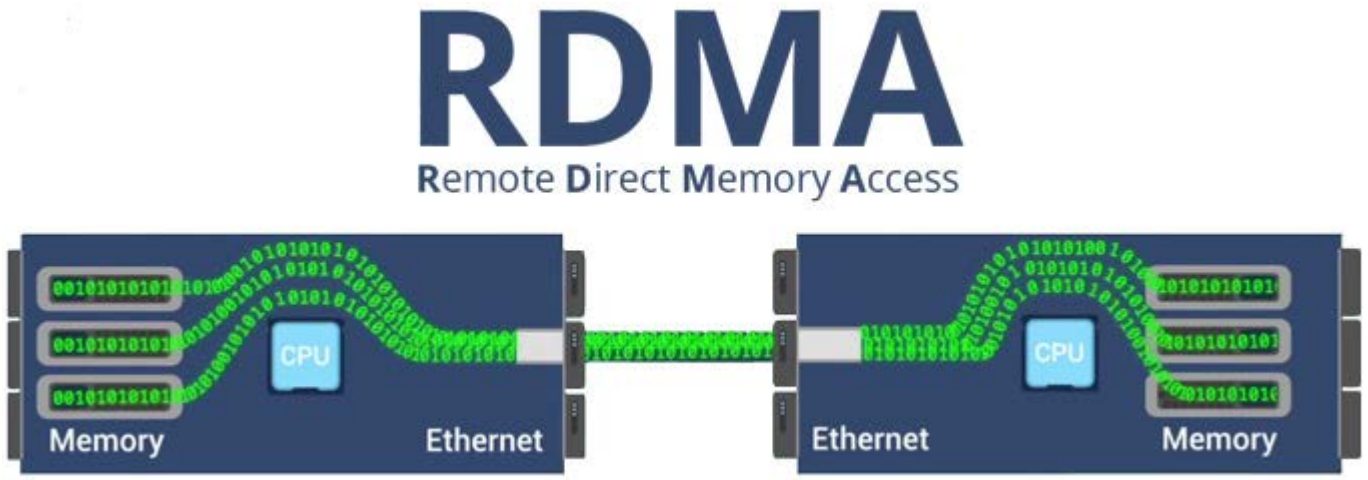
基于RDMA的分布式系统研究介绍

2022.06.24.

目录

研究背景介绍



- **高带宽**
- **低时延**
- **内核旁路**
- **零拷贝**

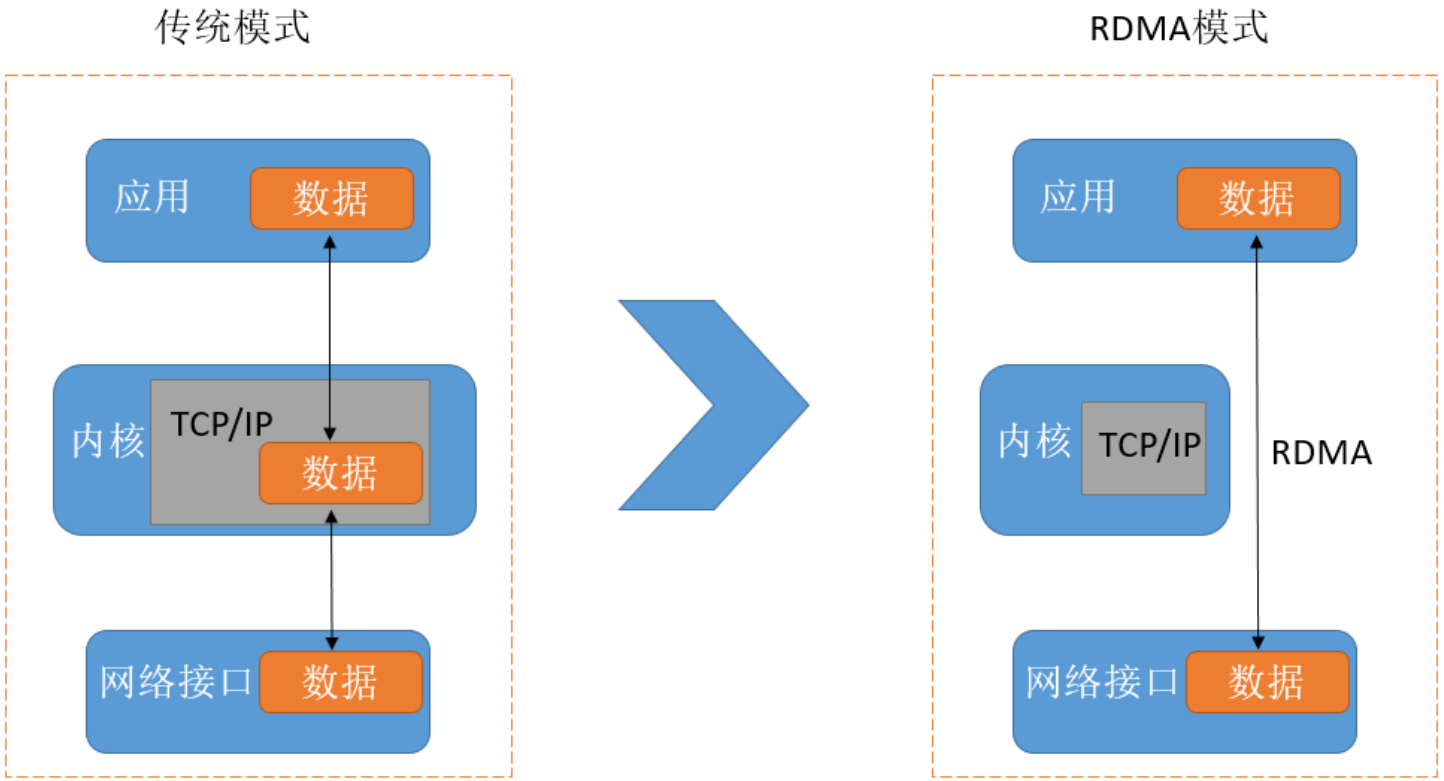
RDMA 利用预编程的网卡（RDMA NIC, 简称 RNIC）和成对创建的队列对（Queue Pair, 简称 QP）使如下优势成为可能：

(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。

研究背景介绍



举例来说，40Gbps 的 TCP/IP 流能耗尽主流服务器的所有 CPU 资源；而在使用 RDMA 的 40Gbps 场景下，CPU 占用率从 100% 下降到 5%，网络时延从毫秒级降低到 10μs 以下。

- 协议栈处理占用 CPU 负载
- 协议栈处理带来数十微秒的时延
- 消息处理上下文切换占用高

- 绕开远端 CPU
- 绕开 TCP/IP 协议栈（协议栈处理时延降至1微秒左右）
- 绕开操作系统上下文切换

目录

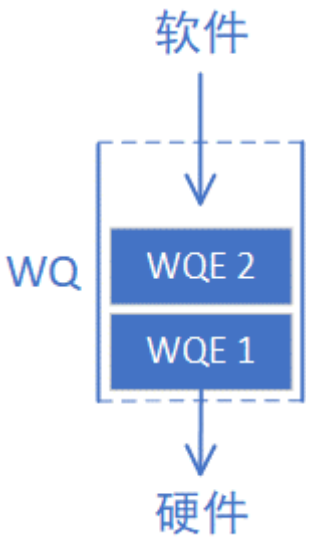
- 研究背景介绍
- RDMA协议简介
- 研究方向介绍
  - RDMA原语优化
  - 基于RDMA的分布式组件KV存储
  - 使用RDMA加速数据库通信层
  - 基于RDMA和NVMM的分布式文件系统
- 总结

