·先进计算与数据处理· 文章编号:1000-3428(2018)08-0048-06 文献标志码:A 中图分类号:TP393

基于 SDN 多播的分布式共享内存研究

张 雷,支小莉

(上海大学 计算机工程与科学学院,上海 200436)

摘 要: 在分布式共享内存(DSM)中使用多播进行节点间的消息传输,可以大幅节省通信开销,但传统多播具有不可控性。为此,提出一种将基于软件定义网络(SDN)实现的多播应用于 DSM 中的方案。设计一种基于多播的 DSM 系统,并以基于 SDN 实现的多播为其提供数据传输。实验结果表明,与传统以太网相比,该方法能够有效降低 DSM 中的数据传输延迟,为 DSM 系统提供良好的通信性能保障。

关键词: 分布式共享内存;多播;不可控性;软件定义网络;数据传输延迟

中文引用格式: 张 雷,支小莉. 基于 SDN 多播的分布式共享内存研究[J]. 计算机工程,2018,44(8):48-53,60.

英文引用格式: ZHANG Lei,ZHI Xiaoli. Research on distributed shared memory based on SDN multicast[J]. Computer Engineering,2018,44(8):48-53,60.

Research on Distributed Shared Memory Based on SDN Multicast

ZHANG Lei,ZHI Xiaoli

(School of Computer Engineering and Science,Shanghai University,Shanghai 200436,China)

【Abstract】 In Distributed Shared Memory(DSM),using multicast to transmit messages between nodes can greatly save communication costs,but traditional multicast is uncontrollable. To solve this problem,a scheme to apply multicast based on Software Defined Network(SDN) to DSM is proposed. A DSM system based on multicast is designed,and data transmission is provided for it with SDN based multicast. Experimental results show that,compared with traditional Ethernet,this method can effectively reduce the data transmission delay in DSM and provide good communication performance guarantee for DSM system.

【Key words】 Distributed Shared Memory(DSM); multicast; uncontrollable; Software Defined Network(SDN); data transmission delay

DOI: 10.19678/j.issn.1000-3428.0047852

0 概述

分布式共享内存(Distributed Shared Memory,DSM)是分布式系统节点之间进行数据共享与交换的一种快速、有效的机制,它能将不同节点的物理内存映射成一个逻辑上统一的虚拟地址空间,任意进程都可以访问该空间内的任意地址,进程间的消息传递对程序员而言是透明的。因此,DSM 系统的编程较简单,且其具有较好的扩展性。

由于分布式系统节点之间的网络通信延迟较大,具体实现时需要考虑在保证数据一致性的同时降低总访问延迟。文献[1-2]较早对 DSM 的具体实现方式进行细致分析,并提出一种以共享页面方式实现的 DSM 系统 Ivy。但 Ivy 存在数据颠簸和假共享的问题,性能较差。随着科技的进步,研究人员对 DSM 的实现形式进行改进,提出基于变量的共享方式,其代表系统是 Midway[3]。而目前,DSM 的普遍实现方式是采用基于对象的共享方式,如 X10、Grace 等[4-7]。

传统 DSM 系统采用基于以太网的点对点、广播或多播的方式进行节点间数据传输。其中,多播是一种效果较好的传输形式[8]。多播是一对多的通信方式,其可以只向特定多播组内的节点发送消息,能够有效节省通信开销,提高网络带宽利用率。传统多播是基于以太网的,但以太网只能提供不保证带宽与延迟的尽力而为的服务。因此,基于以太网的多播在带宽与延迟方面具有不可控性,需要使用额外策略为多播提供性能保障。目前常用方法是由路由器根据相应路由算法构造一棵覆盖源节点和目的节点的多播树。在传统以太网的多播树计算中,路由器只维护局部网络信息,由其计算出来的多播路径容易发生部分链路通信过载而部分链路却通信空

基金项目: 上海市科委科研计划项目(15DZ1100305);上海市科技创新行动计划项目(16511101200)。

作者简介: 张 雷(1992—),男,硕士,主研方向为并行计算;支小莉,副研究员。

收稿日期: 2017-07-06 **修回日期:** 2017-09-15 **E-mail:** xlzhi@shu.edu.cn

闲的状况,且路由器节点之间需要频繁交换信息,导致算法复杂度高、网络负载大。

软件定义网络(Software Defined Network,SDN)的提出,为解决多播的不可控性问题提供了一种新途径。SDN 是一种数据控制分离、软件可编程的新型网络体系架构,其采用集中式的控制平面和分布式的转发平面,2 个平面相互分离,控制平面利用控制-转发通信接口对转发平面上的网络设备进行集中控制,并提供灵活的可编程功能。与传统以太网相比,基于 SDN 网络的多播由 SDN 控制器维护全网拓扑信息,并根据全网拓扑信息计算路由路径,SDN 交换机只负责转发而无需参与网络信息的维护或计算。

