由表 1 可见,该咨询类别判断模型在各咨询类别下均有较好的判别效果,说明模型中句子的咨询类别度量方式在进行咨询类别判断时是有效的.

由于句对匹配与否也是一个二分类问题,这里同样使用常见的准确率、精确率、召回率及 $F1$ 值作为评价准则.同时使用信息系统中的 recall@n 进行模型效果评价.实验结果如表 2 所示.

表 2 句对匹配度模型对比效果

Tab.2 Contrastive effect of sentence pair matching

模型	准确率	精准率	召回率	$F1$ 值	R_2@1	R_5@1
Ubuntu-LSTM	0.709	0.696	0.696	0.696	0.778	0.519
SMN	0.728	0.750	0.648	0.695	0.809	0.568
weighted-SMN	0.745	0.766	0.674	0.717	0.817	0.578
CTASM	0.756	0.777	0.688	0.729	0.834	0.608

由实验结果可以看到,CTASM 模型在进行句对匹配度建模时,在准确率、精准率、$F1$ 值及 R_2@1 值、R_5@1 值上的效果均好于其他对比模型.仅在召回率上略低于 Ubuntu-LSTM 模型.说明 CTASM 模型是有效的,模型中使用的关键词加权机制及"从咨询类别的角度进行单词的相似度及相关度计算"这一策略有助于更好地学习心理咨询领域内对话匹配模式.

实验结果中,SMN 在多个指标上效果均好于 Ubuntu-LSTM 模型,说明在进行建模的时候仅仅使用句子的单一向量表示是不够的,利用句子编码过程中生成的隐状态信息、考察句对在句子及单词两个层面上的相似度可以更好地学习句对匹配的模式.

由 weighted-SMN 模型效果好于 SMN 模型可以看出,通过对句子的咨询类别进行判断,并由此找到句子中的关键词,对相似度矩阵进行加权这种方式对句对的匹配度判断是有帮助的.从模型的角度来说,SMN 模型是 weighted-SMN 模型的一个特例,当 weighted-SMN 中权值参数 α 设为 0 时,weighted-SMN 即退化为 SMN 模型,因此从这一角度来说 weighted-SMN 模型要优于未加权的 SMN 模型.

由 CTASM 模型效果好于 weighted-SMN 模型可以看出,将单词从咨询类别这个角度进行向量化及考察句对中单词之间的相关度也有利于匹配度的判断.该策略可以从咨询类别这个角度将语义上不那么相似的单词,从咨询类别这个角度相似化,使模型可以从多个角度对句对匹配模式进行学习.

4 结论

本文提出了一种应用于校园心理咨询的对话匹配度预测模型.模型首先对句子的咨询类别进行判断,并在咨询类别的指导下,从多个角度对句子及单词的相似度及相关度矩阵进行计算,最后通过卷积神经网络对校园心理咨询对话中句对的匹配模式进行学习.实验结果表明,本文提出的 CTASM 模型在校园心理咨询相关语料上的效果要优于比较模型.这说明 CTASM 模型中寻找关键词并进行加权的机制以及从咨询类别这个角度计算相似度及相关度矩阵的方式,可以帮助提高心理咨询领域句对匹配模式的学习效果.

CTASM 模型可以应用于心理咨询对话系统中,提供聊天服务,通过对来访者发来的信息进行回复,对具有轻度心理问题及心理紊乱的人进行辅导,最终达到情感慰藉与情感支持的作用.同时模型也具有一定的通用性,对于其他专业领域,只要领域内语料可以按类别进行划分,CTASM 模型就可以应用于该语料上进行句对匹配度建模.

参考文献(References)

[1] 黄红,张佩珍.大学生心理行为指导[M].上海:上海大学出版社, 2003.

[2] 张伟男,刘挺.聊天机器人技术的研究进展[J].中国人工智能学会通讯,2016,1: 17-21.

[3] CHEN H, LIU X, YIN D, et al. A survey on dialogue systems: Recent advances and new frontiers[J]. ACM SIGKDD Explorations Newsletter, 2017, 19 (2): 25-35.

[4] GANGADHARAIAH R, NARAYANASWAMY B M, ELKAN C. Achieving fluency and coherency in

task-oriented dialog[J]. Artificial Intelligence, 2017, arXiv:1804.03799.

