文章编号：2095-2163(2022)01-0124-04 中图分类号：TP391 文献标志码：A

面向集群网络的全序数据传输策略研究

王春辉
(启东市第二中等专业学校，江苏 启东 226200)

摘 要：为了提高集群网络中全序数据传输性能，本文提出了一种新颖的传输框架——TOC，该框架由 TOC-RE 策略和 TOC-Re 策略两部分组成。首先，提出了层次化时间戳聚合机制，使 TOC-RE 策略能够提供尽力而为的服务；其次，设计了丢包处理机制和故障恢复机制，使 TOC-Re 策略实现可靠的传输服务。采用测试平台评估 TOC 的性能，结果表明 TOC 以较低的开销实现了高吞吐量和低延迟，具有一定的可扩展性。

关键词：集群网络；全序数据传输；丢包处理机制

Research on total sequence data transmission strategy for the cluster networks

WANG Chunhui
(Qidong second secondary vocational school, Qidong Jiangsu 226200, China)

【Abstract】 In order to improve the total sequence data transmission performance in the cluster network, this paper proposes a novel transmission framework—TOC, which consists of two parts: TOC-RE strategy and TOC-Re strategy. First of all, this paper proposes a hierarchical time stamp aggregation mechanism to enable the TOC-RE strategy to provide best-effort services. Then, this paper designs a packet loss processing and failure recovery mechanism to enable the TOC-Re strategy to achieve reliable transmission services. Finally, this article uses a test platform to evaluate the performance of TOC. The results show that TOC achieves high throughput and low latency with low overhead, and has certain scalability.

【Key words】 cluster network; total sequence data transmission; packet drop handle mechanism

0 引 言

在诸如集群的分布式系统中，当一台主机读取或写入不同远程主机上的多个对象时，无法保证消息同时到达不同的远程主机，通常需要加锁来实现一致性；另外，分布式数据的多个分片生成日志到多个副本，每个副本可能会以不同的顺序从分片接收日志，从而违反数据一致性。由此可知，数据传输的顺序严重影响着分布式系统的性能。对此，本文提出了一种全序数据传输框架，即 TOC 框架。在 TOC 中，消息按组发送并在虚拟管道中序列化，这样一来不同的接收方进程能够以一致的顺序从发送方进程传递消息。TOC 框架由提供尽力而为数据传输服务的 TOC-BE 策略和提供可靠传输的 TOC-Re 策略组成。

1 尽力而为全序数据传输策略设计

1.1 消息和屏障时间戳机制

尽力而为全序数据传输(TOC-BE)发送方为每条消息分配一个非递减的时间戳，分散传输中的消息具有相同的时间戳。同步主机的单调时间，并使用本地时钟时间作为消息时间戳，消息时间戳确定接收方的交付顺序，接收方按照时间戳的升序发送到达消息。

当接收者传递带有时间戳 T 的消息时，必须确保其已经收到并传递了所有时间戳小于 T 的消息。一种简单的方法是只传输时间戳不减少的消息，并丢弃所有无序消息。然而，由于不同的网络路径具有不同的传播和排队延迟，这种方法会丢弃太多消息。对此，引入了屏障时间戳的概念，屏障时间戳与链路或节点相关联，表示所有未来到达消息的消息时间戳的下限。每个接收者维护自己的屏障时间戳，并传递时间戳小于屏障时间戳的消息。

TOC-BE 利用网络排队结构，使时间戳下限聚合更具可扩展性和效率。TOC-BE 利用可编程交换机来聚合屏障时间戳信息，接收端根据交换机提供的屏障时间戳信息对消息进行重新排序。

在 TOC-BE 中，为每个消息包附加了两个时间戳字段：第一个是消息时间戳字段，由发送方设置，不会被修改；第二个字段是屏障时间戳，由发送方初始化，交换机进行修改。交换机或主机从网络链路 L 接收到屏障时间戳为 B 的数据包时，意味着未来

