哈爾濱工業大學

毕业设计（论文）中期报告

题 目：基于 **RDMA** 与 **NVM** 的低延迟分布式存储系统

专　　业　计算机科学与技术

学　　生　张婉茹

学　　号　**1170500116**

指导教师　王宏志教授

日　　期　**2021.04.05**

哈尔滨工业大学教务处制

目录

1．论文工作是否按开题报告预定的内容及进度安排进行

1.1 研究概述

基于 RDMA 和 NVM 分布式键值存储系统的设计：该设计以降低请求延迟为主要目标。同时，充分发挥非易失性内存的持久化、可字节寻址等特性，降低系统成本和资源开销。

同时针对一致性问题，调研前人设计的算法（Raft[1][2][3],Paxos[4]等），并针对这些算法在 RDMA 和 NVM 上会遇到的问题，阅读部分论文，这些前期工作修改了协议流程中对一致性成功的判定方式以及领导者选举和日志恢复中的部分流程，利用 RDMA Write 操作的绕过目标服务器 CPU 的特性、NVM 的持久性和可字节寻址的特性，使得修改后的协议具有低时延、高吞吐量和低 CPU 消耗等特点。针对上述工作，本次毕业设计尝试提出更好的解决方案。

1.2 研究进度

开题报告进度安排及完成情况如表 1- 1 所示：

表 1-1 进度安排表

时间	安排	完成情况
2020.11-2021.12	进行前期准备，阅读相关资料和文章[5][6][7][8][9][10][11]，了解 RDMA 编程和非易失性内存的概念。	已完成
2020.01-2020.02	设计本文系统的总体框架，以及各部分模块的实现方式。	已完成
2020.02-2020.04	在分布式节点上搭建 KV 存储系统，通过 RDMA 进行通信，使用 NVM 存储数据。	已完成
2020.04-2020.05	主备份节点间数据复制的实现，对一致性问题进行初步的研究和思考。	未完成
2020.05-2020.06	和前期工作设计实现的系统作对比实验，撰写毕业论文，并准备毕业答辩。	未完成

1.3 是否按照预定的内容及进度安排进行

本次毕业设计，严格按照开题报告预定的时间完成对应的内容，并根据不断出现的新问题及时寻找解决办法，到目前为止，各时间段均完成了阶段性的成果。目前已完成系统的总体框架的设计，并搭建了一个基于 RDMA 和 NVM 的分布式 KV 存储系统，并编写

了测试代码，对系统的稳定性，正确性，请求延迟性能进行了测试，并与将数据存储在普通磁盘的系统延时性能做了对比。

2．已完成的研究工作及成果

2.1 RDMA 和 NVM 的技术分析

为了实现读写请求低延时，数据访存高吞吐的系统性能，本文设计的分布式键值存储系统将数据存储于非易失性内存。与基于 DRAM 的传统存储系统设计相比，由于非易失性内存的持久性，系统节省了将部分数据刷新到次级持久存储介质（磁盘或固态硬盘）的 IO 时间；同时，非易失性内存具有与动态随机存取内存相近的读写速度。这些优点使得非易失性内存非常适合用于构建低延迟的分布式存储系统。

远程直接内存访问技术（RDMA）是本文设计的低延迟分布式存储系统的关键技术之一，其提供直接读写远程计算机内存地址空间的能力。RDMA 传输技术绕过了操作系统内核的网络协议栈，并且无需接收端的 CPU 参与，因此其具有延迟低的优点。RDMA 技术支持在无需对方计算机 CPU 参与的情况下直接读写对方计算机的内存地址空间，且数据传输是零拷贝的，这大大降低了网络传输的延迟和系统整体 CPU 的性能消耗。

RDMA 包括四种原语，RDMA 读（READ），RDMA 写（WRITE），RDMA 发送（SEND）和 RDMA 接收（RECV），这四种 verb 可以分为两种语义：内存语义和通道语义。

