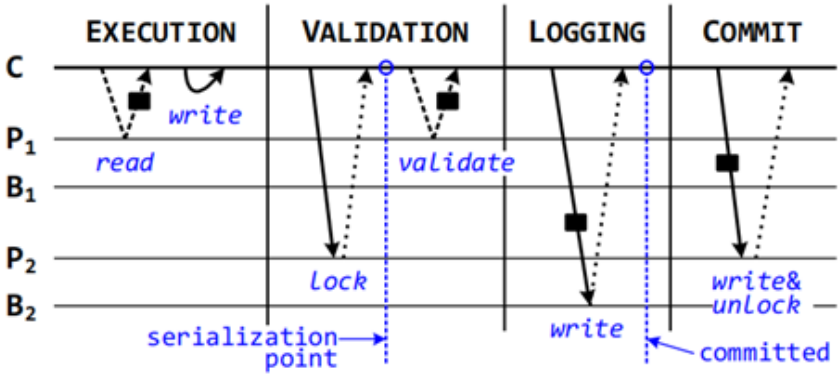


图5.DrTM+H的经验优化阶段划分——一个读P1写P2的事务实例

布朗大学通过列举并对比现存的分布式内存系统架构：包括使用传统 TCP/IP 协议和以太网的无共享架构、使用IB网络的无共享架构、引入 RDMA 的共享内存架构等，提出了存算分离架构NAM[14]。如图6所示，NVM将存储节点和计算节点分离，任意一个计算节点可以访问任意一个存储节点，由此可以实现存储可扩展和易调节的负载均衡。此后，在和 Oracle 实验室、麻省理工、德国达姆斯达特工业等大学的合作中发表了多篇文章。NAM-DB[15] 基于 NAM，实现了事务处理系统快照隔离级别下的可扩展。另一工作[16]探讨了 NAM 架构下的树形索引设计，设计并衡量了使用 RDMA 单边、双边或混合原语的索引访问方法。

进一步，RDMA 新型架构下系统瓶颈的转移使重构网络、非网络相关算法变得必要。一方面，CPU 计算开销取代网络成为了瓶颈。Active-Memory 方案[17]以最小化 CPU 冗余计算为目标，重新设计了强一致性主从副本管理方案以提供高可用。如图 7 所示，Active-Memory 方案的第一步使用 RDMA 单边写在写入 RDMA 友好的撤销日志（undo log）的同时直接更新副本数据，第二步仅需更改日志的标志位即可，由此实现了至少两倍的性能提升。另一方面，考虑到数据冲突的瓶颈作用，Chiller[18] 重新设计了数据分区和事务执行算法。它以最小化数据冲突为目标，将常被一起访问的数据放在同一分区上，并将事务划分为冷热两块分开执行，实现了相对传统分区算法两倍的性能提升。