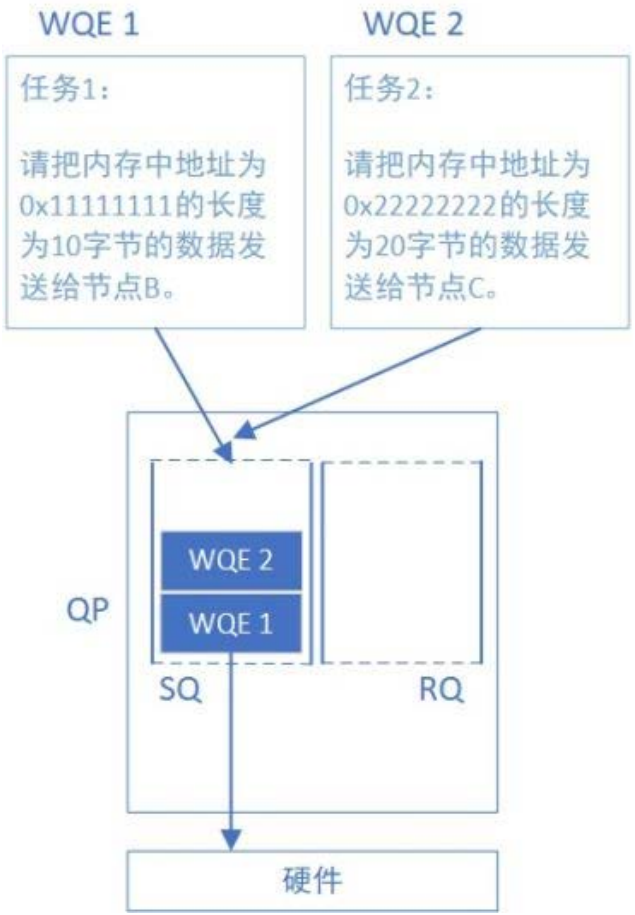
RDMA协议简介

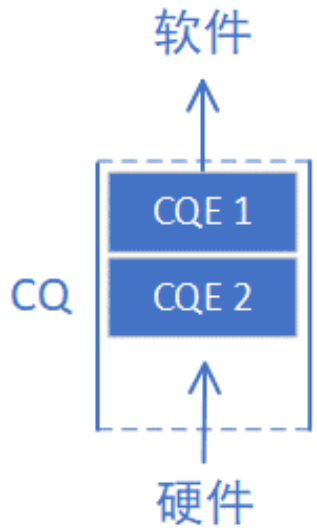
WQ（Work Queue）

QP（Queue Pair）

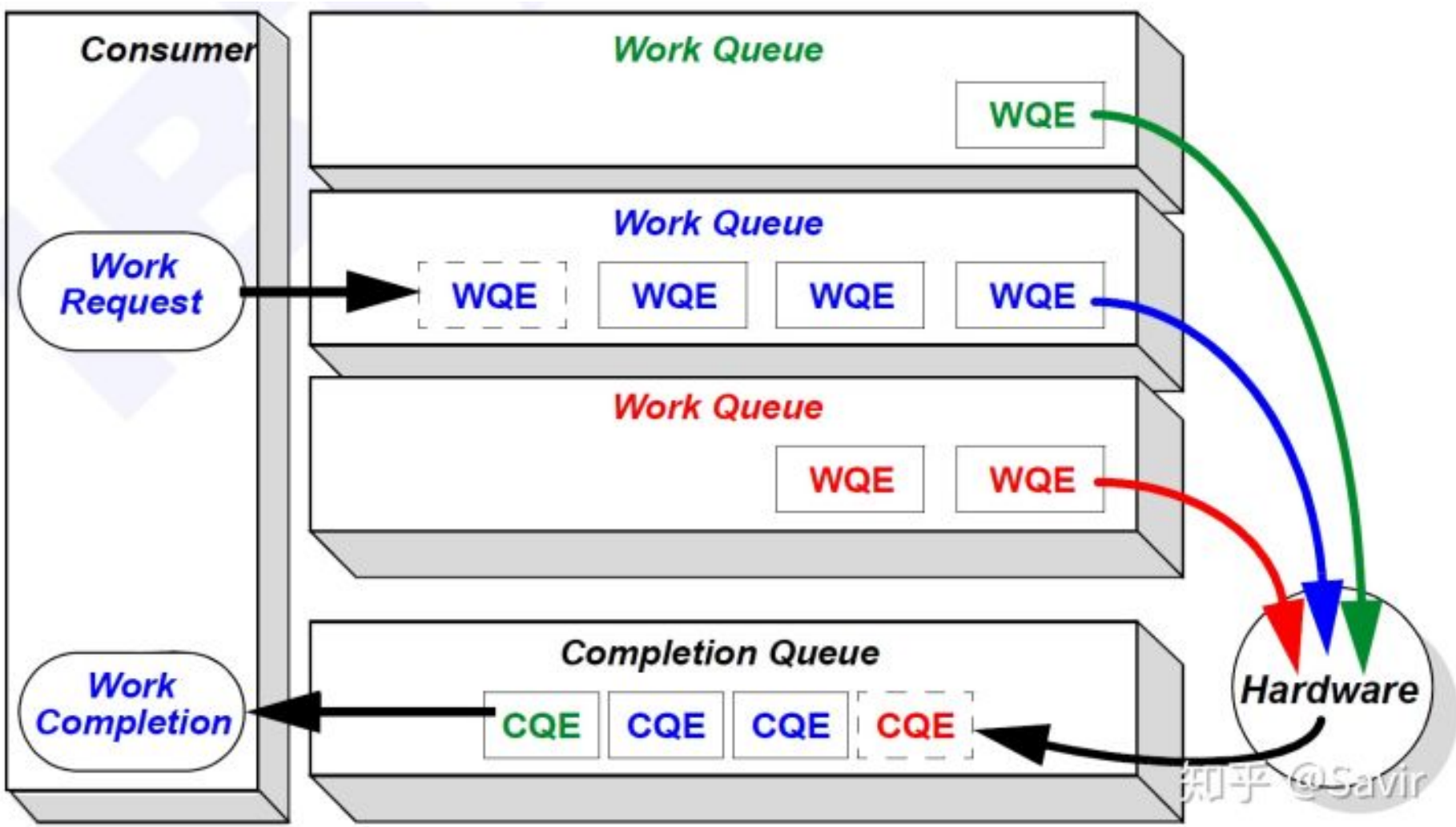
CQ（Completion Queue）





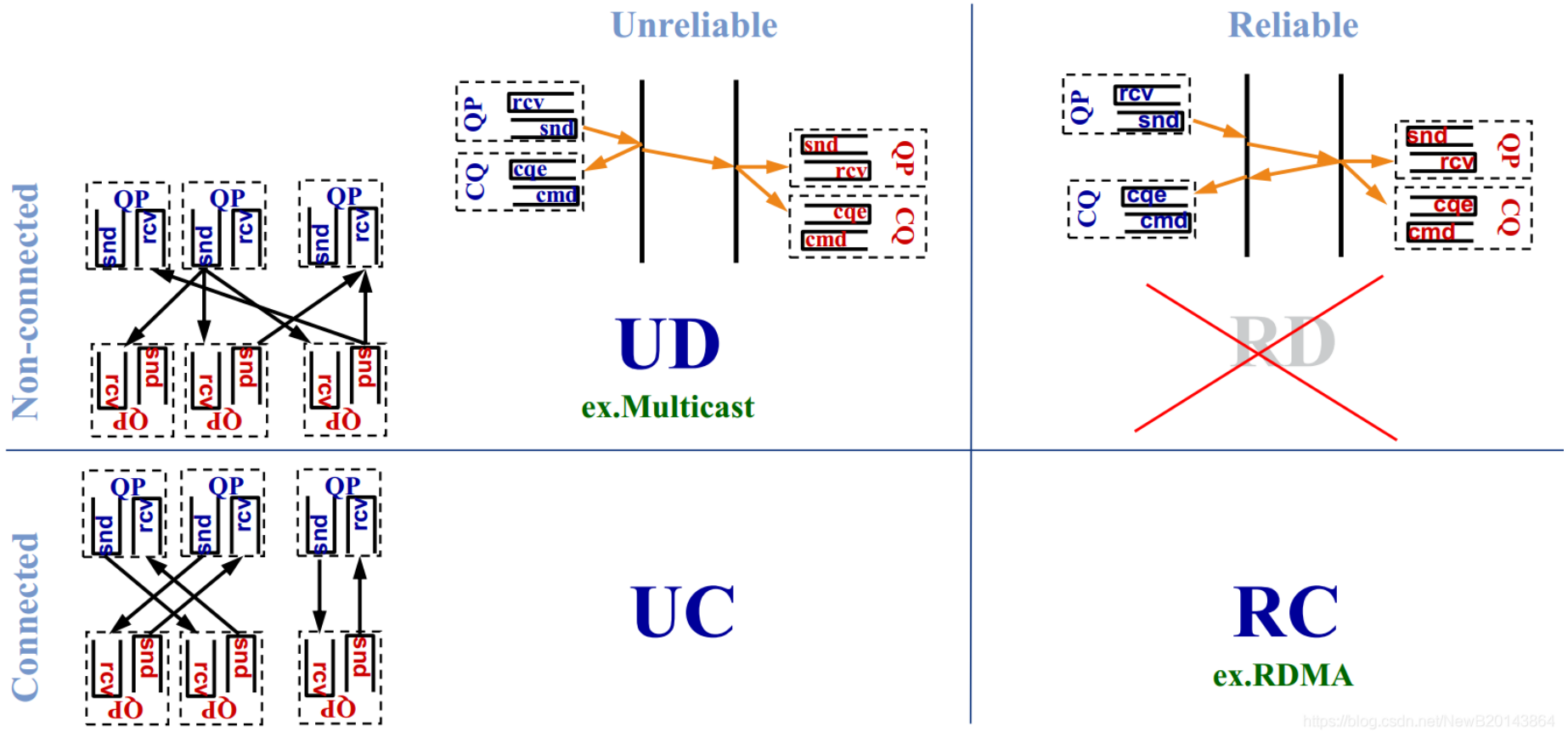


RDMA协议简介


Consumer
Work Queue
WQE
Work Queue
Work Request
WQE
WQE
WQE
WQE
Work Queue
WQE
WQE
Completion Queue
Work Completion
CQE
CQE
CQE
CQE
Hardware
知乎 @Savir

RDMA协议简介

RDMA协议定义RC、UC、UD三种通信模式。



RDMA协议简介

RDMA协议定义了双边、单边2种通信原语。

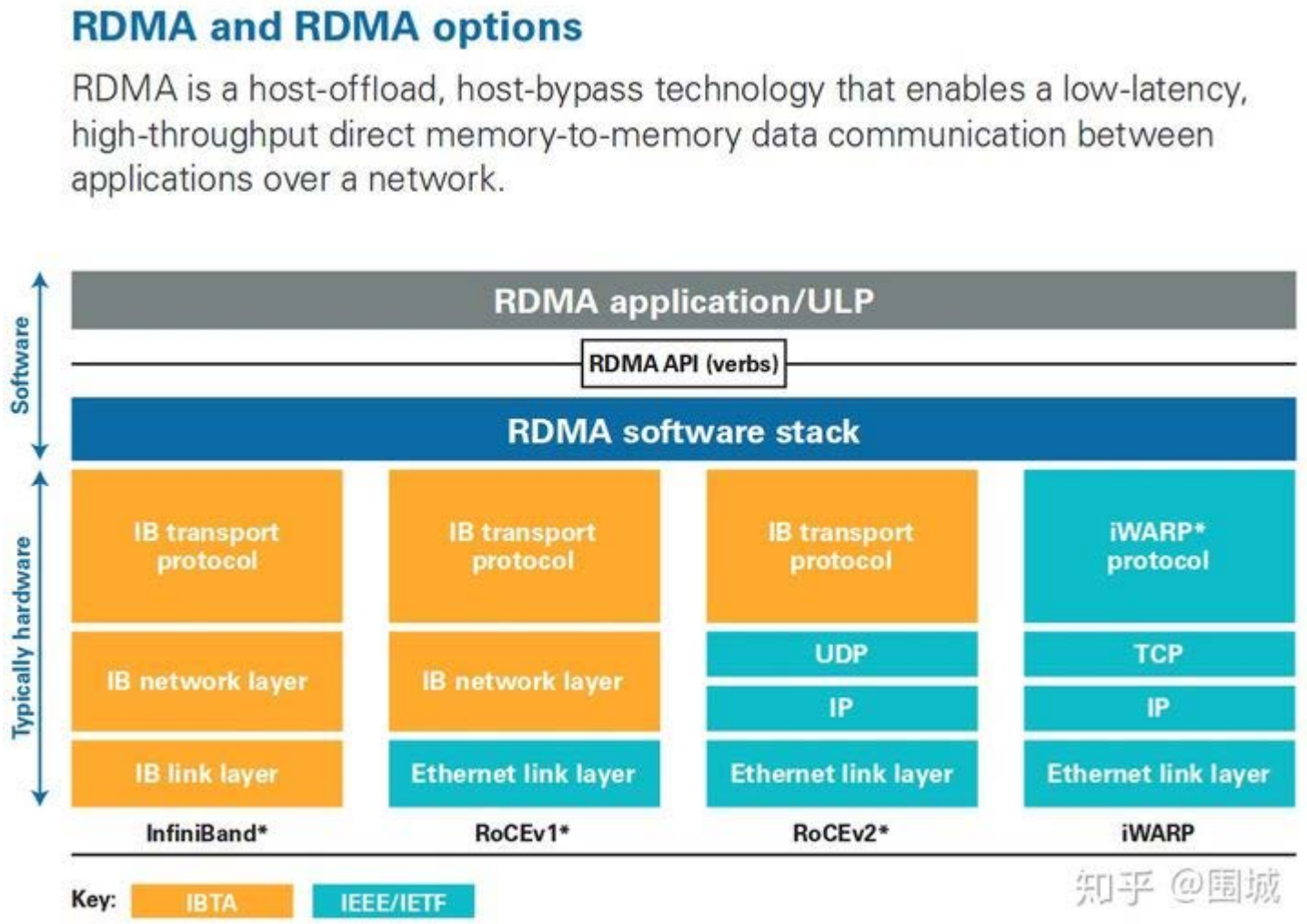
	UD	UC	RC	RD
Send (with immediate)	✓	✓	✓	X
Receive	✓	✓	✓	X
RDMA Write (with immediate)		✓	✓	X
RDMA Read			✓	X
Atomic: Fetch & Add / Cmp & Swap			✓	X
Max message size	MTU	2GB	2GB	2GB

单边原语是RDMA规范中最具创新性的特性，通过RDMA协议把本地内存总线延伸到其他主机，传输效率高，适合较大数据的传输。

RDMA协议简介

协议类型：

- Infiniband
- RoCE
  - RoCEv1
  - RoCEv2
- iWARP



RDMA协议简介

协议类型：

- Infiniband
- RoCE
  - RoCEv1
  - RoCEv2
- iWARP

RDMA 三种实现的对比

对比项	InfiniBand	iWarp	RoCE
标准组织	IBTA	IETF	IBTA
性能	最好	稍差	与 InfiniBand 相当
成本	高	中	低
网卡厂商	Mellanox-40Gbps	Chelsio-10Gbps	Mellanox-40Gbps Emulex-10/40Gbps

目录

- 研究背景介绍
- RDMA协议简介
- 研究方向介绍
  - RDMA原语优化
  - 基于RDMA的分布式组件KV存储
  - 使用RDMA加速数据库通信层
  - 基于RDMA和NVMM的分布式文件系统
- 总结

