分布式环境下端到端的多路并行传输机制研究

刘立明
(国家气象信息中心 高性能计算室,北京 100081)

摘 要:在分布式环境中,不同的管理域之间的节点由于受到访问控制机制的限制,导致端到端的数据传输无法直接进行,因此必须借助数据源节点和目的节点之间的节点进行中转。此外,在复杂的分布式环境中,数据传输的源端和目的端之间存在多条传输路径,需要考虑在端到端的数据传输过程中如何对多条传输路径进行均衡利用,以提高数据传输效率。为此,利用端到端的数据传输的特点,基于缓冲区、数据分片、多路并行传输等技术,设计了一种适应高性能计算环境中的多路并行数据传输机制,研究了可靠性保证策略。理论分析表明,多路并行传输机制可以有效解决分布式环境下的数据传输访问控制和传输效率等问题,尤其是对于较为复杂的分布式环境下进行的大数据传输,该机制充分利用了整个网络环境的传输带宽,同时解决了中间节点的数据落地问题,为端到端的数据传输提供了高效可靠的保障。

关键词:分布式环境;数据传输;传输效率;负载均衡;端到端
中图分类号:TP309 **文献标识码**:A **文章编号**:1673-629X(2017)06-0001-06
doi:10.3969/j.issn.1673-629X.2017.06.001

Research on End-to-end Multipath Parallel Transfer Mechanism in Distributed Environments

LIU Li-ming
(HPC Department, National Meteorological Information Centre, Beijing 100081, China)

Abstract: In distributed environment, nodes in different management domains can't transfer data to each other directly because of restrictions of access control mechanism, so that the end-to-end data transfer has to rely on the mid nodes between the source and destination to transit. In addition, there are more than one transfer path between the source and destination in complex distributed environments, and a way to utilize those transfer paths balanced to improve the end-to-end data transfer efficiency needs to be considered. According to the characteristics of end-to-end data transfer, an efficient end-to-end multipath parallel transfer mechanism has been designed with the technology of buffer area, data segmentation and multipath parallel transfer and effective strategies to ensure the transfer reliability. The results of analyses show that this mechanism has effectively solved the problems of access control and transfer efficiency in distributed environment which is a highly efficient and reliable transfer security mechanism for end-to-end data transfer, and that especially for big data transmission in more complex distributed environment, the bandwidth of the network environment could be made full use of and the problem of the data to disk on intermediate node has been solved.

Key words: distributed environment; data transfer; transfer efficiency; load balancing; end-to-end

1 研究背景

分布式环境是一个在地理位置上广泛分布的、开放的分布式系统,是把地理位置上分散的资源集成起来的一种基础设施[1]。分布式计算环境为分散在整个 Internet 的用户提供各种计算资源,这些资源由不同管理域中的节点提供,用户可按需定制使用。

以计算网格的典型应用场景高性能计算(High Performance Computing, HPC)为例[2],对分布式环境下一种典型的数据传输场景进行描述。一个分布式作业的执行过程分为 Stage-In、Execute 和 Stage-Out 三个阶段。前两个阶段都与数据传输密切相关。如图 1 所示,位于管理域 A 中的用户提交一个作业到位于管理域 C 中的计算节点,在 Stage-In 阶段,需要将作业运行所需要的位于管理域 B 中 FTP 服务器上的输入文件传输到位于管理域 C 中的计算节点上。当作业执行结束后进入 Stage-Out 阶段,计算节点需要将作

收稿日期:2016-06-13 **修回日期**:2016-09-22 **网络出版时间**:2017-03-13
基金项目:国家发改委中国气象局"十二五"重点工程建设项目(ZQC-H14175)
作者简介:刘立明(1981-),男,硕士,工程师,研究方向为分布式计算。