所谓内存语义，是指应用程序能够像访问本地内存地址空间一般，直接读写远程计算机的内存地址空间。RDMA-READ 和 RDMA-WRITE 原语就是内存语义。在指定所需读写的对端计算机的内存地址后，内存语义 verb 就可以在无需对方接收端 CPU 参与的情况下完成内存读写。此过程由于无需接收端 CPU 的参与，消除了接收端线程调度的时间，从而使得延迟时间大为降低。这使得内存语义在延迟上优于通道语义。此外，内存语义比通道语义的带宽更高。

所谓通道语义，是指需要通过发送消息到对方计算机，并由对方计算机将所需读写的内存地址的内容或状态返回给请求方。RDMA-SEND 和 RDMA-RECV 就是通道语义 verb。在一次请求中，通道语义要求被请求的一方需要先向接收队列中发送一个 RDMA RECV，然后才能处理请求放发送的请求。与内存语义不同的是，通道语义需要接收端的 CPU 参与，因此具有较高的延迟和较低的带宽。

2.2 总体架构设计

如图 2-1 所示为系统整体架构。集群中的服务器节点，客户端均通过支持 RDMA 的 Infiniband 网络连接。在本文的系统中，数据以键值对（Key/Value pair）的形式分布在集群中的服务器节点的非易失性内存中。我们使用 Key 的哈希值作为键值数据的索引。这是一个 32 位二进制非负整数， 并为每个服务器分配其所管理的哈希区间。为便于对存储区域进行管理，本文所设计的 Key size 为 16 字节，Value size 是变长的。

考虑到通道语义原语的具有较高的延迟和较低的带宽，而内存语义的原语能够实现

CPU bypass，从而有效降低接收端 CPU 的利用率，进而提升系统性能。故本文设计采用基于内存语义的 RDMA 单边访问操作进行网络传输，具体设计见图 2- 1 中，Client1 和 Server1 的传输方式。

get 和 put 请求都分两步完成。以 put 为例，Client 端先通过 send 原语发送 Key 和 Value 的大小，Server 端根据哈希值查找哈希表，返回应该写入的非易失性内存的地址，Client 根据这个地址通过 Write 原语直接写入远程客户端的非易失性内存上，无须通知远程客户端，从而减少了不必要的 Value 的复制的时间浪费。由于本文设计的 Value 是变长的，所以系统进行 Value 复制的时间开销需要考虑。