[5] LI J, GALLEY M, BROCKETT C, et al. A diversity-promoting objective function for neural conversation models[J]. OALib Journal, 2015, arXiv:1510.03055.

[6] MOU L, SONG Y, YAN R, et al. Sequence to backward and forward sequences: A content-introducing approach to generative short-text conversation[J]. Machine Learning, 2016, arXiv: 1607.00970.

[7] XING C, WU W, WU Y, et al. Topic aware neural response generation[J]. 2016, arXiv:1606.08340.

[8] ZHOU H, HUANG M, ZHANG T, et al. Emotional chatting machine: Emotional conversation generation with internal and external memory[J]. 2017, arXiv: 1704.01074.

[9] LI J, GALLEY M, BROCKETT C, et al. A persona-based neural conversation model[J]. 2016, arXiv: 1603.06155.

[10] LI J, MILLER A H, CHOPRA S, et al. Learning through dialogue interactions[J]. 2016, arXiv: 1612.04936.

[11] LI X, MOU L, YAN R, et al. StalemateBreaker: A proactive content-introducing approach to automatic human-computer conversation[J]. 2016, arXiv: 1604.04358.

[12] ZHOU H, HUANG M, ZHANG T, et al. Emotional chatting machine: emotional conversation generation with internal and external memory[J]. 2017, arXiv: 1704.01074.

[13] ASGHAR N, POUPART P, HOEY J, et al. Affective neural response generation[J]. 2017, arXiv: 1709.03968.

[14] LOWE R, POW N, SERBAN I, et al. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems[J]. 2015, arXiv:1506.08909.

[15] LU Z, LI H. A deep architecture for matching short texts[C]//Advances in Neural Information Processing Systems. Lake Tahoe, USA: NIPS Press, 2013: 1367-1375.

[16] HU B, LU Z, LI H, et al. Convolutional neural network architectures for matching natural language sentences[C]//Advances in neural information processing systems. Montreal, Canada: NIPS Press, 2014: 2042-2050.

[17] ZHOU X, DONG D, WU H, et al. Multi-view response selection for human-computer conversation [C]// Proceedings of the Conference on Empirical Methods in Natural Language Processing. Austin, USA: IEEE Press, 2016: 372-381.

[18] WU Y, WU W, LI Z, et al. Topic augmented neural network for short text conversation[J]. 2016, CoRR abs/1605.00090.

[19] WU Y, WU W, XING C, et al. Sequential matching network: A new architecture for multi-turn response selection in retrieval-based chatbots[C]// Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics. Vancouver, Canada: IEEE Press, 2017, 1: 496-505.

[20] ELMASRI D, MAEDER A. A conversational agent for an online mental health intervention[C]//International Conference on Brain and Health Informatics. Springer, 2016: 243-251.

[21] CHITTARANJAN G, BLOM J, GATICA-PEREZ D. Mining large-scale smartphone data for personality studies[J]. Personal and Ubiquitous Computing, 2013, 17(3): 433-450.

[22] WEI H, ZHANG F, YUAN N J, et al. Beyond the words: Predicting user personality from heterogeneous information[C]//Proceedings of the 10th ACM international Conference on Web Search and Data Mining. Cambridge, UK: ACM Press, 2017: 305-314.

[23] GUO A, MA J. Archetype-based modeling of persona for comprehensive personality computing from personal big data[J]. Sensors, 2018, 18(3): 684.

第48卷第9期 中国科学技术大学学报 Vol.48,No.9
2018年9月 JOURNAL OF UNIVERSITY OF SCIENCE AND TECHNOLOGY OF CHINA Sep. 2018

文章编号:0253-2778(2018)09-0748-07

一种Ceph块设备跨集群迁移算法

邵曦煜,李 京,周志强

(中国科学技术大学计算机科学与技术学院,安徽合肥 230027)

摘要:在虚拟机全系统在线迁移中,由于镜像文件数据量巨大,对整个迁移过程的效率有着关键的影响,因此优化迁移时间,成为虚拟机迁移技术的研究热点.对于以分布式存储系统(其中较为常见的是Ceph块设备)作为镜像文件存储方式的虚拟机进行迁移时,镜像文件需要经过源存储节点到源计算节点,再到目的计算节点,最后到目的存储节点.这种方式忽略了底层存储系统特点可以给迁移带来的好处,针对上述问题,提出了一种Ceph块设备跨集群迁移算法,采用源存储节点并行向目的存储节点迁移数据的方式,利用了存储节点的计算和网络能力.实验表明,该算法加快了迁移速度,同时适当增加存储节点数目能进一步提升算法效率.