作者简介：王春辉(1982-)，男，学士，高级讲师，主要研究方向：计算机科学与技术。

收稿日期：2021-10-12

来自链路 L 的到达数据包的消息时间戳和屏障时间戳将大于 B。

为了计算屏障时间戳，发送方使用非递减的消息时间戳，初始化消息中所有数据包的字段。交换机为每个输入链路 $i \in J$ 维护一个寄存器 R_i，其中 J 是所有输入链路的集合；再将具有屏障时间戳 B 的数据包从输入链路 i 转发到输出链路 o 后，交换机首先更新寄存器 $R_i = B$，然后将数据包的屏障时间戳修改为 $B_{new} = \min_{i \in J} R_i$。

每个交换机以这种方式计算屏障时间戳，屏障时间戳通过交换机进行逐跳的聚合，最终接收方获得网络中所有可达主机和链路的屏障。

当接收方收到一个带有屏障时间戳 B 的数据包时，首先将数据包缓存在一个优先级队列中，该队列根据消息时间戳对数据包进行排序。接收者知道所有未来到达数据包的消息时间戳将大于 B，因此其将消息时间戳低于 B 的所有缓冲数据包传递给应用程序进行处理。如果应用程序进程收到时间戳大于 B 的消息，则将其丢弃，并向发送方返回一条 NAK 消息。由此可知，无序的消息不会违反正确性。

为了在 Lamport 逻辑时钟上保持因果顺序，本地时钟时间应该高于传递的时间戳[1]。每个进程都有发送者 S 和接收者 R 的角色，而 R 收到的屏障 T 是从包括 S 在内的所有发送者聚合的。由于本地时钟是单调的，当 R 收到屏障 T 时，S 的时间戳高于 T。

1.2　信标机制设计

空闲的链路会影响屏障时间戳的计算，为了避免空闲链路的影响，会定期在每个链路上发送信标。与消息包不同，信标包只携带屏障时间戳字段，没有有效载荷数据。

主机和交换机都可以发送信标数据包，目的地是其一跳邻居。对于主机生成的信标，屏障时间戳是主机时钟时间。交换机使用本地屏障时间戳来初始化信标的屏障时间戳，当主机或交换机输出链路在信标间隔时间 T_{beacon} 内没有观察到任何消息包时，其会生成一个信标包。

当主机、链路或交换机发生故障时，其邻居的屏障时间戳会停止增加。为了检测故障，每个交换机的每个输入链路都有一个超时计时器，如果在特定的时间内没有接收到信标或数据包，则认为输入链路已失效，并从输入链路列表中删除，移除故障链接后，屏障时间戳恢复增加。

2　可靠全序数据传输策略设计

2.1　丢包处理机制

在可靠全序数据传输（TOC-Re）策略中，当接收者发送带有时间戳 T 的消息时，必须确保已经发送完时间戳少于 T 的所有消息。因此，如果接收方不知道数据包丢失，则无法根据屏障时间戳可靠地传递消息，采用两阶段提交方法来处理数据包丢失。首先发送方将消息放入发送缓冲区，在附加了时间戳字段后发送消息，沿路径的交换机不会聚合数据包的时间戳屏障；接收方将消息存储在接收缓冲区中，并使用 ACK 进行回复。发送方使用 ACK 来检测和恢复数据包丢失；当发送方收集到时间戳小于或等于 T 的数据包的所有 ACK 时，其会发送一个带有提交屏障 T 的提交消息，该提交消息被发送到邻居交换机；每个交换机聚合输入链路上的最小提交屏障，并产生传输到输出链路的提交屏障，当接收者收到提交屏障 T 时，接收者在接收缓冲区中传递小于或等于 T 的消息。