业执行所产生的输出结果(数据文件)反馈给提交作业的网格用户。

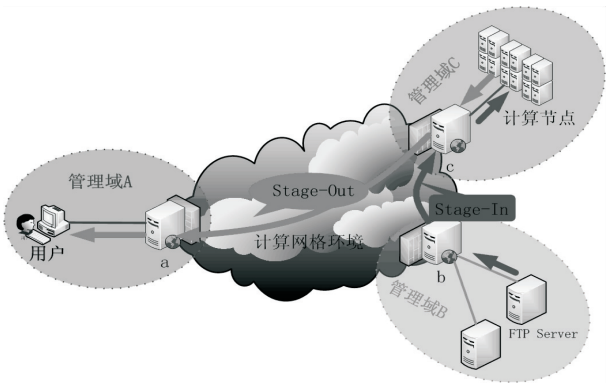

图 1　计算网格环境下“高性能计算”应用的数据传输场景

在这样一个简单的分布式环境中,不难发现在数据传输过程中存在的问题:用户登录节点、计算节点、作业执行需要的数据资源所在节点分别位于不同的管理域,而这些管理域之间存在访问权限的问题,一般采用防火墙进行控制。每个管理域提供一个“头节点”(head node)供其他管理域中的节点访问,所有内部“私有节点”(private node)都不对其他管理域开放,这使得在 Stage-In 阶段无法直接建立计算节点与作业执行需要的数据资源所在节点之间的连接进行数据传输;同样,在 Stage-Out 阶段也无法直接建立计算节点与用户登录节点之间的传输连接。

在上述场景中,数据传输双方无法以传统的 C/S 模式直接建立连接,数据传输过程只能利用网格环境中的中间授权节点进行中转。以 Stage-In 阶段为例,现在使用较多的处理方式是:首先将数据资源从其所在节点传输到该节点所属的管理域 B 的头节点 b,然后网格用户利用网格身份登录管理域 B 的头节点 b,将数据传输到节点 b 可以直接访问的管理域 C 的头节点 c,最后网格用户再登录节点 c,将数据资源传输到执行作业的计算节点上。利用这种重复“登录→传输”的中转方式将数据从源节点传输到目的节点。这种传输方式的缺点很明显:对于网格用户来说,需要利用网格账号多次登录不同网格节点才能完成一次数据传输操作;对于所用到的各个中转节点来说,会在节点上留有数据副本,需要用户在作业执行结束后手动删除。这种传输模式无论是对用户使用的便捷性还是对数据传输效率的保障都是个巨大的挑战。

在分布式计算环境中,涉及传输过程的数据源端和目的端之间的传输操作无法直接进行,只能间接通过中间节点,找到从“源”到“目的”的传输路径,并借助该路径按照特定的策略通过一系列中转操作将数据发送到目的节点,这种传输模式称为端到端的数据传输。这种数据传输模式不同于传统客户端/服务器(C/S)模式,因为传输双方无法直接建立 C/S 连接;也不同于 KaZaA[3]、Napster[4]等 P2P 模式,因为在 P2P 模式中资源请求节点先通过泛洪(Flooding)查询来发现资源,找到资源后直接建立点对点的传输连接。

端到端的数据传输具有以下特点:首先,进行传输的数据源节点和目的节点之间无法直接互联。其次,有了数据中转节点的加入,在端到端的传输过程中存在多条可行的传输路径。由于这些路径上的链路带宽不同、距离不同等因素存在,导致链路传输能力具有较大差别。不同传输路径的选择对端到端的数据传输效率有很大影响。

在分析影响端到端的数据传输效率的关键问题的基础上,将其归结为数学问题进行研究,设计并提出了一种适合端到端数据传输的多路并行传输机制,同时提供了可靠性保证策略,并对其进行了理论分析。

2　端到端的数据传输问题的数学描述

用带权无向图 $G=(V,E)$ 表示分布式计算环境,图的节点 $V=(v_0,v_1,\cdots,v_n)$ 表示计算节点和传输节点,图的边 $E=(E_0,E_1,\cdots,E_m)$ 表示节点间的网络连接。权值 w 表示每条链路上的时间损耗(反映空闲网络带宽),加权函数 $w:E\rightarrow R$ 表示从链路到实型权值的映射。所以,每条可行的传输路径 $P=(v_0,v_1,\cdots,v_k)$ 的权为该路径上所有可行链路的权值之和:

$$w(p)=\sum_{i=1}^{k}w(v_{i-1},v_i) \tag{1}$$

如果图中 $u\rightarrow v$ 找到 n 条传输链路 $P_1,P_2,\cdots,P_t$,那么求解最短路径的过程就是寻找单个传输任务的最短传输时间的过程。

最短传输链路的权可表示为:

$$\delta(u,v)=\begin{cases}\min\{w(p):u\xrightarrow{p}v\}, & \text{存在从 } u \text{ 到 } v \text{ 的路径}\\ \infty, & \text{不存在从 } u \text{ 到 } v \text{ 的路径}\end{cases} \tag{2}$$