关键词:虚拟机;在线迁移;全系统;分布式存储;Ceph块设备

中图分类号:TP393 **文献标识码**:A doi:10.3969/j.issn.0253-2778.2018.09.009

引用格式:邵曦煜,李京,周志强. 一种Ceph块设备跨集群迁移算法[J]. 中国科学技术大学学报,2018,48(9):748-754.

SHAO Xiyu, LI Jing, ZHOU Zhiqiang. Migration algorithm for Ceph block device cross clusters[J]. Journal of University of Science and Technology of China, 2018,48(9):748-754.

Migration algorithm for Ceph block device cross clusters

SHAO Xiyu, LI Jing, ZHOU Zhiqiang

(*School of Computer Science and Technology, University of Science and Technology of China, Hefei* 230027, *China*)

Abstract: The migration time of huge image files is vital to the efficiency of the whole-system on-line migration of a virtual machine. Therefore, optimizing the migration time has become a hot research area in virtual-machine migration technology. For virtual machines based on a distributed storage system, in which the more common one is Ceph block device, image data must go from the source storage nodes to the source client nodes, then to the destination client nodes, and finally to the destination storage nodes. This way ignores the benefit from the storage system to the migration. Given the above problems, a migration algorithm for Ceph block device cross clusters is effective. Image data go from the source storage nodes to the destination storage nodes in parallel, which uses the network of storage nodes. The result of the experiment shows that this algorithm shortens the migration time, and a few more storage nodes can improve efficiency of this algorithm.

Key words: virtual machine; live migration; whole-system; distributed storage; Ceph block device

收稿日期:2018-03-24;**修回日期**:2018-05-15

作者简介:邵曦煜,男,1993年生,硕士研究生,研究方向:云计算,E-mail:zjjhsxy@mail.ustc.edu.cn

通讯作者:李京,1966年生,博士/教授.E-mail:lj@ustc.edu.cn

0　引言

虚拟机迁移就是把一台主机(源主机)上运行着的虚拟机迁移至另一台主机(目的主机)上运行,这涉及虚拟机内存状态的迁移和 CPU 状态的迁移,在把虚拟机的这些运行状态从源主机向目的主机传输完成之后,再在目的主机上恢复运行,完成整个虚拟机迁移的过程.虚拟机迁移的技术为服务器虚拟化的技术提供了一种非常简便的方法,也为当前流行的许多虚拟化产品,如 QEMU-KVM 等,提供了各自的迁移工具.

虚拟机在线迁移则指的是在整个迁移的过程当中,虚拟机始终保持运行状态,仅仅有非常短暂的停机时间,虚拟机中的服务始终可以响应用户的请求[1-3].在很多应用场景之中,待迁移的虚拟机所在的环境,也许不存在网络共享存储设备.当这类场景中的虚拟机需要迁移时,磁盘数据也必须同时被迁移,这种包括磁盘数据在内的虚拟机迁移被称作是跨存储池虚拟机迁移,也被称作全系统迁移[4].由于磁盘迁移的数据量巨大,对整个迁移过程的效率有着关键的影响,因此如何优化磁盘数据迁移所需的时间,也成为虚拟机迁移技术的研究热点.

现在已有很多工作研究了磁盘数据迁移的优化方式.文献[5]提出的策略是先将源主机的内存状态和 CPU 状态迁移至目的主机,然后在目的主机上将虚拟机恢复运行,再根据虚拟机读写磁盘数据的请求,从源主机读取数据,这种策略被称作按需取块,好处在于迁移时间可以达到基于共享存储的虚拟机在线迁移的水平,但是需要长期依赖源主机,降低了虚拟机的可用性,因此无法适用于源主机需要关闭或降低负载的场景.文献[6]提出的策略是先将源主机的磁盘数据预迁移至目的主机,同时截获该阶段中所有对磁盘数据的写请求,并按顺序转发保存到目的主机,在预迁移完磁盘数据之后,目的主机重做保存的这些写操作.这种策略是一种预迁移和写操作回放相结合的同步策略,好处在于减少迁移时间,但是对于写密集型的应用,目的主机在恢复之后可能会有较长时间的磁盘 IO 阻塞.这些策略都缩短了迁移时间,降低了迁移数据量,对磁盘数据的迁移做出了一定的优化.