研究方向介绍

RDMA相关工作主要可以分为五大类

研究方向	研究内容
原语使用	探寻RDMA的最优使用方法：单、双边原语的选择，原语使用方法优化等;
架构设计	引入RDMA后的分布式系统架构探讨——共享内存、存算分离等;
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;
非网络相关算法的重构	由于RDMA转移了网络瓶颈，需要重新设计其他部分的相关算法，比如数据分区算法的重写等;
多种新硬件结合	将RDMA和其他新硬件，如非易失存储器（Non-Volatile Memory，简称NVM）等。而新硬件的结合实际上又可分为以上四个方面的研究。

目录

- 研究背景介绍
- RDMA协议简介
- 研究方向介绍
  - RDMA原语优化
  - 基于RDMA的分布式组件KV存储
  - 使用RDMA加速数据库通信层
  - 基于RDMA和NVMM的分布式文件系统
- 总结

RDMA原语优化

Using One-Sided RDMA Reads to Build a Fast, CPU-Efficient Key-Value Store *USENIX ATC '13*

PRISM: Rethinking the RDMA Interface for Distributed Systems *SOSP '21*

Pilaf

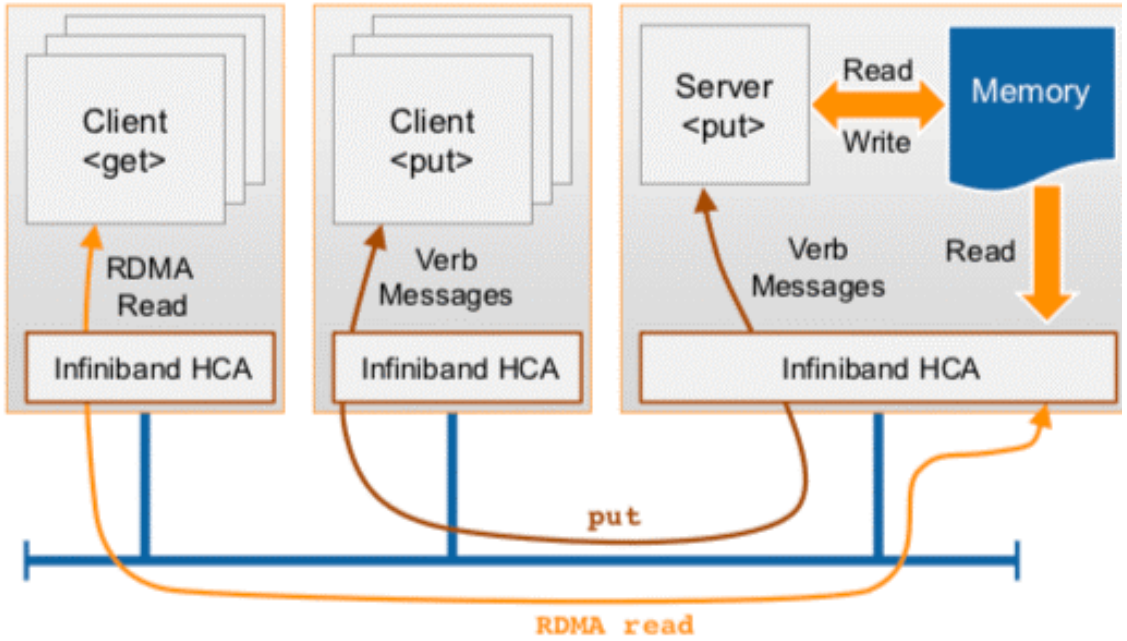


Figure 3: Pilaf restricts the clients' use of RDMAs to read-only `get` operations and handles all `puts` at the server using verb messages.

Get这种只读请求由客户端使用单边RDMA原语获取

Put、Delete等请求仍由服务端进行处理

RDMA原语优化

Using One-Sided RDMA Reads to Build a Fast, CPU-Efficient Key-Value Store *USENIX ATC '13*

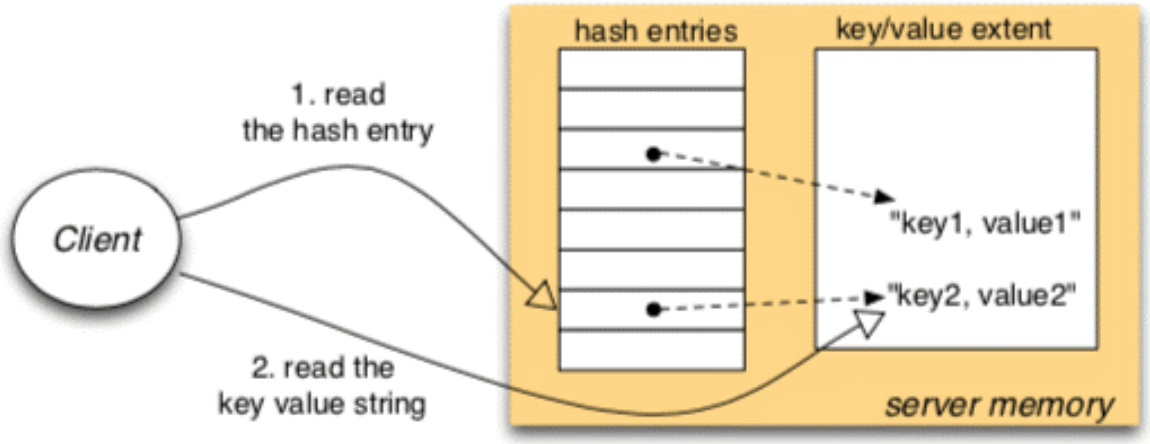


Figure 4: The memory layout of the Pilaf server's hash table. Two memory regions are used, one contains an array of fixed-size hash table entries, the other is an extent storing variable sized key-value extents. Clients perform `get` operations in two RDMA reads, first fetching a hash table entry, then using the address information in that entry to fetch the associated key-value string.