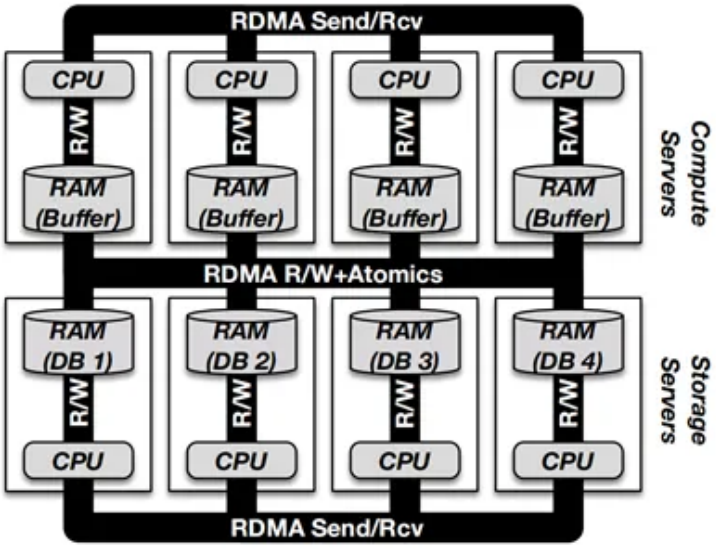


图6.NVM存算分离架构

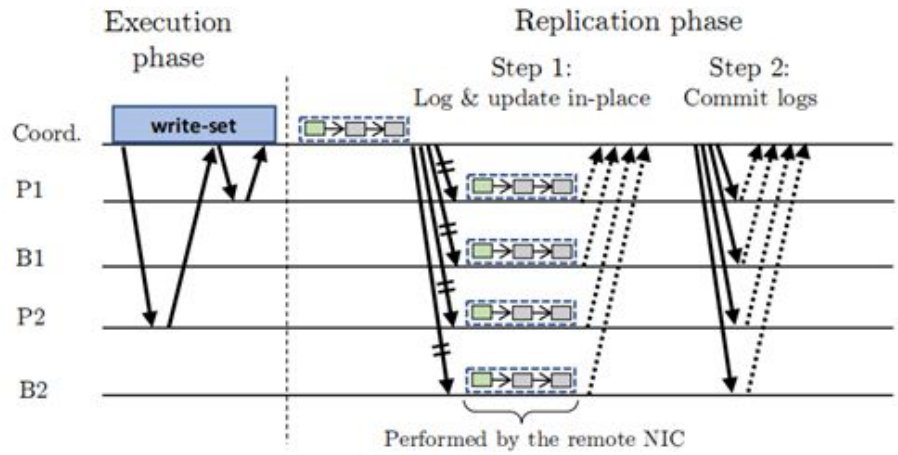


图7.Active-Memory副本同步方案

此外，还有众多高校展开了对 RDMA 设计基础和相关算法重构的研究。新加坡国立大学，加州大学圣巴巴拉分校和浙江大学提出了适用数据库系统、文件系统等各类系统的全局 RDMA 共享内存模型 GAM[19]。GAM通过基于目

录的缓存一致性协议，管理分布在多个节点上的空闲内存，并支持一组用户友好的内存操作 API。密歇根大学安娜堡分校参考 Lamport 面包店算法设计了支持读写锁的 DSLR 算法[20]。DSLR 利用 RDMA 单边原语，通过保证先到先服务避免了饥饿，并应用一种基于超时的方法来检测死锁。苏黎世联邦理工设计了支持更多锁类型的 RDMA 两段锁算法[21]。通过在服务端分开存放数据和锁表，并利用 RDMA 单边写写信息到服务端，由服务端执行加锁并返回，支持了包括共享锁、排他锁、意向共享锁、意向排他锁、共享意向排它锁等多种锁类型，因此也存在加锁开销大的问题。


**结论**

通过将 RDMA 相关工作分为四大类，并以发表机构聚簇，以时间线为索引，本文希望能在给读者全局概念的同时，启发创新灵感。通过概览所列文章，可以看到网络相关算法重构是发表论文最多的，也是利用 RDMA 最直观的分类。相比之下，网络瓶颈转移所引起的非网络相关算法重构尚处于起步阶段，或成为下一个可能的研究方向。另一方面，可以看到同一机构前后发表的论文间多有逻辑递进关系，因此对比研读常常能起到加深理解的作用。

由于篇幅所限，本文并未列出所有相关论文，而只是以部分经典论文为起点，给出一个大致的逻辑框架，还需要在此基础上继续探索完善。总体而言，对 RDMA 新型硬件在分布式系统中的应用研究和工业落地方兴未艾，变革传统架构和算法，带来更极致性能的同时必然引起更多的关注和创造更丰富的可能性。

***参考文献：**

[1]Michael Stonebraker. The Traditional RDBMS Wisdom is (Almost Certainly) All Wrong[EB/OL].(2013-05-30)[2021-06-28].https://slideshot.epfl.ch/play/suri_stonebraker.

[2]Using RDMA Efficiently for Key-Value Services. ACM SIGCOMM Computer Communication Review, 2014.

[3]Design Guidelines for High Performance RDMA Systems. USENIX, 2016.

[4]FaSST: Fast, Scalable and Simple Distributed Transactions with Two-Sided (RDMA) Datagram RPCs. OSDI, 2016.

[5]FaRM: Fast Remote Memory. NSDI, 2014.

[6]No compromises: distributed transactions with consistency, availability, and performance. SOSP, 2015.

[7]Accelerating Relational Databases by Leveraging Remote Memory and RDMA. SIGMOD, 2016.

[8]Fast General Distributed Transactions with Opacity. SIGMOD, 2019.

[9]Using Restricted Transactional Memory to Build a Scalable In-Memory Database. EuroSys, 2014.

[10]Fast In-memory Transaction Processing using RDMA and HTM. SOSP, 2015.

[11]Fast and General Distributed Transactions using RDMA and HTM. ACM EuroSys, 2016.

[12]Deconstructing RDMA-enabled Distributed Transactions: Hybrid is Better! OSDI, 2018.

[13]Characterizing and Optimizing Remote Persistent Memory with RDMA and NVM. USENIX ATC, 2021.

[14]The End of Slow Networks: It's Time for a Redesign. PVLDB, 2016.

[15]The End of a Myth: Distributed Transactions Can Scale. PVLDB, 2017.

[16]Designing Distributed Tree-based Index Structures for Fast RDMA-capable Networks. SIGMOD, 2019.

[17]Rethinking Database High Availability with RDMA Networks. PVLDB, 2019.

[18]Chiller: Contention-centric Transaction Execution and Data Partitioning for Modern Networks. SIGMOD, 2020.

[19]Efficient Distributed Memory Management with RDMA and Caching. VLDB, 2018.

[20]Distributed Lock Management with RDMA: Decentralization without Starvation. SIGMOD, 2018.

[21]Strong consistency is not hard to get: Two-Phase Locking and Two-Phase Commit on Thousands of Cores. PVLDB, 2019.