那么,端到端的数据传输可转化为在带权无向图 G 中的多任务并发传输问题,任务数量用 $n(n>1)$ 表示。从 u 到 v 间的传输链路 $P_1,P_2,\cdots,P_t$ 上的网络负载是动态变化的,每个任务在传输过程中根据链路繁忙程度选择最快传输链路即可获得最大传输效率。所以,端到端的传输问题可转化为最大化地均衡利用 G 中可行传输链路以获得最高的吞吐率,使得总传输时间最短,可表示为:

$$q(n)=\sum_{j=1}^{n}\delta_j(u,v) \tag{3}$$

为了和传统路由传输加以区分,在表 1 中将端到端的数据传输和传统的路由传输进行了对比。

表 1　传统路由传输与端到端数据传输的对比

指标	传统的路由传输	端到端的数据传输
所在层次	网络层	应用层
底层协议	IP 协议	应用层协议(如:FTP,HTTP)或传输层协议(如:TCP,UDP)
中转对象	分组	File 或 Data Block
传输对象大小	每个分组都很小,一般都≤10 KB	每个 File 或 Data Block 都较大,一般都≥1 MB
传输对象频率	整个网络中信息量交换频繁,数据量相对来说较小	整个网络中数据量交换频繁,传输数据量大
中转节点	从源节点到目的节点之间的路由器作为中转节点	分布式环境中的物理节点作为中转节点

端到端的数据传输应考虑两个重要问题:采用何种传输机制来提高传输效率;对于存在的多条传输路径,如何充分利用以实现合理的网络分流,提高网络资源的利用率。

3　相关工作

3.1　网格文件传输服务 GridFTP

GridFTP 是美国 Argonne 国家实验室在 GT(Globus Tools)项目中设计的数据传输协议,它对标准 FTP 协议(RFC959)进行了扩展,有 12 所大学和科研机构参与该项目研发。GridFTP 为 Globus Toolkit 提供了高效的传输工具[5]。

(1)自动调整 TCP 缓冲窗口的大小(Auto Negotiation)。用户可以选择手动设置或者让 GridFTP 自动设置 TCP 缓冲窗口的大小,提高数据传输的性能。

(2)支持由第三方控制的数据传输(Third-party Data Transfer)。GridFTP 将操作控制通道的逻辑单元称为协议解释(PI),对数据通道的操作过程称为数据传输过程(DTP),实现了传输和控制的分离。Client 端通过控制通道发送命令,让 Server 端与其他节点建立传输连接,实现数据传输。

(3)支持并行数据传输(Parallel Data Transfer)。GridFTP 通过多个并行的 TCP 可有效利用链路带宽[6],它对指令和数据通道进行了扩展,支持并行数据传输。

(4)支持部分文件传输(Partial File Transfer)。GridFTP 支持数据文件中特定部分的数据传输。它引入新的控制指令支持对一个数据文件的任意部分进行传输操作。

Condor Stork[7]、Globus RFT 以及 gLite FTS 等文件传输服务都使用 GridFTP。但 GridFTP 和防火墙等管理工具的结合没有很好地被解决,三方传输功能受到限制[8]。

3.2　CNGrid 数据传输子系统

CNGrid 数据传输子系统[9]是由国防科技大学和清华大学联合开发的,应用在 CNGrid 系统端到端的数据传输中,可满足两种传输情景。它设计了简单数据中转的传输方法,称为一跳传输和二跳传输,如图 2 所示。其中,图中的黑色节点是具有公网 IP 的服务器(称为头节点),其余节点为私有节点(只能被头节点访问)。