Get请求分两步完成：

Step1：从服务端的固定内存区域中读取布谷鸟过滤器中的值；

Step2：根据得到的地址偏移再发起一次单边读，从而完成get操作。

最好情况下需要发起两次单边读。

RDMA原语优化

可能存在的并发性问题：

1. 由于布谷鸟过滤器的实现特性，如果在读取过程中一个条目被更改，可能会下一跳的目的地发生变化，从而导致读取到错误的数据；

2. 如果在读取一个hash条目的过程中对其进行写入，可能还会导致条目指向目标kv的地址被读取错误，从而访问到无效的内存

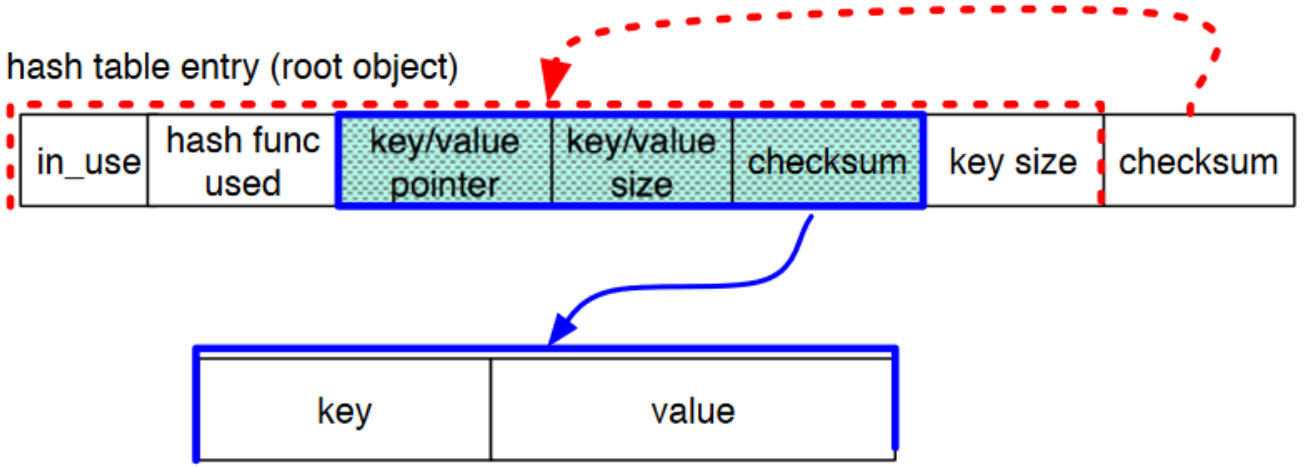
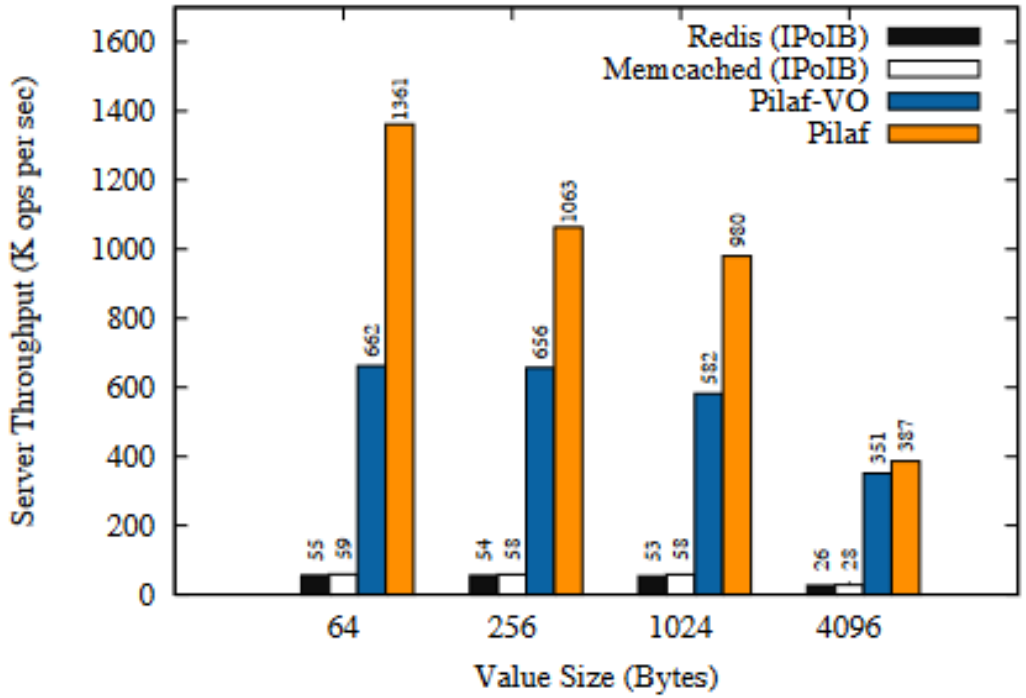
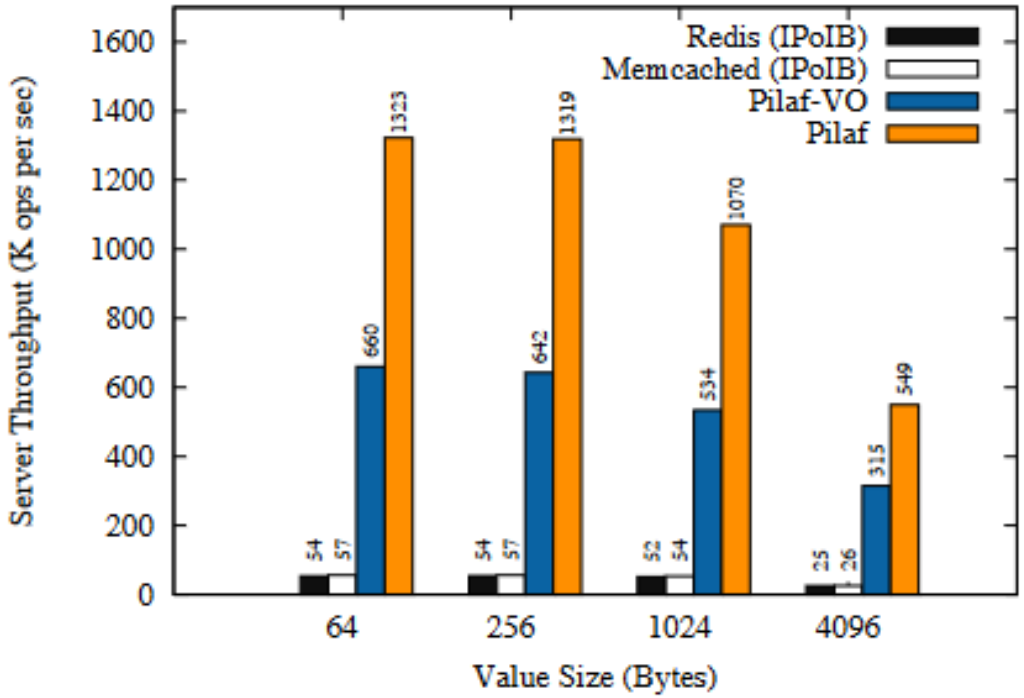


Figure 5: *Self-verifying* hash table structure. Each hash table entry is protected by a checksum. Each entry stores a self-verifying pointer (shown in shaded fields) which contains a checksum covering the memory area being referenced.

RDMA原语优化



(a) Peak throughput (90% gets, 10% puts)



(b) Peak throughput (50% gets, 50% puts)

Figure 10: Throughput achieved on a single CPU core for Pilaf, Pilaf-VO, Redis, and Memcached.

RDMA原语优化

PRISM: Rethinking the RDMA Interface for Distributed Systems *SOSP '21*

Trade-off：两次网络往返 or CPU参与？

这篇文章使用两台40GB以太网连接的服务器测试了单边RDMA和双边eRPC的性能。单边读512B数据需要3.2us，双边RPC需要5.6us，这意味着使用两次单边读要比使用一次RPC慢0.8us。

这说明，单边RDMA操作只有在不需要更复杂的协议时才会提供性能优势。

RDMA原语优化

作者思考了需要哪些新功能来支持在RDMA上运行的分布式系统，
从而不需要CPU参与或进行额外的网络往返。

功能	描述
定位数据结构	大多数应用程序，使用复杂的数据结构来构建索引，存储变长对象。遍历这些结构需要多次RDMA读取。如果能够在指针上执行间接操作可以消除一些往返。
支持异地更新	将新数据写入到一个单独的缓冲区中，然后自动更新指针以从旧值交换到新值（类似并发编程中的Copy-On-Write）
乐观并发控制	扩展RDMA的CAS（Compare-and-Swap）功能，允许实现复杂的、基于版本的乐观并发控制，这种方法很适合异地更新方法。
链式操作	将有前后关系的操作链接起来，使一个操作依赖于另一个操作，而只在一个往返流程中执行它们。

RDMA原语优化

PRISM（Primitives for Remote Interaction with System Memory）为现有RDMA接口增加了四个额外的特性，摘要如图。

Indirect Reads and Writes:

- READ(ptr *addr*, size *len*, bool *indirect*, bool *bounded*) → byte[]
  Returns the contents of the target address – either *addr* or, if *indirect* is set, **addr*.
  If *bounded* is set, reads min(*len*, *addr->bound*) bytes; otherwise reads *len* bytes.
- WRITE(ptr *addr*, byte[] *data*, size *len*, bool *addr_indirect*, bool *addr_bounded*, bool *data_indirect*)
  Writes data to a target address – either *addr* or, if *addr_indirect* is set, **addr*.
  If *addr_bounded* is set, writes min(*len*, *addr->bound*) bytes; otherwise writes *len* bytes.
  If *data_indirect* is set, *data* is interpreted as a server-side pointer and its target is used as the source data.

Allocation:

- ALLOCATE(qp *freelist*, byte[] *data*, size *len*) → ptr
  Pops the first buffer *buf* from the specified free list (represented as a RDMA queue pair). The specified *data* is written to *buf*, and the address of *buf* is returned.