基于以上分析,本文建立一种基于多播的 DSM 框架,将其实施于 SDN 网络中,并使用基于 SDN 的多播为系统提供可控的性能保障。最后通过实验验证该框架的可行性与高效性。

1　基于多播的 DSM 基本架构

DSM 系统由一组节点通过网络互连而形成,其目的是提供一个逻辑上统一的编程空间。由于节点之间网络通信延迟较大,DSM 系统在具体实现时会遇到许多问题,如维持数据与副本之间的一致性、减少访问时的延迟等。

目前,DSM 的实现方式主要有基于页面的共享方式、基于变量的共享方式以及基于对象的共享方式等。其中,基于对象的共享方式因抽象、封装、灵活和自治等特点而备受研究人员的青睐,且其更易于编程。一个对象是一组状态及其上一组方法的封装体,对象的属性表示对象的状态,只有对象方法可以对对象属性进行操作修改并保证信息安全。方法与数据的封装有助于程序的模块化,且对象具有并行性。因此,基于对象的共享方式非常适合 DSM,目前在中、小型系统中应用较广泛[9-11]。

结合基于对象的共享方式和多播通信机制,本文设计一种 DSM 系统并将其运行于 Linux 平台。

1.1　DSM 中的共享对象模型

在 DSM 系统中,用户节点之间共享的是不同类型的对象,共享数据封装在对象中,用户通过对象成员函数访问封装在其中的共享数据。系统提供 4 种基本类型的共享对象:shareInt,shareChar,shareFloat,shareDouble。对象经过运算符重载后,用户可在创建共享对象后直接使用对象进行运算。共享对象在本地节点创建,所有的共享对象构成一个抽象的共享空间,该共享空间由所有节点共享。任意节点都可以访问任何共享对象而无需考虑他们的具体物理位置,对象的定位和管理由 DSM 系统完成。图 1 所示为基于对象的共享方式示意图。

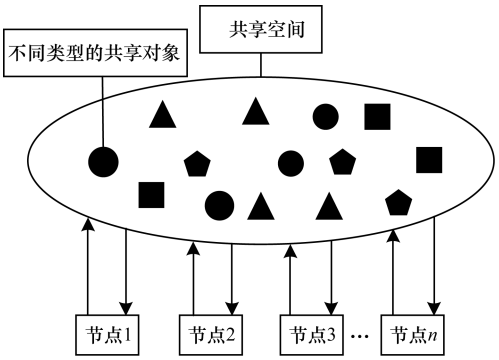

图 1　基于对象的共享方式示意图

在 DSM 系统中每个共享对象都需要一个唯一索引号,这是共享对象在分布式系统中的唯一身份象征。索引号由 2 个部分组成:1)系统事先确定好的表示对象类型的一个对象类型号;2)用户提供的一个整数 ID 号。系统将这 2 个号码组合生成一个唯一索引号。不同类型对象之间的 ID 号可以相同,同一类型对象内的 ID 号不可重复。任意节点都可在声明对象类型并给定 ID 号后将共享空间中相应类型的对象链接到本地,并通过调用对象成员函数来进行对象的访问。这种方式能完成不同类型对象的共享,并有效保护对象中的共享数据。

1.2　基于多播的 DSM 读写策略

DSM 读写策略用于保证共享对象在节点间的静态与动态分布,并降低访问延迟。不同的 DSM 读写算法适用于不同的应用需求,DSM 读写策略主要有单读单写策略、多读单写策略和多读多写策略[12]。多读单写策略能在提高访问效率的同时维持较小的通信开销。本文将多读单写策略与多播机制相结合并应用于 DSM 系统中,具体过程为:系统中对象具有固定的拥有关系,即对象只能被它的创建节点固定拥有,该节点称为对象拥有节点,具有对象全部的读写权利。其他远程节点若想读取对象,首先要通过多播发送读请求命令以查找对象拥有节点,对象拥有节点响应命令,将对象的只读副本发送给请求节点,请求节点读取对象只读副本并保存在本地进程,下次可直接读取只读副本。其他远程节点若想写对象,首先要通过多播发送写请求命令并查找对象拥有节点,由对象拥有节点完成写对象命令并通过多播更新其他节点中的对象的只读副本。对象拥有节点可直接读或写所拥有的对象,写对象后同样要通过多播来更新其他节点中的对象只读副本。

系统中每个节点需要维护 2 个链表:本地链表和副本链表。系统运行时本地节点通过维护的本地链表管理和查找由本地节点创建的共享对象,通过副本链表管理和查找在本地节点保留的远程节点对象的只读副本。