传统的迁移方法一般均忽略了底层存储系统的特点可能给镜像文件的迁移带来的好处.在分布式存储系统被广泛应用的今天,虚拟机磁盘数据(也即镜像文件)的存储位置可以分布在多个存储节点上.在传统的迁移方式中,镜像文件需要从源存储节点到源计算节点,再到目的计算节点,最后到目的存储节点.如果可以利用存储节点的计算和网络能力以及分布式存储多副本和分块存储的特点,将镜像文件直接从源存储节点传输到目的存储节点,可以降低两端计算节点的负载,提高网络利用率,加快镜像迁移.其中,较为常见被作为虚拟机镜像文件使用的分布式存储系统就是 Ceph 块设备,研究虚拟机镜像文件的迁移算法,可以将 Ceph 块设备的迁移算法作为特例来研究.

本文提出了一种 Ceph 块设备跨集群迁移算法,可用在基于 Ceph 分布式存储、并且存储节点具有和集群外节点进行网络通信的能力.该算法采用源存储节点并行向目的存储节点迁移数据的方式,利用了存储节点的计算和网络能力.实验表明,该算法加快了迁移速度,同时适当增加存储节点数目能进一步提升了算法效率.在虚拟机在线迁移系统中,只需要循环调用该算法,即可实现镜像文件的在线迁移.

1　研究背景

为了有助于更好地理解本文所提算法,本节将较为详细地介绍 Ceph 集群的系统架构、Ceph 数据寻址流程、Ceph 数据读写流程.

Ceph 是一个高可靠的、高可扩展的、高性能的分布式存储系统[7-8].Ceph 的系统逻辑结构如图 1 所示,Ceph 集群主要由 3 种节点组成:其一是若干个监控节点 monitor,负责系统状态的检测和维护;其二是为数众多的具有计算能力的存储节点 OSD (object storage device),负责完成数据存储和维护功能;其三是若干个客户端节点 client,用于发出对 Ceph 块设备的读写请求.这 3 种节点的分类是从功

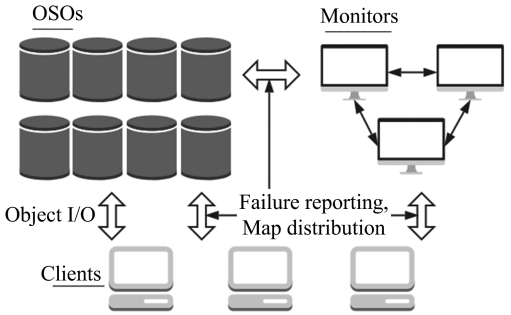

图 1　Ceph 系统逻辑结构

Fig.1　Logic structure of Ceph

能上而不是物理上进行的，因此一个物理机可以作为多种节点同时使用.

在 Ceph 集群中，数据的存储位置不再需要通过全局查表获得，而是利用 CRUSH 算法计算得到[9]，这样做不需要额外的管理设备，避免了集群对中心节点的依赖，实现了无中心结构的分布式集群. Ceph 的数据寻址流程如图 2 所示，对象(object)是 Ceph 实现底层存储组织管理的基本单元，归置组(placement group，PG)是对对象的存储进行组织和位置映射的逻辑分组.一个归置组负责组织若干个对象，但一个对象只能被映射到一个归置组中，因此归置组和对象是一对多的映射关系.同时，一个归置组被映射到多个 OSD 节点上进行存储，而且每个 OSD 节点上都会承载大量的归置组，因此归置组和 OSD 节点是多对多的映射关系.对于一个文件，Ceph 首先将其分为大小相等的若干个对象，默认大小为 4 M，其中最后一个对象的大小可以不同，每个对象有各自唯一的对象 ID，然后对对象 ID 进行哈希，用归置组数目对哈希值取模得到归置组 ID，再根据归置组 ID 利用 CRUSH 算法计算获得一组 OSD 编号，即为数据所有副本的存储位置，这就是数据寻址的整个过程.