Enhanced Compare-and-Swap:

- CAS(*mode*, ptr *target*, byte[] *data*, bitmask *compare_mask*, bitmask *swap_mask*, bool *target_indirect*, bool *data_indirect*) → byte[]
  Atomically compares (**target* & *compare_mask*) with (*data* & *compare_mask*) using the operator specified by *mode*, and, if successful, sets **target* = (**target* & ~*swap_mask*) | (*data* & *swap_mask*). Returns the previous value of **target*.
  If *target_indirect* or *data_indirect* is set, the corresponding argument is first treated as a pointer and dereferenced; this is not guaranteed to be atomic.

Operation Chaining:

- CONDITIONAL: Executes operation only if previous operation was successful. Operations that generate NACKs or errors, or CAS operations that do not execute, are considered unsuccessful.
- REDIRECT(*addr*) Instead of returning operation's output to the client, write it to *addr* instead.

Table 1. PRISM Primitives

RDMA原语优化

论文设计了三个具有代表性的分布式应用来说明PRISM的优势，分别是：

- PRISM-KV：一个键值存储，在RDMA上实现读和写操作；
- PRISM-RS：实现ABD仲裁复制协议的复制存储系统；
- PRISM-TX：一个事务性存储系统，它使用PRISM的原语实现基于时间戳的乐观并发控制协议。

这里仅展示PRISM-KV的测试结果，依次是YCSB-C（100% Read）和YCSB-C（50% Read/50% Write）负载的实验结果。其他实验可见原文。

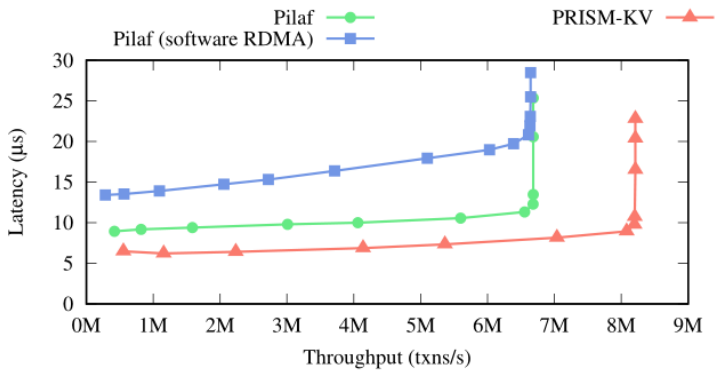


Figure 3. Throughput versus average latency comparison for PRISM-KV and Pilaf, 100% reads, uniform distribution.

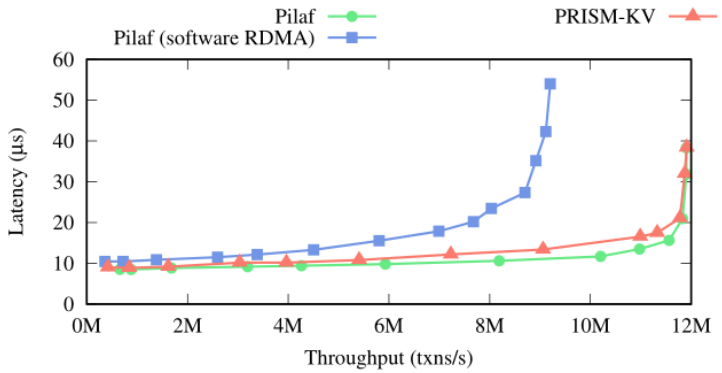


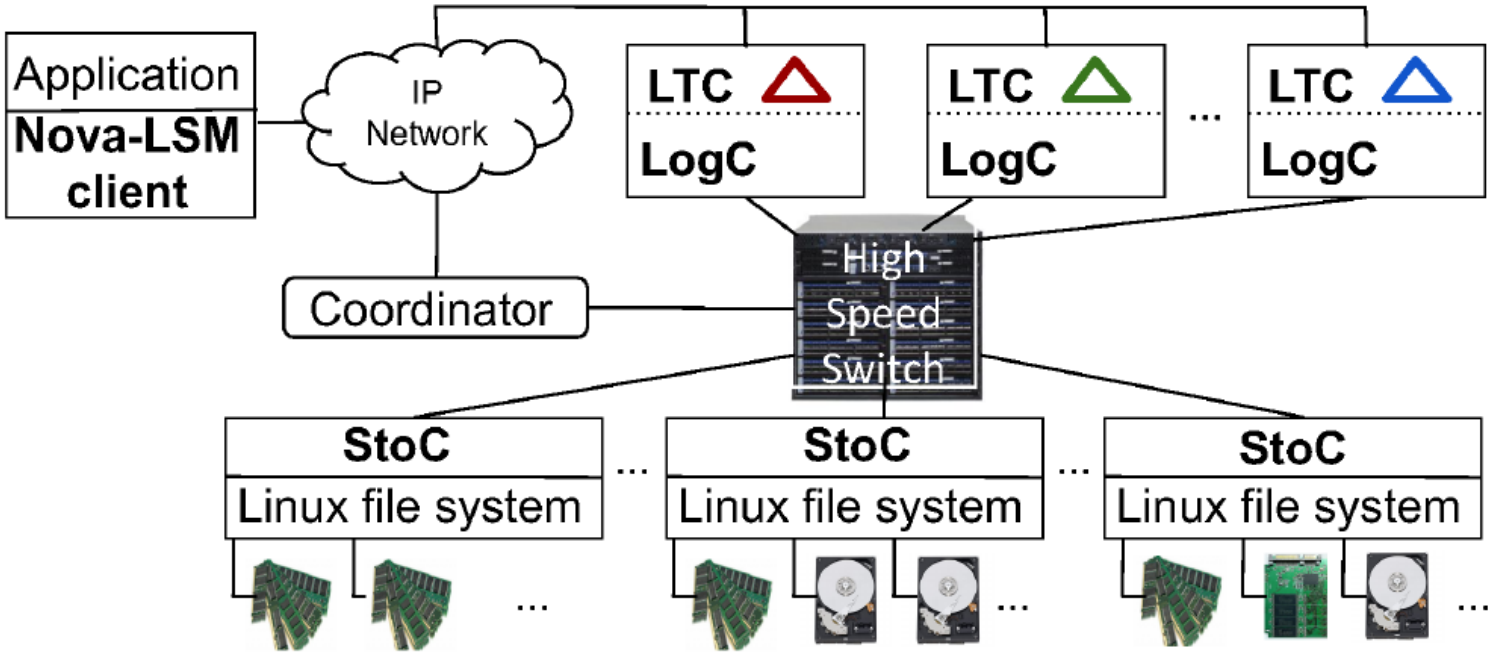
Figure 4. Throughput versus average latency for PRISM-KV and Pilaf, 50% reads, uniform distribution.

目录

- 研究背景介绍
- RDMA协议简介
- 研究方向介绍
  - RDMA原语优化
  - 基于RDMA的分布式组件KV存储
  - 使用RDMA加速数据库通信层
  - 基于RDMA和NVMM的分布式文件系统
- 总结

基于RDMA的分布式组件KV存储

Nova-LSM: A Distributed, Component-based LSM-tree Key-value Store *SIGMOD 21'*



NovaLSM主要分为三个组件:

- Logging Component (LogC)
- LSM-tree Component (LTC)
- Storage Component (StoC)