2.2　故障恢复机制

使用网络控制器通过信标超时检测故障。控制器需要确定故障的进程和故障情况，与控制器断开连接的进程被视为故障。例如，如果主机出现故障，则其上的所有进程都被视为故障。当进程故障时，无法可靠地确定进程最后一次提交和最后一次传输的时间戳。从提交时间戳到将时间戳传输给接收者会产生传播延迟，因此有可能找到一个由失败进程 P 提交但没有传输到任何接收者的时间戳 T。这样一来，所有接收者会接收缓冲区中来自 P，且在时间戳 T 之前的消息，并丢弃 T 之后的消息。P 的故障时间戳是 P 的所有邻居报告的最大最后提交时间戳。如果多个故障同时发生，将尝试在网络中找到一组无故障的节点，并将故障节点和没有故障的接收器分开。

邻居检测到故障后会将最后提交时间戳 T 通知控制器，控制器确定故障的进程及其故障时间戳，将故障的进程 P 及其故障时间戳 T 广播给所有正确的进程。每个正确的进程都会丢弃接收缓冲区中时间戳高于 T，由 P 发送的消息。每个正确的进程丢弃发送缓冲区中发送到 P 的消息。如果丢弃的消息处于分散传输状态，需要中止分散传输的过程，即召回分散传输到其他接收者的消息；发送方向这些接收方发送召回消息，每个接收方丢弃接收缓冲区中的消息，并向发送方响应 ACK；控制器从所有正

确的进程中收集完成信息,通知网络组件,从失败的组件中删除输入链路。

如果网络故障影响 S 和 R 之间的连通性,那么 S 会要求控制器将消息转发给 R,并等待来自控制器的 ACK;如果控制器也无法传递消息,则 R 将被认为是故障,并记录无法传递的召回消息;如果控制器收到召回消息的 ACK 但无法将其转发给 S,则 S 将被认为故障。总而言之,如果一个进程在特定的时间内没有响应控制器,则可以认为该进程发生故障。

当进程从故障中恢复,则该进程需要传递或丢弃接收缓冲区中的消息。控制器会通知进程其故障,进程联系控制器以获取主机故障通知,传递缓冲消息后,恢复的进程需要作为新进程加入 TOC-Re。

3 全序数据传输实现

3.1 主机端实现

在终端主机上实现一个 TOC 库,该库建立在 RDMA 协议上。TOC 从处理器周期计数器中获取时间戳,并将其分配给软件中的消息,由于 RDMA 在不同队列中缓存消息,无法确保网卡到交换机链路上时间戳的单调性。因此,使用 RDMA 的不可靠服务(UD),将每条 TOC 消息分割成一个或多个 UD 数据包。

TOC 在软件中实现端到端的流量和拥塞控制。当目标进程第一次被访问时,其会与源进程建立连接,并为其提供一个接收缓冲区,其大小为接收窗口。数据包序列号(PSN)用于丢失检测和碎片整理。拥塞控制采用 DCTCP,其中显示拥塞控制标记位于 UD 数据包部[2]。当应用程序进行发送传输时,其存储在发送缓冲区中。每个目的地都维护一个发送窗口,是接收窗口和拥塞窗口中的最小值。当发送缓冲区中某个分散的所有消息都在相应目的地的发送窗口内时,其会附加当前时间戳并发送出去,即当分散传输的某些目的地或网络路径拥塞时,其会被阻止在发送缓冲区中,而不是减慢整个网络的速度。为了避免活锁,散射从发送窗口获取信用。如果目的地的发送窗口不足,则分散传输将保持在等待队列中,而不释放信用,确保最终可以发送大量分散,但代价是浪费可用于无序发送其他分散的信用。

TOC 中的 UD 数据包添加了 24 个字节的头部:消息时间戳、屏障时间戳、提交屏障时间戳、数据包序列号、一个操作码和一个标记消息结束的标志。

3.2 网络内处理

使用 P4 实现网络内处理,并将其编译到 Tofino[3-4]。一个 Tofino 交换机有 4 个管道、4 个输入端口和 4 个输出端口,每个端口都独立地计算屏障时间戳。TOC 的每个输入链路需要 2 个状态寄存器,分别存储两个屏障,以实现 TOC-BE 和 TOC-Re 策略。对于每个数据包,输入链路的屏障寄存器在交换机的第一个状态流水线阶段更新。由于每个阶段只能计算两个屏障中的最小值,因此交换机使用带有 $O(\log N)$ 流水线阶段的寄存器二叉树来计算最小链路屏障 B_{new}。在最后的流水线阶段,数据包中的屏障时间戳字段更新为 B_{new},控制平面软件会例行检查链路屏障,并在链路屏障明显滞后时报告故障。