1.3 DSM 中数据传输基本形式

DSM 系统通过 socket 套接字编程来实现多播通信。根据需求,系统需要传输的数据主要有对象类型、对象 ID 号、操作请求、共享对象内容,这些数据需要绑定在一起同时进行传输并解析。因此,系统中数据的传输以一个结构体为基本单位,如图 2 所示。其中,操作请求可以是对共享对象的读请求、写请求或写更新请求等,给定其不同整数值以表示不同的操作请求类型。

```
struct transport{
int shareData;//整型共享数据
int objType;//对象类型
int ID;//对象ID号
int reqType;//操作请求类型
}
```

图 2 传输整型数据所用结构体

1.4 多节点请求与数据保护

DSM 节点不仅需要处理单一请求,也需要处理多个并发请求。因此,有必要采用多线程编程,每次在端口接收到来自多播组的命令后都创建一个线程专门处理该命令。此外,为防止多线程对缓冲区数据的竞争访问,采用互斥锁对缓冲区数据进行保护,线程每次访问缓冲区数据时都需要查询锁的状态,如果锁是释放的,则可以访问缓冲区数据,否则,需要等待直至锁释放。

2 DSM 架构在 SDN 网络中的实施

在 Linux 平台中,依据上文所述,基于多播的 DSM 框架可以实现一个 DSM 系统,该 DSM 系统需要运行于支持多播的网络中。SDN 网络由 SDN 控制器、SDN 交换机和 Linux 终端组成。将 DSM 系统实施于 SDN 网络中时,直接在 Linux 终端运行 DSM 系统,但此时 SDN 网络不具备任何转发功能,更不支持多播,其接收到 DSM 多播请求后无法作出处理。而 SDN 网络中 SDN 控制器负责全网转发控制,SDN 交换机只根据 SDN 控制器的命令转发数据。因此,需要对 SDN 控制器进行编程以使其能够支持 DSM 的多播,进而完成 DSM 中的数据传输。

目前,已经有许多研究人员提出不同的基于 SDN 网络的多播实现方法[13-15],且均已取得较好的效果[16-17]。本文选择使用开源控制器 ryu 作为 SDN 控制器,对 ryu 控制器进行编程以实现多播。SDN 多播需要支持 DSM 系统,并能为 DSM 提供可控的延迟、带宽等性能保障。因此,ryu 控制器需要具有网络状态感知、DSM 多播组以及用户节点的管理、DSM 多播组路由计算等功能。网络状态感知主要负责收集全网链路状态信息,DSM 多播组以及用户节点的管理功能负责处理节点的多播请求并管理多播组,多播组路由计算功能主要负责根据全网状态信息为多播成员计算最佳路径,并下发流表。各功能具体实现方式如下文所述。

2.1 网络状态感知

SDN 控制器需要感知全网状态信息,用于 DSM 用户节点之间多播路由路径的计算。为充分利用网络带宽来降低网络延迟,系统中网络状态的感知分为全网拓扑感知与全网链路带宽感知 2 个部分。

SDN 控制器使用链路层发现协议(Link Layer Discovery Protocol,LLDP)进行全网拓扑感知。SDN 控制器针对每个 SDN 交换机发送包含 LLDP 协议的 Packet-out 报文,SDN 交换机将接收到的报文转发到所有活动接口,与活动接口连接的交换机接收到报文后识别 LLDP 协议,并通过 Packet-in 发送给 SDN 控制器,控制器解析数据包,并创建此 SDN 交换机与邻接 SDN 交换机的连接记录,最终完成全网拓扑记录。SDN 控制器定时发送 LLDP 协议以刷新网络拓扑信息,LLDP 报文只消耗极少带宽,且 SDN 控制器和 SDN 交换机通过安全通道交换信息,因此,该过程对网络产生的负载极小。

除拓扑信息外,还需要获取全网链路带宽信息,本文使用基于 OpenFlow 协议的方法获取链路带宽。ryu 控制器具有封装好的端口流量信息统计应用程序编程接口(Application Programming Interface,API),调用该 API 后,ryu 控制器通过安全通道周期性地下发 port statistics 消息以获得 SDN 交换机端口的统计信息,通过统计端口流量信息可获取 SDN 交换机端口的当前流量带宽,用端口最大带宽减去当前流量带宽则得到可用剩余带宽,将其保存在 SDN 控制器中。

网络状态感知模块将获取到的全网状态信息封装成一个 API 接口,SDN 控制器进行 DSM 用户节点间多播路由路径计算时会调用该 API 接口,以获取全网状态信息。