基于RDMA的分布式组件KV存储

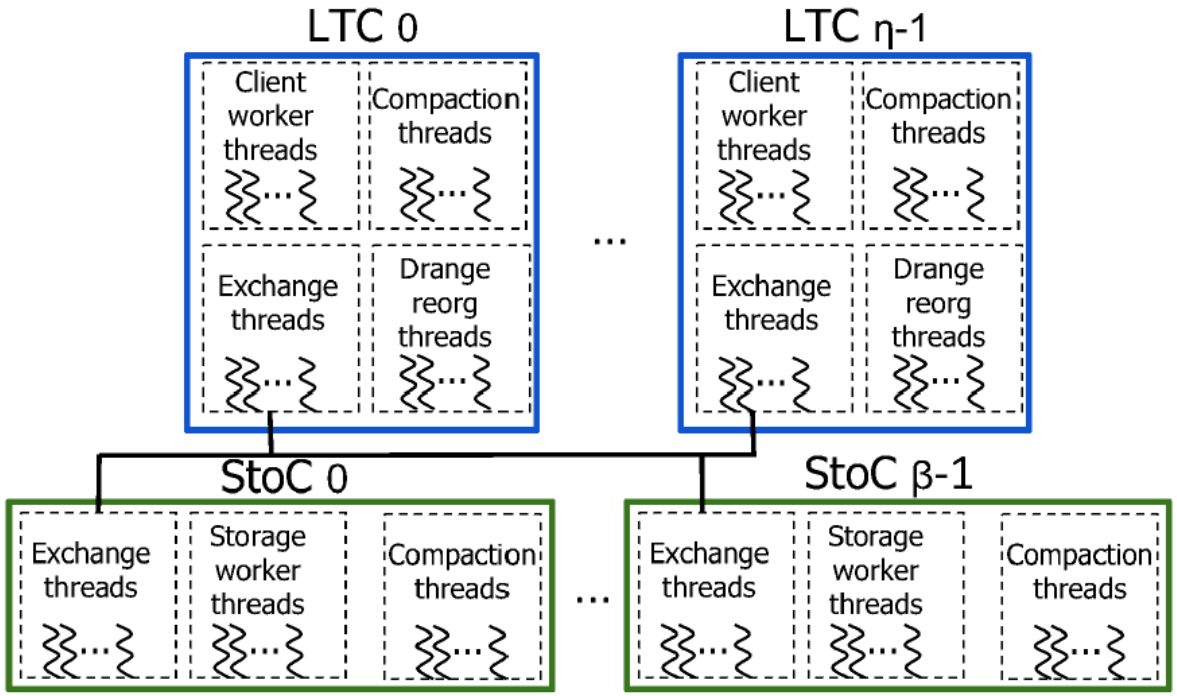
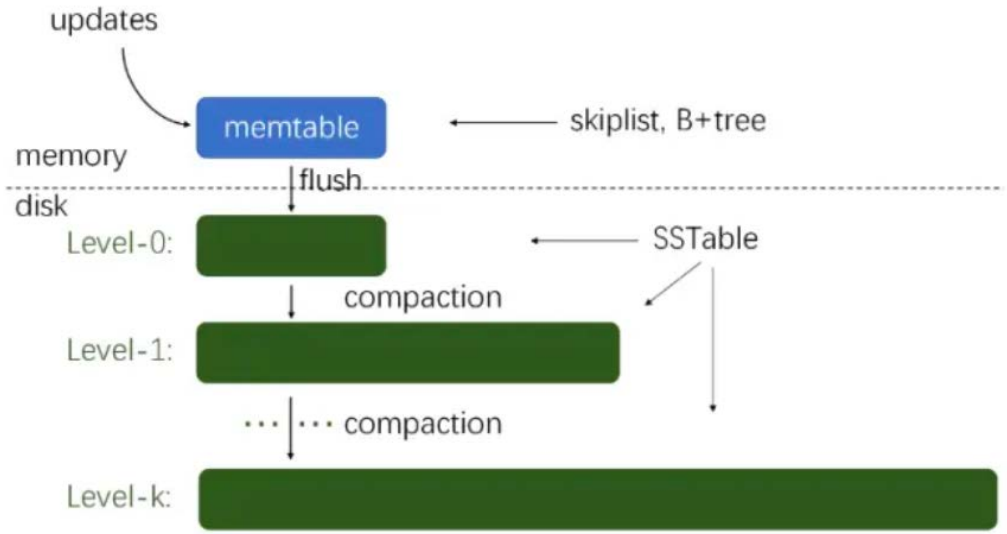


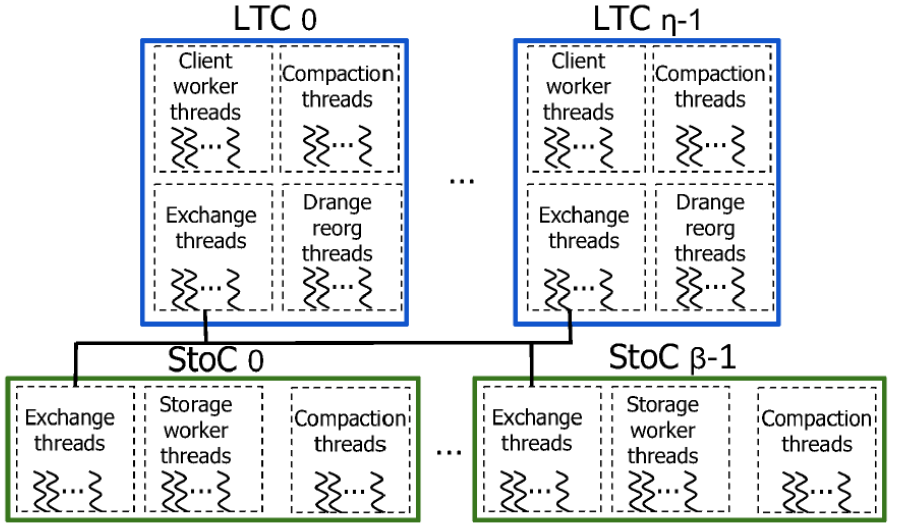
Figure 5: Thread Model.

- 将基于LSM-Tree的KV存储系统几个核心组件单独组件化——存算分离;
- 组件之前通过RDMA统一接口交互，均可独立扩缩容;
- 每个组件中按功能职责划分线程池，为精细化调优提供基础;