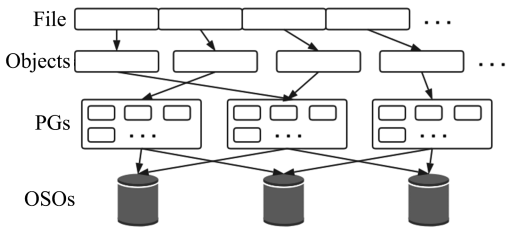

图 2 Ceph 数据寻址流程

Fig.2 Data addressing process of Ceph

Ceph 采用多副本的方式来进行数据的安全维护，以数据具有 3 副本为例，每个归置组会映射到 3 个不同的 OSD 节点上，存储为 3 个副本，其中一个为主副本，其余的为次副本.以一个对象为例，Ceph 的数据写入流程如图 3 所示.由于 Ceph 采用的是强一致性的副本策略，因此需要在所有副本都完成写入操作之后，才会向客户端发送完成写入操作的反馈.客户端首先在本地通过 CRUSH 算法完成寻址流程，获取一组 OSD 编号，然后客户端与主 OSD 节点进行通信，发起写入请求，主 OSD 节点接收到写入请求后，在本地利用 CRUSH 算法获取其余 OSD 节点的编号，再分别向其余 OSD 节点发起写入请求，其余 OSD 节点完成数据的写入操作之后，各自向主 OSD 节点发送应答消息，当主 OSD 节点确认其余 OSD 节点均完成写入操作之后，再完成数据的写入，并向客户端返回写入操作完成的确认信息.如果需要读取数据，客户端只需要完成相同的寻址过程，并直接和主 OSD 通信即可读取数据.

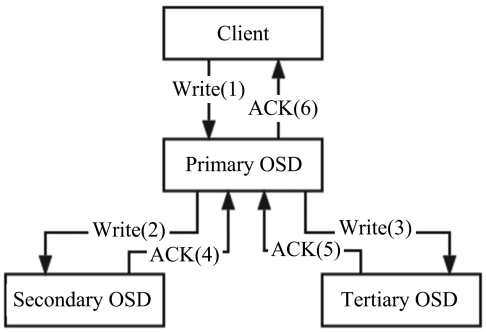

图 3 Ceph 数据写入流程

Fig.3 Data writing process of Ceph

2 Ceph 块设备跨集群迁移算法

在传统的迁移算法中，镜像文件迁移的整个流程如图 4 所示.源客户端节点需要先将镜像文件从源 OSD 节点读取到本地，再向目的客户端节点进行传输，最后写入到目的 OSD 节点.这种方法的好处主要是保证了镜像文件的存储方式对用户是透明的.

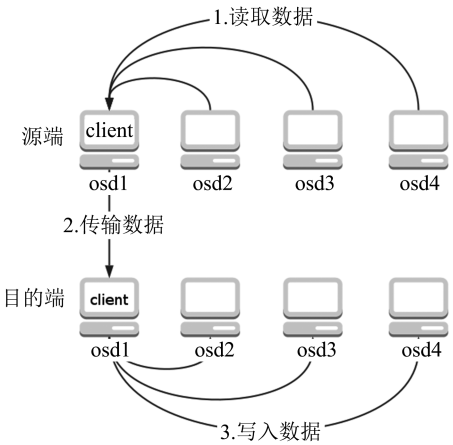

图 4 传统迁移算法流程

Fig.4 Process of traditional migration algorithm

本文提出一种 Ceph 块设备跨集群迁移算法，将镜像文件从源 OSD 节点直接传输到目的 OSD 节点，并且实现了多个源 OSD 节点并行向目的 OSD 节点传输的方式[10]，数据总传输量为镜像文件的一个副本，与传统的迁移算法相同，但是该算法减少了源客户端节点在集群内读取镜像文件的数据传输，即为图 4 中第一阶段的数据传输量，同时并行传输

的方式也相当于增大了网络带宽，增加了单位时间内的数据传输量，加快了图 4 中第二阶段的数据传输，因此可以缩短迁移时间.由于数据传输的连接方式为 TCP 连接，数据读写的方式为 Ceph 块设备提供的读写接口，保证了算法的可靠性.该算法需要利用 OSD 节点进行跨集群网络通信，因此并不能适用于所有的应用场景，如广域网环境中，OSD 节点作为专用的存储节点.为了保障存储节点数据的安全和隐私，也许只能和处于同一局域网的集群内节点进行通信，而不能对另一个集群的 OSD 节点进行数据传输，此时该迁移算法就无法发挥作用.