对于没有可编程交换芯片的交换机,在交换机的处理器上实现网络内处理。商用交换机不能处理数据平面的数据包,但其处理器可以处理控制平面的数据包。与服务器处理器和网卡相比,交换机处理器通常具有较低的计算能力,较低的带宽。由于交换机处理器无法处理每一个数据包,数据包直接由交换芯片转发,处理器会定期在每个输出链路上发送信标。信标的屏障时间戳存储在每个输入链路的寄存器中,处理器上的线程会定期计算链路屏障的最小值并向所有输出链路广播新的信标。

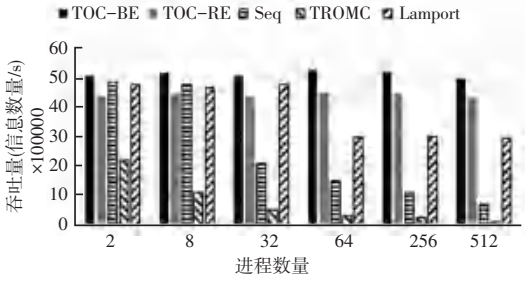
如果交换机供应商不提供交换机处理器的公开访问接口,可以将信标处理任务卸载到主机,为每个网络交换机指定一个主机代表。信标包使用单侧 RDMA 写入来更新主机代表上的屏障,定期计算最小屏障时间戳,并将其广播给下游主机代表。

4 实验评估

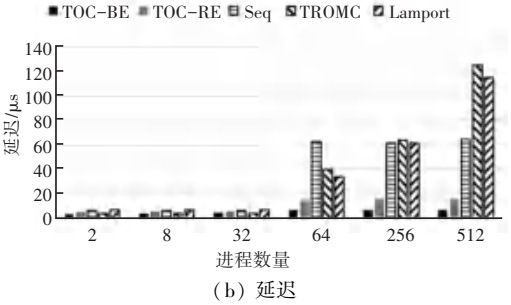
实验评估平台由 10 台 Arista 交换机和 32 台服务器组成[5],每台服务器都配置有 2 个至强 E5-2650 处理器和一个运行 RoCEv2 的迈络思 ConnectX-5 网卡。指定一个处理器作为每个交换机和网卡的主机代表来处理信标,主机代表直接连接到交换机,信标包不需要绕道。在小规模实验(2~32 个进程)中,每个进程都运行在不同的服务器上,每个进程使用一对线程分别进行发送和接收。对于大规模的实验(64~512 个进程),每个服务器托管相同数量的进程。

TOC 的可扩展性能如图 1 所示。其中图 1(a)展示了 TOC 与其他全序广播策略的吞吐量对比。每个进程向所有进程广播消息,随着进程数量的增加,TOC 仍然可以实现较高的吞吐量。TOC 的吞吐量受处理器处理和 RDMA 消息传递速率的限制,由于需要确保传输的可靠性,TOC-Re 的吞吐量要稍低于 TOC-BE,Seq 策略会引入额外的网络延迟,当

Seq 的吞吐量饱和并出现拥塞时，延迟会急剧增加，如图 1(b)所示。由于仅使用一个进程进行传输，信令策略的吞吐量很低。



(a) 吞吐量对比



(b) 延迟

图 1　可扩展性评估

Fig. 1　Evalutation of the scalability

模拟了接收者随机丢弃消息的场景，以评估数据包丢失对延迟的影响，如图 2 所示。该场景中，进程的数量为 512 个，当丢包率高于 0.001%时，TOC-BE 和 TOC-Re 的延迟均开始增长。在 TOC-BE 和 TOC-Re 中，链路上丢失的信标数据包都会延迟屏障的传输。在 TOC-Re 中，丢失的消息将触发重传，因此 TOC-Re 对丢包更为敏感。