2.2 DSM 多播组与用户节点管理

同一个 DSM 系统中的用户节点共享一个多播组,用户节点使用 DSM 时会首先申请加入特定多播组,以及时接收到来自该 DSM 多播组的消息。在本文 DSM 系统中,多播使用 IGMP v2 协议。

用户节点加入 DSM 多播组包括 2 个过程:节点本地 socket 服务端的绑定,SDN 控制器中节点 mac 地址的注册。

节点本地 socket 服务端的绑定由 DSM 系统完成。节点的一个本地端口与 DSM 多播地址 multi_addr通过 socket 套接字绑定在一起,节点通过绑定的本地端口监听来自 DSM 多播组的消息。

节点在 SDN 控制器中进行 mac 地址注册的目的在于将节点 mac 地址添加到 SDN 控制器所维护的相应 DSM 多播组中,节点 mac 地址用源地址 src_addr来表示。注册时,节点通过绑定的 socket 向与节点相邻的 SDN 交换机发送包含 IGMP 协议的

Report 报文。初始时 SDN 交换机无法匹配该报文,便将报文以 Packet_in 的形式转发给 SDN 控制器。Packet_in 是 SDN 网络中一种特殊的报文,当 SDN 交换机无法匹配某条消息时,便会将该消息以 Packet_in 报文的形式转发给 SDN 控制器,然后由 SDN 控制器处理该消息。SDN 控制器通过解析 Packet_in 报文可以获得用户节点源地址 src_addr,通过解析 IGMP 协议可以获得请求加入的 DSM 多播组地址 multi_addr,将源地址 src_addr 添加到相应的 DSM 多播组中,从而完成用户节点 mac 地址的注册。注册完成后,源地址 src_addr 就成为相应 DSM 多播组的成员。SDN 控制器通过维护 Multicast_Group_Recods(multi_addr,src_addr1,src_addr2,…)来保存多个 DSM 多播组信息。

用户节点退出 DSM 多播组时,发送一个包含 IGMP 协议的 Leave 报文,SDN 控制器解析该报文获取源地址 src_addr 和多播组地址 multi_addr,然后在维护的相应多播组中删除源地址 src_addr 并更新多播成员。

2.3　DSM 多播组路由计算

SDN 控制器利用由网络状态感知模块提供的全网状态信息为多播用户选择最佳的多播路径。

SDN 控制器通过调用网络状态感知模块的 API 来获取 SDN 交换机的连接记录,并生成全网有向拓扑图。拓扑图中的节点是 SDN 交换机,节点之间的边是 SDN 交换机之间的连接端口,权重是端口可用剩余带宽。需要注意的是,此时调用 API 所建立的拓扑图仅为 SDN 交换机之间的连接图,不包含 DSM 终端节点。在 DSM 终端节点加入或退出多播组后,需要在拓扑图中添加或删除相应的终端节点和边。

最佳路径的选择原则是:首先基于最小跳数确定 k 条路径;然后从中选择具有最大可用剩余带宽的路径作为最佳路径。使用 Dijkstra 算法为多播计算基于跳数的最短路径。Dijkstra 算法的主要特点是以起始点为中心向外层层扩展,直至扩展到终点。每次迭代时选择的下一个顶点是除标记点之外离源点最近的顶点,可以保证源节点到目的节点的路径最短,以此方法确定 k 条最短路径;然后,在确定的 k 条最短路径中,依据节点之间可用剩余带宽求出链路平均可用剩余带宽,选择平均可用剩余带宽最大的路径为最佳路径。

当某一节点向 DSM 多播组内其他成员发送消息时,初始时 SDN 交换机无法处理该消息,便将该消息以 Packet_in 的形式转发给 SDN 控制器,SDN 控制器解析报文以获取源地址 src_addr 和目的多播地址 multi_addr,并通过多播路由计算功能计算源地址 src_addr 到目的多播地址 multi_addr 所有成员节点的最佳多播路径。然后,SDN 控制器下发流表到路径上的 SDN 交换机,SDN 网络中流表用于指示 SDN 交换机如何转发数据,SDN 交换机根据流表内容将来自入端口的数据转发到不同出端口。流表下发完成后,来自源地址 src_addr 的消息直接经过 SDN 交换机转发到其他多播成员节点,而无需再转发给 SDN 控制器。

3　基于 SDN 多播的 DSM 写更新流程

DSM 中所有用户节点首先在 SDN 控制器中进行注册。注册完成后,用户通过调用 API 接口并提供特定类型对象 ID 号将共享空间中共享对象链接到本地进程,然后执行写命令,该过程无需关心共享对象在网络中的具体分布。