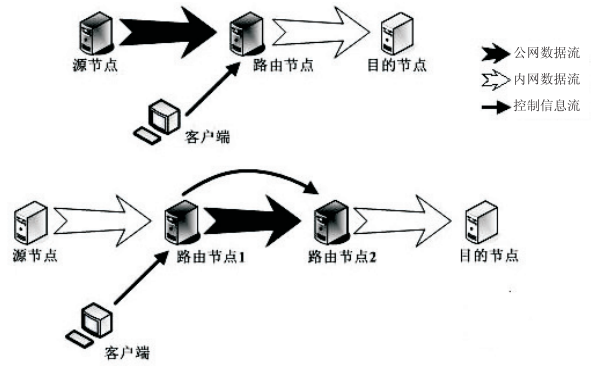

图 2　CNGird 数据传输子系统的研究场景

CNGrid 数据传输子系统提供了一种分布式环境下简单的端到端的数据传输方法,但该方法存在一定的局限性:首先,这种数据传输路径由用户指定,用户需要清楚实际部署环境中的部署情况和传输情景;其次,该系统只简单地实现了一跳传输和两跳传输情景,对于需要多于两跳才能完成的传输情景无能为力;第三,传输过程中会在路由节点上生成临时文件,占用大量磁盘空间,最后还需要进行删除处理的善后操作,增加了操作的复杂性和写磁盘带来的性能开销。

4　影响端到端传输效率的关键问题

4.1　存储转发问题

在端到端的数据传输系统中,CNGrid 数据传输子系统利用磁盘暂存传输数据再进行转发,这种传输方式需要在传输任务完成后再将暂存的临时数据文件删除。这种方式产生的时间开销极大地降低了传输效率。第一,读写磁盘的速度要远远低于处理器的处理速度,两者不在一个数量级上,所以利用磁盘文件落地后中转会严重浪费 CPU 的处理周期。第二,中转文件要在中间节点存储完成后才能进行转发,会浪费大量的等待时间,从而极大地降低了传输效率。

4.2　点对点传输问题

单流式顺序传输对于数据传输来说有很多不利的方面。第一,标准传输协议都是基于 TCP/IP 协议簇设计的,它们的窗口机制并不匹配高速网络设备的处理速度,致使现有传输协议不能充分利用链路的网络带宽[10]。第二,虽然标准传输协议的顺序传输保证了传输过程的数据容错处理,但是却在很大程度上降低了传输效率,因为在传输时需要对数据流上的传输字节

的顺序进行监控,无法进行并行传输。

4.3　多路传输问题

在端到端的传输过程中利用磁盘做存储转发只是利用单流式顺序传输使文件落地,无法利用分布式环境中的多条空闲链路,使得整个网络环境的利用率极低,无法获得较高的传输时效。同时,这些可行的传输链路在网络带宽、拥塞程度、节点性能等方面可能会有较大差别,利用它们进行中转传输时不能统一对待。如果可以充分考虑这些传输链路各方面的差别,研究一种合理的多路利用策略,以充分利用多条可行传输路径对传输任务进行传输,则可以提高整个网络的利用率以及数据传输的效率。

5　多路并行传输机制的设计

5.1　数据缓冲区策略

众所周知,读写寄存器的速度与 CPU 的处理速度最接近,但是寄存器的存储空间小、价格高,只适合缓存临时数据,不能用作存储大数据。除此之外,用内存作为数据存储可以获得磁盘无法比拟的读写速度,更接近处理器的处理速度,是一种高效的存储方法。但是存储数据不能在内存上进行持久化。

设计一种数据缓存机制,让传输过程可以有效利用内存区的存取速度,称为缓冲区机制。即在中转节点上构建一块内存区域专门用作中转数据的缓冲区,实现对转发数据的缓存,降低存储转发过程所产生的时间开销,从而提高中转效率。缓冲区只能存储部分数据,无法存储大数据文件。后面会引入数据分片机制的设计,实现缓存与动态转发机能,快速处理缓冲区的数据块,充分利用内存区域,更大的优势是解决了中转节点需存完整个文件后才能转发的问题,同时解决了内存空间大小的限制,如图 3 所示。

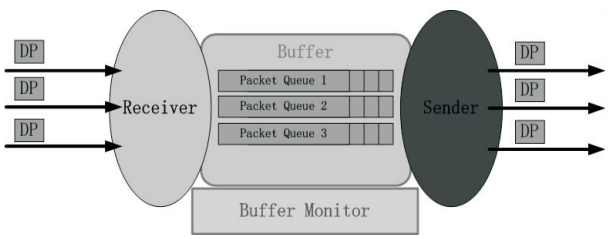

图 3　利用缓冲区机制存储转发示意图

5.2　多路并行传输的实现机制

5.2.1　多路分流策略

前面在分析影响端到端的传输效率的关键问题时,提到当前大多数端到端数据传输系统都是利用在分布式环境中按照一定的策略选择一条比较优的路径(或者指定路径)的方式完成端到端的数据传输,没有充分利用网络中源节点和目的节点之间存在的多条可行传输链路,如图 4 所示。