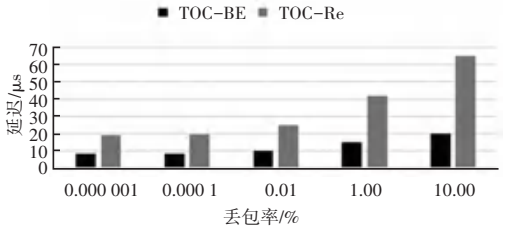


图 2　丢包与延迟的关系

Fig. 2　Relationship between packet loss and delay

由于 TOC 的信标间隔非常小，因此 TOC 的故障检测通常比心跳超时机制更快，如图 3 所示。如果在 10 个信标间隔(即 30 μs)内未收到信标，则检测到故障。核心链路和核心交换机的故障在大多数的情况下并不影响网络连通性，而主机和机柜交换机故障会导致进程与系统断开连接，因此恢复需要更长的时间。TOC 的处理器开销有两部分：接收器的重新排序和交换机的信标处理，如图 4 所示，消息传递吞吐量会随着重新排序消息的增多而略有下降。虽然最大发送和接收缓冲区大小随延迟线性增加，但在缓冲区最多仅占用不超过 3.5 MB。

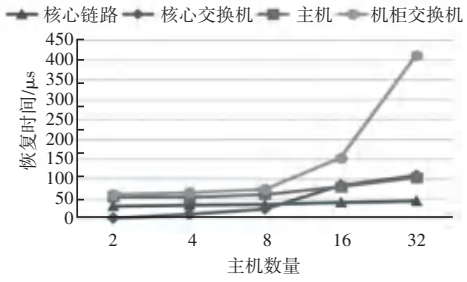


图 3　恢复时间

Fig. 3　Recovery time

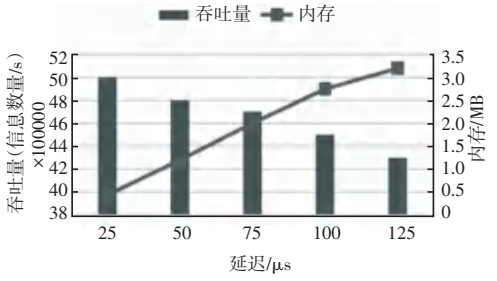


图 4　处理器开销

Fig. 4　CPU overhead

5　结束语

本文提出了用于网络集群的全序通信策略 TOC，该策略以发送方的时钟时间顺序进行消息传输，通过利用可编程的集群网络将聚合顺序信息与转发数据包分开，来实现可扩展性和效率。实验评估表明，TOC 能够提高全序数据传输的效率，还具有较低的故障恢复时间和开销。

参考文献

[1] SHUKUR H, ZEEBAREE S R M, AHMED A J, et al. A state of art survey for concurrent computation and clustering of parallel computing for distributed systems[J]. Journal of Applied Science and Technology Trends, 2020, 1(4): 148-154.

[2] CHEN C, FANG H C, IQBAL M S. Qostcp: Provide consistent rate guarantees to tcp flows in software defined networks[C]//ICC 2020 - 2020 IEEE International Conference on Communications (ICC). IEEE, 2020: 1-6.

[3] BOSSHART P, DALY D, GIBB G, et al. P4: Programming protocol-independent packet processors[J]. ACM SIGCOMM Computer Communication Review, 2014, 44(3): 87-95.

[4] AGRAWAL A, KIM C. Intel tofino2 - a 12.9 tbps p4-programmable ethernet switch[C]//2020 IEEE Hot Chips 32 Symposium (HCS). IEEE Computer Society, 2020: 1-32.

[5] BAI W, CHEN K, HU S, et al. Congestion control for high-speed extremely shallow-buffered datacenter networks[C]// Proceedings of the First Asia-Pacific Workshop on Networking. 2017: 29-35.