基于RDMA的分布式组件KV存储



磁盘IO的瓶颈导致停写、缓写等现象的出现。



采用存算分离的架构灵活增加节点来提高瓶颈部分的能力

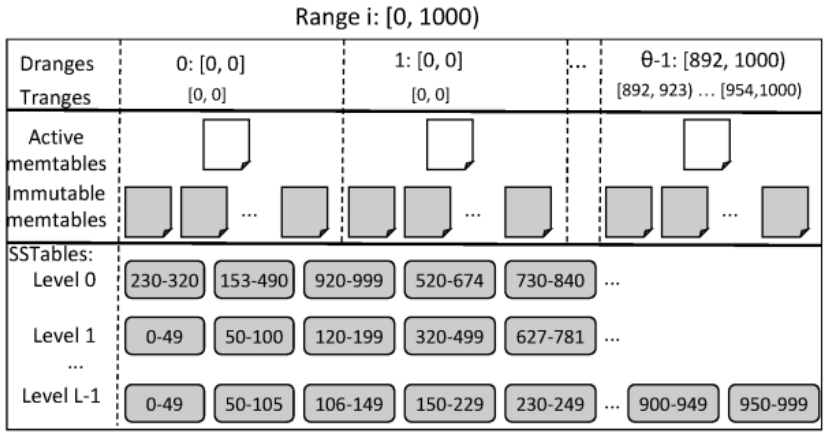
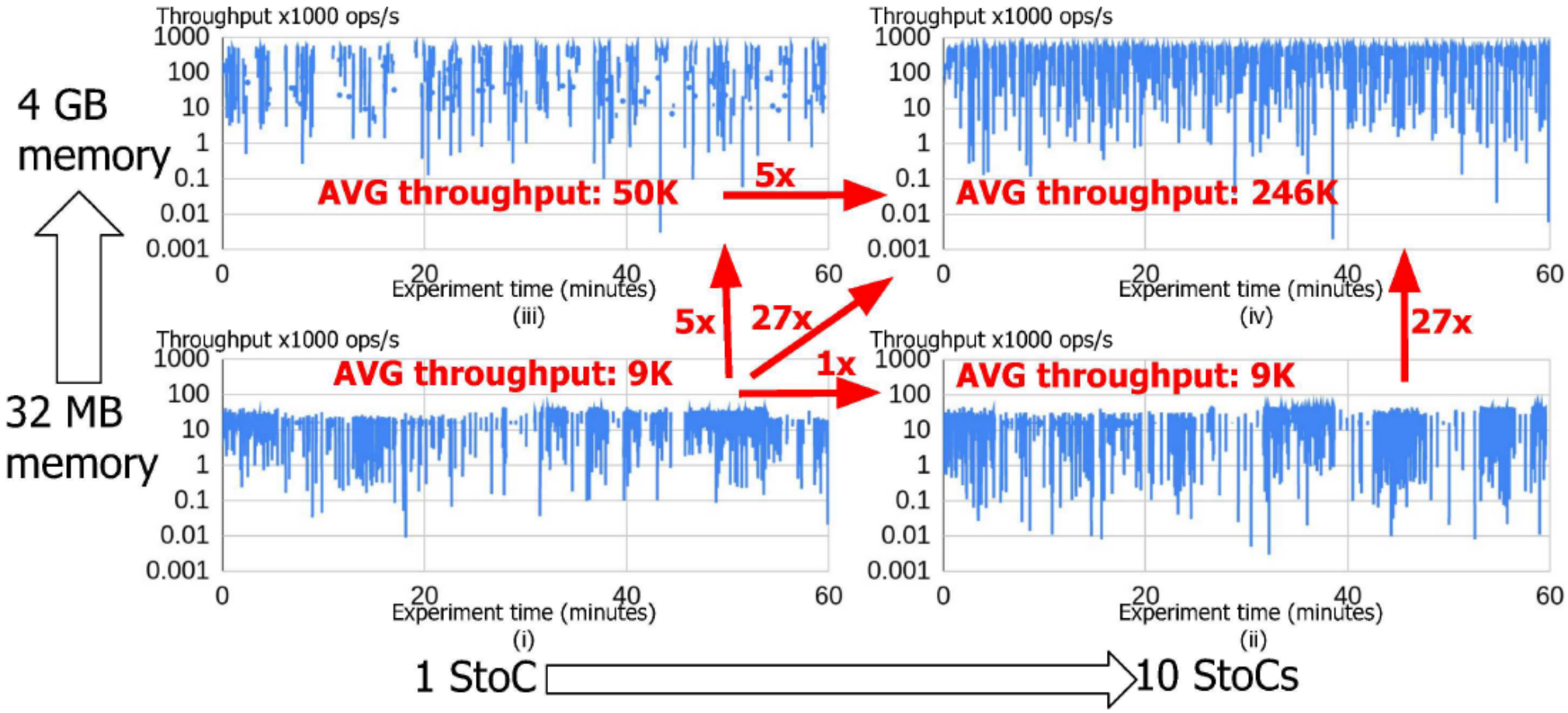


Figure 6: LTC constructs θ Dranges per range.

采用动态分区加速大文件的消化。

基于RDMA的分布式组件KV存储



基于RDMA的分布式组件KV存储

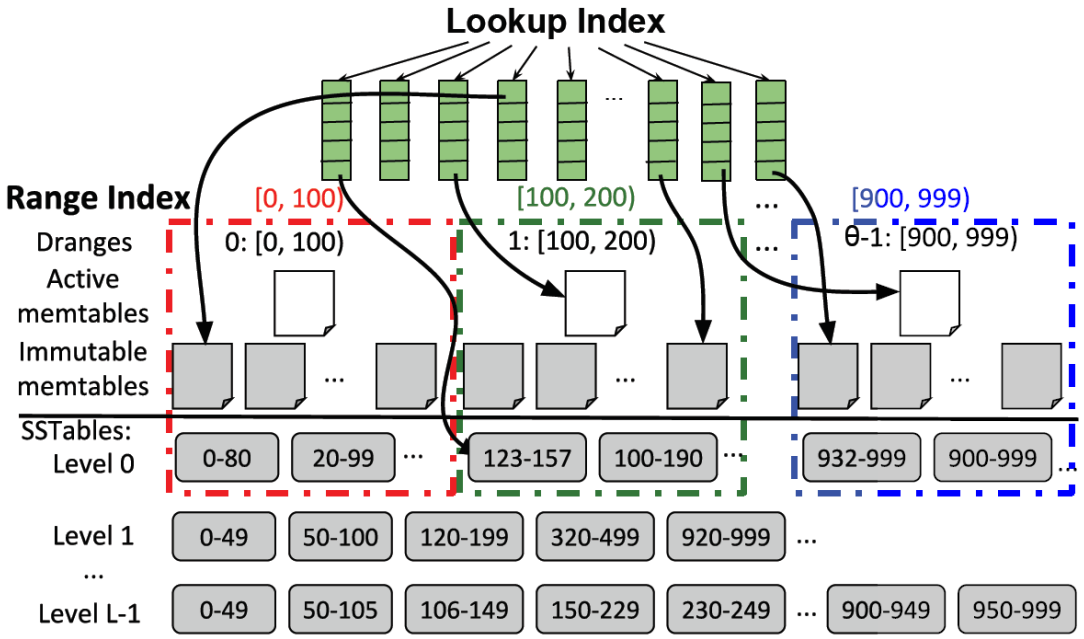


Figure 5: Lookup index and range index.

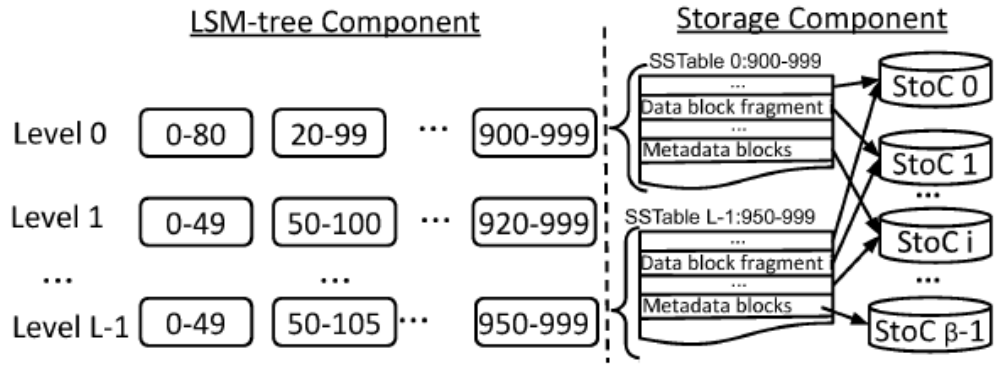


Figure 9: LTC scatters a SSTable across multiple StoCs.

目录

- 研究背景介绍
- RDMA协议简介
- 研究方向介绍
  - RDMA原语优化
  - 基于RDMA的分布式组件KV存储
  - 使用RDMA加速数据库通信层
  - 基于RDMA和NVMM的分布式文件系统
- 总结

总结

研究方向	研究内容
原语使用	探寻RDMA的最优使用方法：单、双边原语的选择，原语使用方法优化等；
架构设计	引入RDMA后的分布式系统架构探讨——共享内存、存算分离等；
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；
非网络相关算法的重构	由于RDMA转移了网络瓶颈，需要重新设计其他部分的相关算法，比如数据分区算法的重写等；
多种新硬件结合	将RDMA和其他新硬件，如非易失存储器（Non-Volatile Memory，简称NVM）等。而新硬件的结合实际上又可分为以上四个方面的研究。

参考文献

[1] PRISM: Rethinking the RDMA Interface for Distributed Systems

[2] Using One-Sided RDMA Reads to Build a Fast, CPU-Efficient Key-Value Store

[3] Low-Latency Communication for Fast DBMS Using RDMA and Shared Memory

[4] Orion: A Distributed File System for Non-Volatile Main Memories and RDMA-Capable Networks

[5] Nova-LSM: A Distributed, Component-based LSM-tree Key-value Store

谢谢