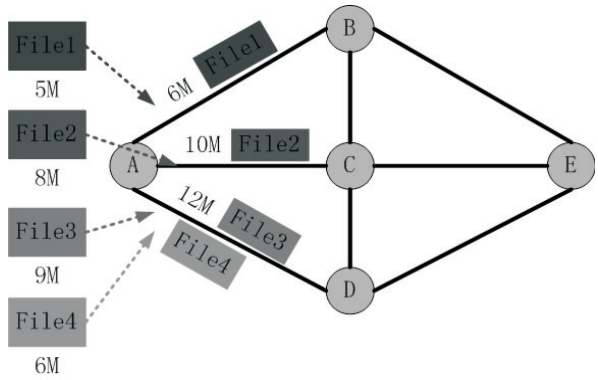

图 4　多路并行传输

从数据源节点到达目的节点存在多条可行路径,每个中转节点接收的数据块都有多条可达目的节点的传输链路可以选择。在每个节点上都充分利用所有可行链路去完成数据块的传输,那么带宽的总体吞吐量将获得很大程度的提高。单条链路 k 上的带宽设为 BW_k,那么在理论上可用于传输的网络总带宽变成:

$$BW_{sum} = \sum_{k=1}^{n} BW_k \tag{4}$$

此外,多路径传输机制带来的另一个好处是:如果在传输过程中遇到局部节点或链路崩溃的情况,多路径传输策略能够使用备用路径完成传输,因此在一定程度上保证了传输的可靠性。

5.2.2　数据分片策略

在分布式计算、云计算等并行计算领域,一般会将单个计算任务分解,每个任务在一定的颗粒度上分解为多个独立的子任务,以便并行处理来充分利用计算资源,提升计算效率。端到端的传输可以借鉴这种原理。为了提高数据传输效率,可以对一个传输任务进行切分,将其切分成若干子传输任务,结合下面介绍的多流并发传输机制,充分利用网络带宽,提高传输任务的执行效率。

为此,将需要传输的数据文件按设定尺寸切分成若干大小相等的数据块(Block,数据文件的最后一个数据块小于等于数据块设置大小),每个数据块具有独立的数据结构,构建成数据包(Packet),作为单独的传输子对象,从而实现了对单个传输任务的并行切分。中转节点的缓冲区会接收并解析每个接收到的数据包的数据结构,包含传输的源地址、目的地址、源文件信息、目的文件信息、数据块号、起始位置、数据块大小、校验和、时间戳、实际数据等信息。然后根据对链路传输能力的评估选择队列进行转发,最终所有数据包到达目的节点后进行任务合并,通过解析数据包头部信息,将数据块写入到目标文件并进行校验,最终完成传输任务。对于数据分块的大小,按照文献[11],设置数据分块大小为 1 MB。

图 5 演示了上述过程,可以清晰地看到与图 4 的

区别。四个传输任务(File1 ~ File4)到达节点 A 后,File1 ~ File3 分别选择链路 ABE(5 MB)、ACE(8 MB)和 ADE(9 MB)进行转发,每条可用链路都有剩余带宽,File4 整体传输只能选择最优的链路 ADE(剩余 3 MB),但是利用数据分片机制可将其切分成 6 个单独的数据块(1 MB)。然后并行分布到三条具有剩余带宽的链路上实现转发,最终在目的节点 E 对所有数据块进行合并。这种机制在很大程度上提高了带宽的利用率。

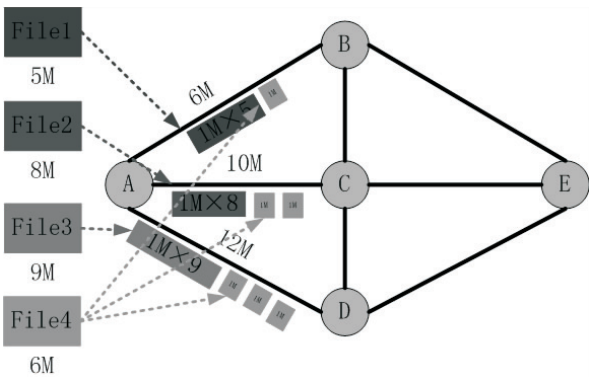

图 5　数据分片传输

此外,数据分片传输带来的另一个好处是:在一定程度上增强了端到端传输过程中数据的安全性。因为在多路径并行传输的情况下,数据包经不同的链路传输到目的端,数据包在任何中间节点上被截获也无法被构造出源文件,所以说数据分片传输增强了数据传输的安全性。