系统根据用户提供的特定类型对象 ID 号查询本地链表,判断所要访问的对象是否为本地节点所拥有,如果本地节点不拥有,系统会向多播组多播一个包含对象类型、对象 ID 号和写操作请求的消息。如图 3 所示,该消息会首先发送到与节点相连的 SDN 交换机,若 SDN 交换机没有匹配的流表项,则将消息转发给 SDN 控制器。SDN 控制器接收到消息后解析报文,获取节点源地址 src_addr 和多播组地址 multi_addr,并查询维护的多播组中 DSM 成员节点的 mac 地址,根据节点源地址 src_addr 和多播成员节点目的 mac 地址,通过路由算法为节点生成到其他 DSM 多播成员的最佳路径,然后下发流表到路径上的 SDN 交换机,SDN 交换机开始转发 DSM 多播组消息。下次该节点多播消息时直接按照 SDN 交换机中流表项配置内容进行转发而无需请求 SDN 控制器。DSM 多播组内其他节点成员每次接收到多播命令后,都创建一个新线程,并查询本地进程是否拥有此对象,如果拥有,则执行写操作命令,并将更新后的对象副本通过 SDN 多播发送出去,以更新其他 DSM 成员节点中的对象只读副本。如果所写对象是本地节点所拥有,则直接写对象并通过 SDN 多播更新 DSM 中的对象只读副本。

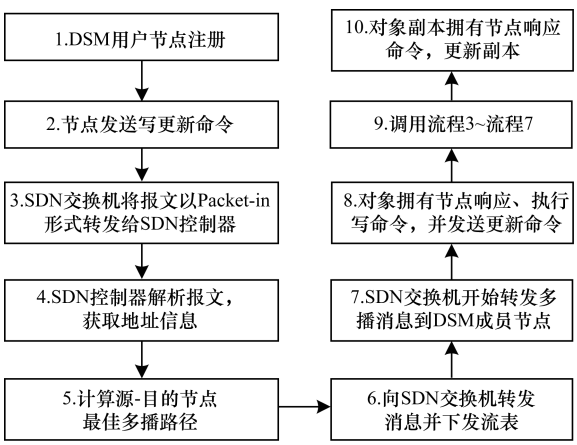

图 3　DSM 写更新流程

4 系统性能测试与分析

为验证本文系统的实际性能,使用网络仿真工具 mininet(内置协议为 OpenFlow v1.3)搭建一个 SDN 网络,并将其连接到一个 ryu 控制器。网络选择经典的“胖树”拓扑,它是数据中心的常用拓扑,具有足够的路径多样性,可以验证系统性能。如图 4 所示,本次实验拓扑包括 12 个终端节点和14 个SDN 交换机节点。

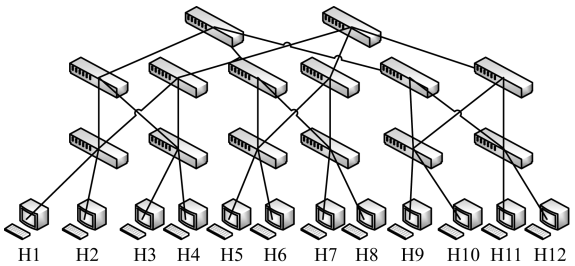

图 4 实验网络拓扑

拓扑中每条链路带宽设为 15 Mb/s。为了与基于传统多播的 DSM 系统进行比较,使用支持 IGMP 协议的路由器搭建相同拓扑的以太网络,传统多播使用最短路径树(Shortest Path Tree,SPT)算法。测试分 2 个部分:1)测试 DSM 中 SDN 多播的延迟、丢包率性能;2)测试同时写更新多个对象时 DSM 的平均执行耗时。

4.1 平均网络延迟与丢包率

本次实验测试多播流量递增时 SDN 多播的平均网络延迟和丢包率,测试工具为 iperf[18]。使用客户端-服务端模式,选取一个终端作为 iperf 客户端,不断发送特定流量的 UDP 数据包,并选取 4 个终端作为 iperf 服务端来接收数据流量包。每次测试中带宽流量从 1 Mb/s ~ 10 Mb/s 逐级增加,共测试 10 次,每次测试时间为 60 s。iperf 工具会得出当前的网络延时、丢包率数据,每次测试网络共运行4 个多播组,在测试完成后记录多播组每个终端所得延迟、丢包率,并取平均值作为最终结果。每个多播组的终端分配情况如表 1 所示。

表 1 多播组终端分配情况

多播组序号	iperf 客户端	iperf 服务端
1	H1	H5,H6,H8,H9
2	H3	H6,H7,H10,H11
3	H5	H7,H9,H11,H12
4	H7	H8,H9,H10,H12