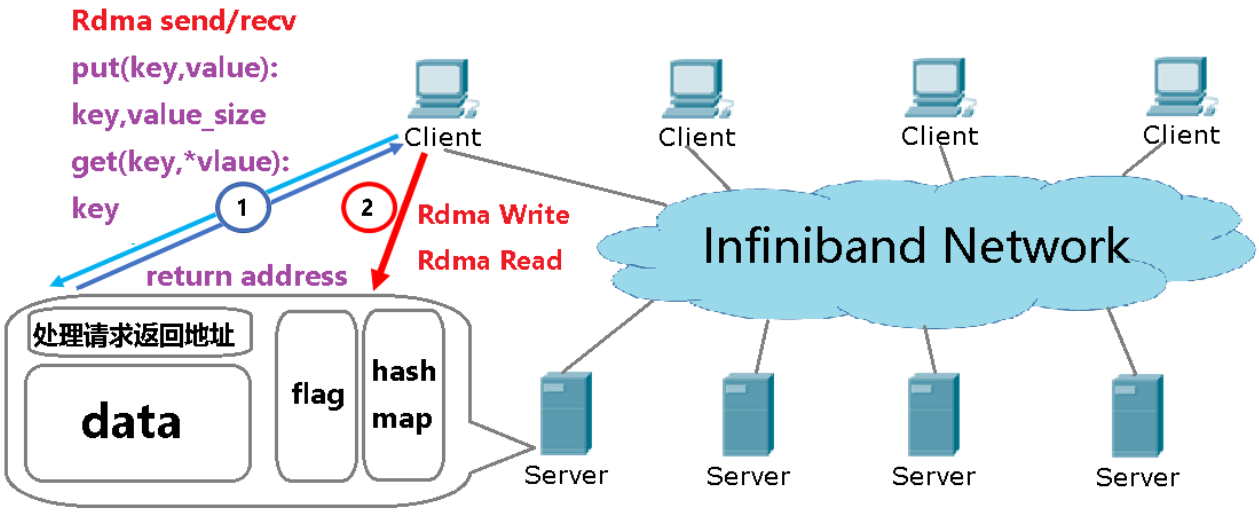

图 2- 1 系统整体架构

本章的余下内容将分小节地介绍系统中的各个关键模块。

2. 3 数据模型与空间管理

本系统提供键值数据模型。Key 的哈希值决定了键值对最终存储的服务器。一个存储在非易失性内存中的 record 结构如下所示 Value_size 是 32 位的无符号二进制数，Key 是 16 个字节的字符串，接下来是不定长的 Value 字符串。

代码 2- 1 键值对的结构体定义

```
1.  struct record{
2.      uint32_t Value_size;
3.      char[16] key;
4.      char[Value_size] Value;
5.  }
```

NVM 的内存管理如下图，首先，把内存分成若干长度为 64 字节的 block，通过系统设计的内存分配器，在非易失性内存中创建一段连续的地址空间，大小为 N 个 block。其次，把内存分成三个部分，标志区，数据区和哈希表。

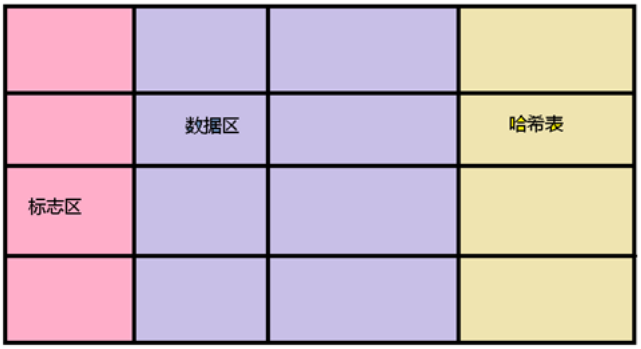

图 2- 2 内存管理

各区功能如下表。

表 2- 1 各区域功能

标志区	把 NVM 的内存分成若干个 64 字节的 block，标志区用来标志哪些内存被占用
数据区	用来存储 Key Value 的 record
哈希表	用来存一个固定大小的 hash 表

哈希表是一个定长的数组，数组的元素都是如下代码中所示的 Node 结构体。哈希表整体上是采用开链法解决冲突。hash_flag 数组用来记录 hashmap 哪些位置被占用，哪些没有，Entry 是指向哈希链表每个哈希值的头节点的位置。

```
typedef struct Node
{
  char key[KEY_LEN];
  BLOCK_INDEX_TYPE loc;
  VALUE_LEN_TYPE record_len;
  HASH_INDEX_TYPE next;
}Node;
Node hashmap[KV_NUM_MAX];
int hash_flag[KV_NUM_MAX];
int entry[HASH_MAP_SIZE];
```

管理着 NVM 的服务器端向应用程序提供标准的键值数据库接口。该接口由三个操作组成：GET、PUT、DELETE。其定义如下：

代码 2- 2 系统 API 定义

```
int NVM_Get(char * _key, uint64_t * record_len ,uint64_t * address);
int NVM_Put(char * _key, uint64_t record_len, uint64_t * address);
int NVM_Del(char * _key);
```

GET 操作向系统请求给定 Key 的 Value，PUT 操作将给定键值对存储到数据库中，DELETE 操作删除系统中指定 Key 的数据。

2. 4 客户端两阶段的请求方式

在本文设计的分布式存储系统的中，客户端与服务器节点之间的通信通过 RDMA 网

卡进行，分为 Send/Recv 和 Read/Write 两阶段，系统在客户端为应用程序封装好 GET，PUT，DELETE 的接口，供上层应用程序使用。

首先，客户端与所有的 Server 进行连接，就有了多个 qp，每个 qp 都有自己的 send_wr,recv_wr,read_wr,write_wr，以及对应的注册的内存区域。接下来，服务器端通过 send 原语向客户端发送 Server 的一些基本信息，如 Server 所管理的哈希区间，非易失性内存的基地址等。Client 端将这些信息都保存下来。服务器端是将非易失性内存通过 mmap 直接映射到虚拟内存空间中，给予客户端读写的权限。