5.2.3　多流并发策略

提高带宽利用率的一类方法是通过调节 TCP 缓冲窗口的大小来提高链路的带宽利用率[12-14]。这种方式的原理是当 TCP 最优缓冲窗口大小(Optimal Buffer Size)等于链路带宽(Bandwidth)与往返时延(RTT)的乘积时,可以最大程度地利用传输链路的带宽。GridFTP 利用这种方式来提高网络带宽利用率。这种调整方法对于应用层的传输来说复杂性较高。一方面,由于网络实际的可用带宽具有动态变化性,所以调整 TCP 缓冲区的做法复杂性高,需要不断动态调整才能满足要求;另一方面,在应用层的传输需要操作传输层 TCP 缓冲窗口大小,从设计的角度来说不符合设计规范。

另一类提高带宽利用率的方法是多数据流并发传输机制[10]。这种方法的本质是在相邻两个节点间构建多个并行的 Socket 对,根据特定的调度机制进行数据的并行发送和接收。具体方法是每个中转节点在应用层建立 N($N>1$)个 TCP 流,根据调度机制进行监听、接收、发送操作。该方法的优点是不用手动调整网络层的 TCP 缓冲区大小,规避了传统传输协议的劣势,同时提高了链路的带宽利用率。文献[10]分析了该机制的优势,并实验对比了与调整缓冲区方案的传输效率区别,证明了多流并发机制在提高链路吞吐率方面的优势,可以获得更高的传输效率。

在实验评测中,通过多组实验数据证明了在百兆网环境中,使用系统默认的 TCP 缓冲区大小时的并发流数目为 5 ~ 12 时,可以获得较优的传输效率。

6　传输过程的可靠性保证

Condor 中设计的 Stork[7] 采用集中式的管理策略保证数据传输的可靠性,缺点是过于依赖监控节点,一旦监控节点异常,可靠性保证机制将失效。受传输层 TCP 协议可靠性保证策略[15]启发,设计一种请求重传与超时重传相结合的可靠性保证策略,并针对网络传输性能动态变化的特点,提供了一种动态估计最大超时时间的方法。

6.1　出错请求重传

在传输源节点为每个数据块构建数据包时,采用 MD5 计算其校验值,并记录到数据包的数据结构中。每个中转节点收到传输数据包时,首先对其解析并进行 MD5 校验,根据校验结果判断收到的数据包是否已被破坏。如果发现数据包被破坏,则丢弃,同时根据该数据包携带的信息向源端发送重传消息,重传消息中携带了请求重传的数据块信息。

6.2　超时重传

在传输源节点为每个数据块构建的数据包在传输时为其记录一个时间戳,目的端每成功接收一个数据包则返回给发送端一个响应消息,响应信息携带了传输成功的数据块的信息,对于收到响应信息的数据块则将其标记为传输成功状态。源端采用轮询方式检查各数据块传输是否超时,如果发现超时,则对相应的数据块发起重传操作。当发现有数据块的重传次数超过最大重传次数的限制时,放弃重传,通知用户传输任务失败,并将传输任务移除。

6.3　动态估计最大超时时间

如果数据包的超时间隔设置太短,则会发生大量不必要的重传操作,传输这些无用的数据包反而会加重网络的负载程度;如果数据包的超时间隔设置太长,一旦发生数据包的丢失现象,过长的重传延迟会严重影响数据传输的效率。Jacobson[16]研究了估算数据包传输超时时间的方法,根据其思想设计了一种动态估计最大超时时间的策略。

首先,在每个传输任务开始前,发送若干携带传输任务信息的数据包进行路径探测,返回的响应信息作为探测结果。如果发现目的节点不可到达,则通知用户传输失败,这样可以防止在目的节点不可达的情况

下盲目传输造成的网络带宽浪费;如果返回消息显示目的节点可达,则记录消息的往返时延(RTT)。然后根据消息数据包大小、往返时延以及实际传输数据包大小,按式(5)计算数据包最大超时时间:

$$TE_{overtime} = \omega * \left(\frac{size_{bp} + size_{ba}}{size_{m}}\right) * \frac{RTT_{m}}{2} \quad (5)$$