由此可见，本文提出的 Ceph 块设备跨集群迁移算法主要的应用场景为：两个 Ceph 集群的 OSD 节点具备互相进行网络通信能力的情况，例如两个同属校园网的 Ceph 集群，由于校园网的特点，在该网络中的节点可以互相进行网络通信，其中一个集群的所有节点需要进行停机维护，则将其上的数据迁移至另一个集群进行存储；或者是由于条件所限，集群内的物理主机同时扮演客户端节点和 OSD 节点两个角色，由于客户端节点的功能需求，本身就具备对集群外节点进行网络通信的能力.

该算法主要分为 3 个阶段：初始化阶段、迁移阶段、结束阶段.

在初始化阶段，源客户端节点和目的客户端节点建立连接进行信息交互，整个流程如图 5 所示.首先，为了确保源镜像文件的数据能够全部迁移至目的镜像文件，目的镜像的容量需要大于等于源镜像的容量，目的客户端节点需要从源客户端节点获取该参数并进行检查，以便迁移算法能够正确运行.确认无误之后，目的客户端节点利用 CRUSH 算法，计算出目的镜像每个对象应在的主 OSD 节点的 IP 地址，随后将这项信息全部传输至源客户端节点进行存储.源客户端节点也利用 CRUSH 算法，计算出源镜像每个对象应在的主 OSD 节点的 IP 地址，以便在迁移阶段中使用这项信息.

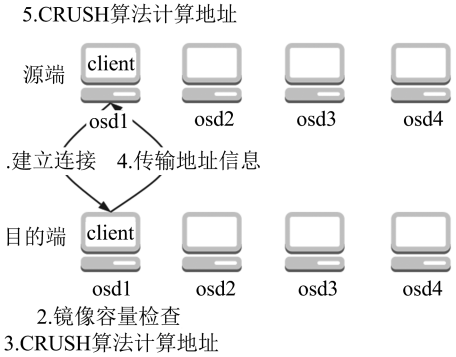

图 5　Ceph 块设备跨集群迁移算法初始化阶段的流程

Fig.5　Process of migration algorithm for Ceph block device cross cluster during the initialization phase

在初始化阶段，涉及源客户端节点和目的客户端节点的准备工作，主要包括源客户端节点和目的客户端节点连接的建立、算法可行性的检查、源镜像文件每个块的定位、目的镜像文件每个块的定位以及信息的传输等功能.该阶段的伪代码如下：

源客户端节点：

1)connect to destination client node
2)send source image size
3)receive ACK
4)if (ACK == ERROR)
5)return ERROR
6)else
7)receive destination OSD addresses
8)for i=1 to BLOCK_NUM do
9)calculate OSD addresses using CRUSH

目的客户端节点：

1)accept connection from source client node
2)receive source image size
3)check image size
4)send ACK
5)if (ACK ! = ERROR)
6)for i=1 to BLOCK_NUM do
7)calculate OSD addresses using CRUSH
8)send destination OSD addresses

在迁移阶段，整个流程如图 6 所示.源客户端节点根据 offset 和 length 两个参数确定本次传输任

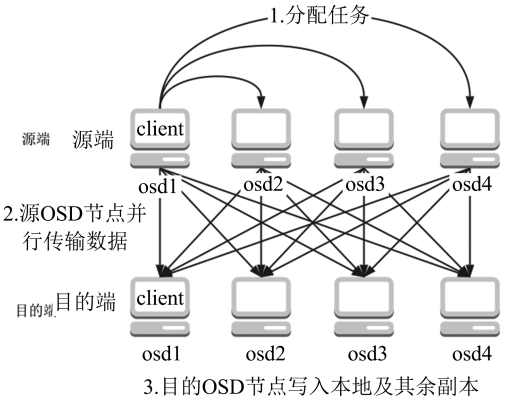

图 6　Ceph 块设备跨集群迁移算法迁移阶段的流程

Fig.6　Process of migration algorithm for Ceph block device cross cluster during the migration phase