当客户端 put(Key,Value)时，Client 端先使用与 Server 端相同的哈希算法计算出 Key 的哈希值，再找出对应的服务器节点，先 send（Key,record_len）,Server 端收到请求后，对消息进行解析，计算哈希值，向内存管理器请求 New 一块相应大小的内存，更新标志区的标志位，更新哈希表，将新的 Node 插入哈希链表，返回新分配的内存的相对地址。Client 在收到 address 后通过 Write 原语将 record 写入远程的服务器节点的非易失性内存上。

当客户端 get(Key,Value)时，Client 端先使用与 Server 端相同的哈希算法计算出 Key 的哈希值，再找出对应的服务器节点，先 send（Key,record_len）,Server 端收到请求后，对消息进行解析，计算哈希值，查找哈希表，如果存在，返回的 message 的第一位设置为 1，找到 record 在 NVM 上的位置，返回该 address。如果不存在，返回 0。Client 在收到 address 后通过 Read 原语将 record 从远程的服务器节点的非易失性内存上读出。

当客户端 del(Key)时，Client 端先使用与 Server 端相同的哈希算法计算出 Key 的哈希值，再找出对应的服务器节点，先 send（Key,record_len）,Server 端收到请求后，对消息进行解析，计算哈希值，查找哈希表，如果存在，返回的 message 的第一位设置为 1，将哈希表上的 node 节点删除，同时在使用 DELETE 操作释放该段内存地址空间。若不存在，则返回 0。

2.5 冗余复制

现有的分布式存储系统通过将多台普通的服务器通过网络进行连接，提供大容量、高性能和可扩展的存储服务。在一个包含 N 个服务器节点的集群中，假设单个服务器节点的故障率为 P，可靠率为 Q（Q = 1 − P）。那么在任意时刻，集群中有服务器故障的概率为 1 − (1 − P)N。此故障概率在 N 较大时是无法被忽略不记的。因此，现有分布式存储系统采用冗余复制的方式来提高系统的容错能力。所谓容错，即在集群中的某个服务器故障之后，仍能正常运转并提供服务。冗余复制将数据被分到多个独立的服务器节点中，并保持副本之间的一致性。在一个副本节点故障时，其它的副本节点仍能正常工作。对于一个保持 R 副本的系统，最多能够容忍 R − 1 个服务器节点同时故障。这将服务器节点的故障率降为大约 PN。这极大地提高了系统的可靠性，因而冗余复制成为分布式存储系统必不可少的设计要素。

系统冗余复制部分和一致性的保障的设计将在中期报告结束后继续进行。

2.6 实验结果

2.6.1 实验环境

操作系统为 Centos 7.9.2009，具体信息如下：

```
[wanru@AEP-43 ~]$ lsb_release -a
LSB Version:    :core-4.1-amd64:core-4.1-noarch
Distributor ID: CentOS
Description:    CentOS Linux release 7.9.2009 (Core)
Release:        7.9.2009
Codename:       Core
```

图 2-3 操作系统信息

RDMA 网卡为第五代网卡，基于 InfiniBand 网络，具体信息如下：

```
[wanru@AEP-43 ~]$ ibstat
CA 'mlx5_0'
        CA type: MT4119
        Number of ports: 1
        Firmware version: 16.27.2008
        Hardware version: 0
        Node GUID: 0xb8599f0300120f28
        System image GUID: 0xb8599f0300120f28
        Port 1:
                State: Active
                Physical state: LinkUp
                Rate: 100
                Base lid: 5
                LMC: 0
                SM lid: 9
                Capability mask: 0x2651e848
                Port GUID: 0xb8599f0300120f28
                Link layer: InfiniBand
```