其中,$size_{bp}$ 为传输数据包大小;$size_{ba}$ 为后行消息数据包大小;$size_{m}$ 为探测消息大小;RTT_{m} 为探测消息往返时延;ω 为超时系数,根据文献[16],一般 ω 取4。

然后,当传输任务被调度并开始执行后,源节点会不断地收到标识各数据块传输成功的响应消息,它将定期根据数据包发送时间 $time_{bp}$ 和消息返回时间 $time_{ba}$,按式(6)动态更新数据包最大超时时间:

$$TE^{\ell}_{overtime} = \omega * (time_{bp} + time_{ba}) \quad (6)$$

这种策略的优点是:传输过程的可靠性不依赖于集中式的节点控制;动态更新最大超时时间可以实时应对网络传输性能动态变化导致的数据包传输时延差异;出错请求重传可以及时发现传输过程中的错误并及时进行处理。

7 结束语

在研究分析影响端到端数据传输效率的关键问题基础上,基于缓冲区、数据分片、多路并行传输等技术,设计了一种适合端到端数据传输的多路并行传输机制,同时提供了可靠性保证策略,旨在充分利用网络带宽以提高传输效率。

理论分析表明,该机制有效解决了分布式环境下数据传输的访问控制和传输效率等问题,尤其是对于较为复杂的分布式环境下进行的大数据传输,该机制充分利用了整个网络环境的传输带宽,同时解决了中间节点的数据落地问题,为端到端的数据传输提供了高效可靠的保障。

参考文献:

[1] 徐志伟,冯百明,李 伟. 网格计算技术[M]. 北京:电子工业出版社,2004.

[2] Wasson G, Humphrey M. HPC file staging profile[C]//Conference on active media technology. [s. l.]:[s. n.],2008:58-70.

[3] 蒋 成. 混合式对等网络 Kazaa 模型结构及其分析研究[J]. 信息安全与技术,2013,4(6):69-71.

[4] Bengt C, Rune G. The rise and fall of napster-an evolutionary approach[C]//Proceedings of the 6th international conference on computer science. [s. l.]:[s. n.],2009:347-354.

[5] Itou T, Ohsaki H, Imase M. On parameter tuning of data transfer protocol GridFTP in wide-area grid computing[J]. IEICE Technical Report Information Networks, 2004, 104(4): 19-24.

[6] Cannataro M, Mastroianni C, Talia D. Evaluating and enhancing the use of the GridFTP protocol for efficient data transfer on the grid[C]//Lecture notes in computer science. Berlin: Springer,2003:619-628.

[7] Kosar T, Livny M. Stork: making data placement a first class citizen in the grid[C]//24th IEEE international conference on distributed computing systems. [s. l.]: IEEE, 2004: 342-349.

[8] Niederberger R, Allcock W, Gommans L, et al. Firewall issues overview[R]. [s. l.]:[s. n.],2006.

[9] 中国国家网格软件数据网格工作组. 数据网格进展汇报[R]. 北京:中科院计算所,2008.

[10] Sivakumar H, Bailey S, Grossman R L. PSockets-the case for application-level network striping for data intensive applications using high speed wide area networks[C]//ACM/IEEE conference on supercomputing. [s. l.]: IEEE, 2000:38.

[11] Kola G, Livny M. DiskRouter: a flexible infrastructure for high performance large scale data transfers[R]. Ameracan: University of Wisconsin,2003.

[12] Lee J, Dan G, Tierney B, et al. Applied techniques for high bandwidth data transfers across wide area networks[C]//Proceedings of international conference on computing in high energy and nuclear physics. [s. l.]:[s. n.],2011:428-431.

[13] Semke J, Mahdavi J, Mathis M. Automatic TCP buffer tuning[J]. ACM SIGCOMM Computer Communication Review, 2000, 28(4):315-323.

[14] Lakshman T, Madhow U. The performance of TCP/IP networks with high bandwidth-delay products and random loss[J]. IEEE Transactions on Networking, 1997, 5(3):336-350.

[15] Tanenbaum A S. Computer networks[M]. 5th ed. Beijing: China Machine Press, 2012:258-309.

[16] Jacobson V, Karels M J. Congestion avoidance and control[J]. ACM SIGCOMM Computer Communication Review, 1988, 18(4):314-329.