务,offset 表示待迁移数据的起始位置在镜像文件中的偏移量,length 表示需要迁移的数据长度.源客户端节点依据这两个参数获取本次需要迁移的对象序号,并获取对象所在的主 OSD 节点(该信息已在初始化阶段中计算获取),将该对象的迁移任务分配给该 OSD 节点.在分配完所有待迁移对象之后,源客户端节点通知每个分配到任务的 OSD 节点,由这些 OSD 节点并行进行对应数据的传输.OSD 节点接收到来自客户端的迁移消息通知以及迁移任务分配之后,按序完成所有待迁移对象的迁移任务.每项任务的信息包括数据在镜像文件中的相对位置以及目的 OSD 节点的地址,源 OSD 节点读取对应的数据,并和目的 OSD 节点建立通信,传输数据.目的 OSD 节点接收到数据之后写入本地以及其余的数据副本,完成一个对象的迁移过程.

在迁移阶段,涉及的是源客户端节点对迁移任务的分配工作以及 OSD 节点数据的传输和接收工作,主要包括了源客户端节点对迁移数据的分块、数据块传输的目的 OSD 节点的选择、对源 OSD 节点的任务分配、OSD 节点的数据传输和读写等功能.该阶段的伪代码如下:

源客户端节点:

1)get BLOCK_NUM by offset and length

2)for i=1 to BLOCK_NUM do

3)calculate offset and length of the block

4)allocate destination OSD node

5)allocate source OSD node to send this block

6)for i=1 to OSD_NUM do

7) send all the blocks' information to source OSD to transmit them

源 OSD 节点:

1) receive all the blocks' information from source client node

2)for i=1 to BLOCK_NUM do

3) send offset and length of the block to destination OSD node

4) read data by librbd

5) send data to destination OSD node

在结束阶段,关闭所有源 OSD 节点和目的 OSD 节点之间的连接,并销毁迁移相关的信息,完成迁移算法的收尾工作.在该算法中,源 OSD 节点和目的 OSD 节点之间的连接采用的是迁移阶段中按需连接,结束阶段中统一关闭的方式,采取这种方式的好处是减少了建立连接以及断开连接的次数,节约了时间和资源开销.

3 实验结果与分析

实验使用了 8 台物理机,在其上搭建了两个 Ceph 集群,每个集群均由 4 个节点组成,每个节点共同作为 monitor 节点和 OSD 节点来使用,同时每个集群中各有一个节点再作为客户端节点,Ceph 集群数据的副本数设定为 3,Ceph 的版本为 v9.2.1,所有节点的系统环境均为 Ubuntu14.04,拥有 2 个型号为 Intel(R) Xeon(R) CPU E5-2660 v3 并且主频为 2600 MHz 的 10 核 CPU,内存为 128 G,带宽为 1000 Mbps.

实验时,我们在源 Ceph 集群上创建镜像,并写满数据,用于迁移,在目的 Ceph 集群创建镜像并留空,用于接收迁移数据.我们分别利用本文提出的 Ceph 块设备迁移算法和传统的通过客户端节点读取数据并迁移的算法,将源镜像文件迁移至目的集群,以评价 Ceph 块设备跨集群迁移算法的性能.随后将两个集群各自减少一个 OSD 节点,成为 3 个 OSD 节点的集群,探究 OSD 节点数目对传统迁移算法和 Ceph 块设备跨集群迁移算法的影响.

本组对比测试中,两个集群 OSD 节点的数目均为 4 个,待迁移的镜像文件的大小为 10 G,两种算法分别迁移了 5 次,迁移时间结果如表 1 所示,同时测得使用 Ceph 块设备跨集群迁移算法时目的 OSD 节点的平均带宽利用率为 37.87%.

表 1 四个 OSD 节点的集群中 10 G 镜像文件的迁移时间对比

Tab.1 Comparison of migration time for 10 G image file cross clusters with four OSD nodes

次数	Ceph 块设备跨集群迁移算法迁移时间/s	传统迁移算法迁移时间/s
1	313	794
2	312	775
3	306	789
4	323	774
5	315	780
平均时间	313.8	782.4

本组对比测试中,待迁移的镜像文件的大小为 20 G,其余条件和第一组实验相同,两种算法分别迁移了 5 次,迁移时间结果如表 2 所示.