图 2-4RDMA 网卡信息

非易失性内存为因特尔的 3DXpoint 内存条 [10][12][12]，具体信息如下：

```
Hostname:   AEP-43
CPU Model:  Intel(R) Xeon(R) Gold 6240M CPU @ 2.60GHz
Kernel:     4.13.0+
DRAM:       185 GB
AEP:        1514.726 GB
```

图 2-5 NVM 信息

2.6.2 实验结果

在 NVM 上，Value 长度固定为 1000 的时候，iops 稳定在 50000 次/s 到 60000 次/s，Value 长度在[0,1000]随机的时候，iops 稳定在 55000 次/s 到 60000 次/s。

在普通 SSD 上，Value 长度固定为 1000 的时候，iops 稳定在 40000 次/s，Value 长度在[0,1000]随机的时候，iops 稳定在 40000 次/s 到 45000 次/s。

为保证数据都保存的到了 SSD 上，而不是通过 Client 写入了虚拟内存中，服务器端需

要定时的进行 fsyncdata，而本文的设计消除了 fsync 操作，数据访存均在 NVM 中。因此，系统性能优于传统设计。

3．后期拟完成的研究工作及进度安排

3.1 后期拟完成的研究工作

尝试完成 MIT 的分布式系统的实验，对分布式系统中的可用性，容错性，一致性问题的解决方法有更深入的了解，进而对毕设的系统进行改进，最后做对比实验，完成毕设论文的撰写。

如果还有多余的时间，将修改本系统的内存分配器部分，目前现有的内存管理方式对非易失性内存的空间利用率不是很高，但本系统的主要目标是低延时，高吞吐，所以空间利用率较低也无大碍。

表 3- 1 后期进度安排

时间	安排	完成情况
2020.04-2020.05	主备份复制的实现，对一致性问题进行深入的研究和思考。	未完成
2020.05-2020.06	作对比实验，撰写毕业论文，准备毕业答辩。	未完成

3.2 进度安排

后续进度安排如表 3- 1 所示。

4．存在的问题与困难

当多个 client 端并发的进行操作时，操作原子性问题有待考虑，要给 server 端的数据加上远程的互斥锁。

5．论文按时完成的可能性

到目前为止，初步的分布式系统框架搭建和测试已经完成，并且取得了较好的性能。同时，对主备份复制和一致性问题也已有了初步方案，后续将进行写代码实现。目前工作进度已过半，暂无延期完成风险，可按时完成论文。

6．参考文献

[1] https://web.stanford.edu/~ouster/cgi-bin/papers/raft-atc14

[2] https://raft.github.io/

[3] http://thesecretlivesofdata.com/raft/

[4] https://www.microsoft.com/en-us/research/publication/paxos-made-simple/

[5] 刘志祥. 基于 RDMA 的非易失性内存文件系统设计与实现[D].重庆大学,2018.

[6] 周坚石. 基于非易失性存储器（NVM）的内存分配器的设计与实现[D].南京大学,2018.

[7] 董康平. 基于非易失性内存和 RDMA 的低延迟分布式键值存储系统的设计与实现[D].上海交通大学,2018.

[8] 陈波. 面向分布式非易失性内存的新型存储系统的设计与实现[D].江苏大学,2019.

[9] 刘昊. 面向非易失性内存的系统软件若干问题的研究[D].上海交通大学,2018.

[10]陈游旻,陆游游,罗圣美,舒继武.基于 RDMA 的分布式存储系统研究综述[J].计算机研究与发展,2019,56(02):227-239.

[11]Youyou Lu, Jiwu Shu, Youmin Chen, and Tao Li. 2017. Octopus: an RDMA-enabled Distributed Persistent Memory File System. In 2017 USENIX Annual Technical Conference (ATC '17). Santa Clara, CA, USA.

[12]WU H, CHEN K, WU Y W, ZHENG W M.Research on the consensus of big data systems based on RDMA and NVM. Big Data Research[J], 2019, 5(4):89-99

[13]https://www.intel.com/content/www/us/en/products/memory-storage/optane-dc-persistent-memory/optane-dc-128gb-persistent-memory-module.html