网络延迟性能测试结果如图 5 所示。由图 5 可以看出,相比以太网,SDN 网络中 DSM 多播具有更小的传输延迟,且在多播流量递增时,SDN 网络中 DSM 多播延迟具有更好的稳定性。丢包率性能测试结果如图 6 所示。由图 6 可以看出,以太网多播在流量为 2 MB 时开始丢包,而 SDN 多播直到 4 MB 时才开始丢包,其丢包率一直低于前者,且流量越大,SDN 多播的性能优势越明显。因此,相对传统 SPT 算法,DSM 中的 SDN 多播具有更好的性能,在流量较大、网络较拥挤时,其能将延迟、丢包率等维持在较低水平,从而为 DSM 提供性能保障。

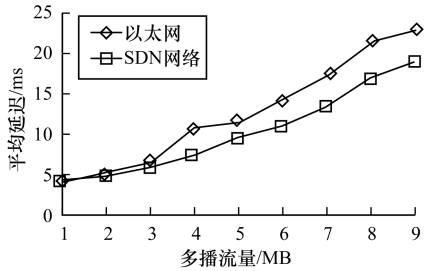

图 5 2 种网络平均延迟对比情况

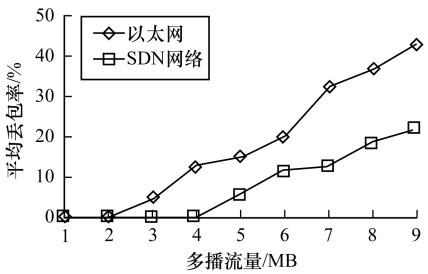

图 6 2 种网络平均丢包率对比情况

4.2 DSM 写更新平均耗时

DSM 中共享对象和对象副本在节点中的不同分布会产生不同的通信次数。为测试 DSM 系统性能,需要尽可能地最大化写更新命令中的通信次数,因此,本文选择测量写更新远程节点对象并完成副本更新的平均耗时,即从写命令发起到待更新节点接收到副本更新命令所需要的时间,以此评估 DSM 系统性能。

实验中每次运行不同数量的 DSM 系统,最多同时运行 9 个 DSM 系统,每个 DSM 系统同时更新 5 个对象。选取其中 5 个 DSM 系统展示写更新时节点分配情况,结果如表 2 所示。每个系统包含 5 个节点,为了最大化通信次数,写命令发起节点、写命令执行节点和待更新节点均是不同节点。本文使用 Linux 命令行工具 tcpdump 监听 DSM 系统每个节点的端口,并抓取端口发送与接收的数据包。DSM 中每个节点都分配一个 IP 地址以方便解析数据包。

表 2 DSM 节点分配情况

DSM 系统序号	写命令发起节点	写命令执行节点	待更新节点
1	H1	H5	H4,H6,H7
2	H3	H6	H4,H7,H8
3	H5	H7	H8,H9,H10
4	H7	H8	H9,H10,H11
5	H9	H4	H10,H11,H12

写命令发起节点发起写命令时,会发送一个多播消息,该多播消息表示写命令起始时间,写命令执行完成后,待更新节点会接收到包含更新命令的多播消息,本文将该消息视为写命令完成时间。通过解析多播数据包的目的地址以识别不同消息并获取时间戳。写命令起始、完成的时间戳差值就是执行一次写更新命令的耗时。DSM 系统执行一次写更新操作的时间较短,实验中不容易捕捉该时间差。因此,让每个 DSM 系统连续写更新 50 次,得到 50 次操作的总时间,然后除以 50 以得到每次写更新的操作时间。

写更新平均耗时测试结果如图 7 所示。由图 7 可以看出,初始时写更新的 DSM 数量较少,2 种网络耗时相差不大,曲线较吻合。当同时运行的 DSM 数量增多时,以太网中的平均耗时产生较大幅度的增加。尤其是 DSM 数量大于 5 后,以太网写更新耗时曲线更陡峭,增幅更大,最终耗时达到 60 ms,系统性能较差。而 SDN 网络中 DSM 的平均耗时比以太网少,且在 DSM 数量增多过程中,其曲线增长始终较平缓,耗时增加幅度较小,系统稳定性较强。在 DSM 数量为 9 时,SDN 网络平均耗时约 30 ms。因此,SDN 网络相比传统以太网具有更好的性能。

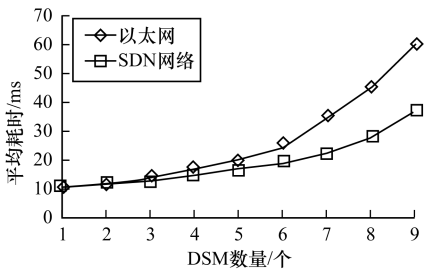

图 7　2 种网络写更新平均耗时对比情况

由上述实验结果可以看出,与传统以太网相比,基于 SDN 实现的多播具有更好的传输性能,将 SDN 多播应用于 DSM 系统中,能有效降低 DSM 命令执行平均耗时,提高系统稳定性。产生该结果的原因主要在于:

1)当网络中存在多条可用路径时,SDN 控制器能根据全网链路状态信息为 DSM 选择有较大剩余带宽和较小跳数的最佳路径,从而有效避免拥塞链路并充分利用全网资源。

2)SDN 网络控制与转发分离。由 SDN 控制器维持全网信息并为 DSM 节点选择最佳路径,SDN 交换机只负责转发,这能够极大提高转发效率。

因此,将基于 SDN 实现的多播应用于 DSM 系统,可以在网络流量较大时有效降低系统平均延迟和命令平均执行时间,从而为 DSM 提供良好的性能保障。

5　结束语

SDN 网络具有集中控制与分布式转发相分离、全网状态感知等优势,将基于 SDN 网络实现的多播应用于 DSM 系统,可以大幅提升传输效率。基于此,本文设计一个 DSM 系统,并将其与基于 SDN 的多播进行结合。实验结果表明,与传统以太网相比,基于 SDN 多播的 DSM 可以有效降低平均网络延迟,减少命令响应时间,提高系统性能。下一步考虑使用基于 SDN 的多播来优化大数据应用。

参考文献

[1] LI K. Shared virtual memory on loosely coupled multiprocessors[D]. New Haven,USA:Yale University,1986.

[2] LI K,HUDAK P. Memory coherence in shared virtual memory systems[C]//Proceedings of ACM Symposium on Principles of Distributed Computing. New York,USA:ACM Press,1986:229-239.

[3] AMZA C,COX A L,DWARKADAS S,et al. TreadMarks:shared memory computing on networks of workstations[J]. Computer,1996,29(2):18-28.

[4] BAL H E,TANENBAUM A S. Distributed programming with shared data[C]//Proceedings of International Conference on Computer Languages. Washington D.C.,USA:IEEE Press,1988:82-91.

[5] CHARLES P,DONAWA C,EBCIOGLU K,et al. X10:an object-oriented approach to non-uniform cluster computing[J]. ACM SIGPLAN Notices:A Monthly Publication of the Special Interest Group on Programming Languages,2005(10):519-538.

[6] ADVE V S,DIG D,ADVE S V,et al. A type and effect system for deterministic parallel Java[C]//Proceedings of ACM SIGPLAN Conference on Object Oriented Programming Systems Languages and Applications. New York,USA:ACM Press,2009:97-116.

[7] BERGER E D,YANG T,LIU T,et al. Grace:safe multithreaded programming for C/C++[J]. ACM SIGPLAN Notices,2009,44(10):81-96.

[8] 吴成富,欧　峰,陈怀民,等. 基于组播技术的共享内存网络设计与实现[J]. 测控技术,2009,28(10):55-58.

[9] 刘谋用,葛霁光. 实时分布式操作系统中共享对象通信模型的实现[J]. 计算机研究与发展,1999,36(3):309-314.

[10] 李　群,谢　立,孙钟秀. 分布式共享内存的技术和实现[J]. 计算机研究与发展,1997,34(5):327-331.

[11] 陈　曦,朱建涛,何晓斌. 一种面向高性能计算的分布式对象存储系统[J]. 计算机工程,2017,43(8):69-73.

[12] PROTIC J,TOMASEVIC M. Distributed shared memory:concepts and systems[J]. IEEE Parallel and Distributed Technology Systems and Applications,1996,4(2):63-71.

(下转第 60 页)

在应用领域,矢量有限元法是求解电磁问题的重要方法之一,该方法将一个离散后的线性系统归结为方程 $Ax = b$ 的求解,而电磁问题的复杂化使得系数矩阵 A 愈加不规则,因而可以考虑通过多次划分 ELL 格式进行存储,以用于更加不规则的线性方程组计算。同时,建立一个稀疏矩阵存储格式的优选模型,根据不同矩阵的特点选取最优的存储格式以提升计算效率,也将是下一步的研究方向。

参考文献

[1] NAUMOV M. Incomplete LU and Cholesky preconditioned iterative methods using CUSPARSE and CUBLAS[EB/OL]. [2017-07-01]. https://docs.nvidia.com/cuda/incomplete-lu-cholesky/.

[2] Nvidia. CUDA CUBLAS Library[EB/OL]. [2017-07-01]. http://devel oper.download.nvidia.com/compute/DevZone/docs/html/CUDALibraries/doc/CUBLAS_Library.pdf.

[3] Nvidia. CUDA CUSPARSE Library[EB/OL]. [2017-07-01]. http://developer.download.nvidia.com/compute/DevZone/doc/html/CUDALibraries/doc/CUSPARSE_Library.pdf.

[4] 张健飞,沈德飞. 基于 GPU 的稀疏线性系统的预条件共轭梯度法[J]. 计算机应用,2013,33(3):825-829.

[5] 陈 尧,赵永华,赵 慰,等. GPU 加速不完全 Cholesky 分解预条件共轭梯度法[J]. 计算机研究与发展,2015,52(4):843-850.

[6] 殷 建. 基于 GPU 的矩阵乘法优化研究[D]. 济南:山东大学,2015.

[7] 阳王东,李肯立. 基于 HYB 格式稀疏矩阵与向量乘在 CPU + GPU 异构系统中的实现与优化[J]. 计算机工程与科学,2016,38(2):202-209.

[8] Computing Developer Home Page[EB/OL]. [2017-07-01]. http://developer.nvidia.com/object/gpucomputing.html.

[9] Nvidia. NVIDIA CUDA C programming guide[EB/OL]. [2017-07-01]. http://developer.download.Nvidia.com/compute/DevZone/docs/html/C/doc/CUDA_C_Programming_Guide.pdf.

[10] BELL N, GARLAND M. Efficient sparse matrix-vector multiplication on CUDA:NVR-2008-004[R]. Santa Clara, USA:Nvidia Corporation,2008.

[11] 秦 晋,龚春叶,胡庆丰,等. 基于 CUDA 编程模型的稀疏对角矩阵向量乘优化[J]. 计算机工程与科学,2012,32(7):78-83.

[12] 白洪涛,欧阳丹彤,李熙铭. 基于 GPU 的稀疏矩阵向量乘优化[J]. 计算机科学,2010,37(8):168-172.

[13] 袁 娥,张云泉,刘芳芳,等. SpMV 的自动性能优化实现技术及其应用研究[J]. 计算机研究与发展,2009,46(7):1117-1126.

[14] BELGIN M. BACK G, RIBBENS C J. Pattern based sparse matrix representation for memory-efficient SMVM kernels[C]//Proceedings of the 23rd International Conference on Supercomputing. New York, USA:ACM Press,2009:100-109.

[15] CHOL J W, SINGH A, VUDUC R W. Model-driven autotuning of sparse matrix-vector multiply on GPUs[J]. ACM SIGPLAN Notices,2010,45(5):115-125.

[16] 梁 添. 基于 GPU 的稀疏矩阵运算优化研究[D]. 武汉:华中科技大学,2012.

[17] 阳王东,李肯立,石 林. 一种准对角矩阵的混合压缩算法及其与向量相乘在 GPU 上的实现[J]. 计算机科学,2014,41(7):290-296.

[18] WILLIAMS S, OLIKER L, VUDUC R W, et al. Optimization of sparse matrix-vector multiplication on emerging multicore platforms:RC24704(W0812-047)[R]. IBM Inc.,2008.

[19] 夏健明,魏德敏. 共轭梯度法的 GPU 实现[J]. 计算机工程,2009,35(17):274-276.

[20] JI Hao, LI Yaohang. Block conjugate gradient algorithms for least squares problems[J]. Journal of Computational and Applied Mathematics,2017,317:203-217.

[21] 张 兰. 稀疏矩阵方程组预处理迭代技术研究[D]. 广州:华南理工大学,2010.

[22] university of Florida sparse matrix collection[EB/OL]. [2017-07-01]. http://www.cise.ufl.edu/research/sparse/matrices/list_by_id.html.

编辑 金胡考

(上接第 53 页)

[13] HUANG L, ZHI X, GAO Q, et al. Design and implementation of multicast routing system over SDN and sFlow[C]//Proceedings of IEEE International Conference on Communication Software and Networks. Washington D.C.,USA:IEEE Press,2016:524-529.

[14] 谢永斌,张 宸. 基于 SDN 的 IP 组播研究[J]. 信息通信,2015(2):202-203.

[15] ZOU J,SHOU G,GUO Z,et al. Design and implementation of secure multicast based on SDN[C]//Proceedings of IEEE International Conference on Broadband Network and Multimedia Technology. Washington D.C., USA: IEEE Press, 2014:124-128.

[16] GAO Q, TONG W, KAUSAR S, et al. Design and implementation of SDN multicast for distributed shared memory[C]//Proceedings of International Conference on Future Generation Communication and Networking. Washington D.C.,USA:IEEE Computer Society,2015:5-8.

[17] 余燕平. 多播路由算法的研究[D]. 杭州:浙江大学,2002.

[18] iPerf-the ultimate speed test tool for TCP, UDP and SCTP[EB/OL]. [2017-07-05]. https://iperf.fr/.

编辑 吴云芳