广域虚拟数据空间中边缘缓存系统的研究与实现

霍建同[1,2]，肖利民[1,2]，霍志胜[1,2]，徐耀文[1,2]
1. 软件开发环境国家重点实验室，北京 100191；2. 北京航空航天大学计算机学院，北京 100191

摘要
针对广域虚拟数据空间系统中边缘客户端访问和共享远程数据时，数据冗余传输造成大量网络带宽浪费的问题，通过研究广域虚拟数据空间系统中的缓存技术，提出边缘缓存机制优化数据访问通路，将数据以文件粒度缓存在靠近边缘客户端的位置，从而提升上层应用访问和共享数据的性能。测试结果表明，作为虚拟数据空间系统的补充，提出的边缘缓存系统可提升广域数据共享的性能。

关键词
广域网；边缘缓存；广域虚拟数据空间；共享数据访问性能

中图分类号：TP319 文献标识码：A doi: 10.11959/j.issn.2096-0271.2021049

Research and implementation of edge cache system in global virtual data space across WAN

HUO Jiantong[1,2], XIAO Limin[1,2], HUO Zhisheng[1,2], XU Yaowen[1,2]
1. State Key Laboratory of Software Development Environment, Beijing 100191, China
2. School of Computer Science and Engineering, Beihang University, Beijing 100191, China

Abstract
Aiming at the problem of a large amount of network bandwidth wasted by data redundancy transmission when the edge client accesses and shares remote data in the GVDS, the caching technology in the wide-area virtual data space system was studied and an edge caching mechanism to optimize the data access path was proposed. The data is cached at a file granularity close to the edge client, thereby improving the performance of upper-level applications to access and share data. The experimental results show that the edge cache system proposed can improve the performance of wide-area data sharing as a supplement to the GVDS.

Key words
wide area network, edge cache, global virtual data space, shared data access performance

1 引言

当前中国国家高性能计算环境中总计算能力突破200 PFlops，总存储容量超过160 PB，拥有2个南北主节点、6个国家级节点、11个普通节点。但各个节点广域分散，计算与存储资源难以统筹使用。当前计算资源已经基本做到全局调度，但存储资源仍处于广域分散、隔离自治的状态，未能实现统一管理和共享访问。随着计算规模和数据量的快速增长，为了满足大型高性能计算应用跨域统一访问、广域数据共享、存储与计算协同的需求，基于国家重点研发计划“高性能计算虚拟数据空间”项目，笔者设计并实现了广域虚拟数据空间系统（global virtual data system，GVDS）[1]，并在5个国家超级计算（以下简称超算）中心进行了部署和验证，**图1**为GVDS的部署情况。

如**图1**所示，虚拟数据空间客户端位于

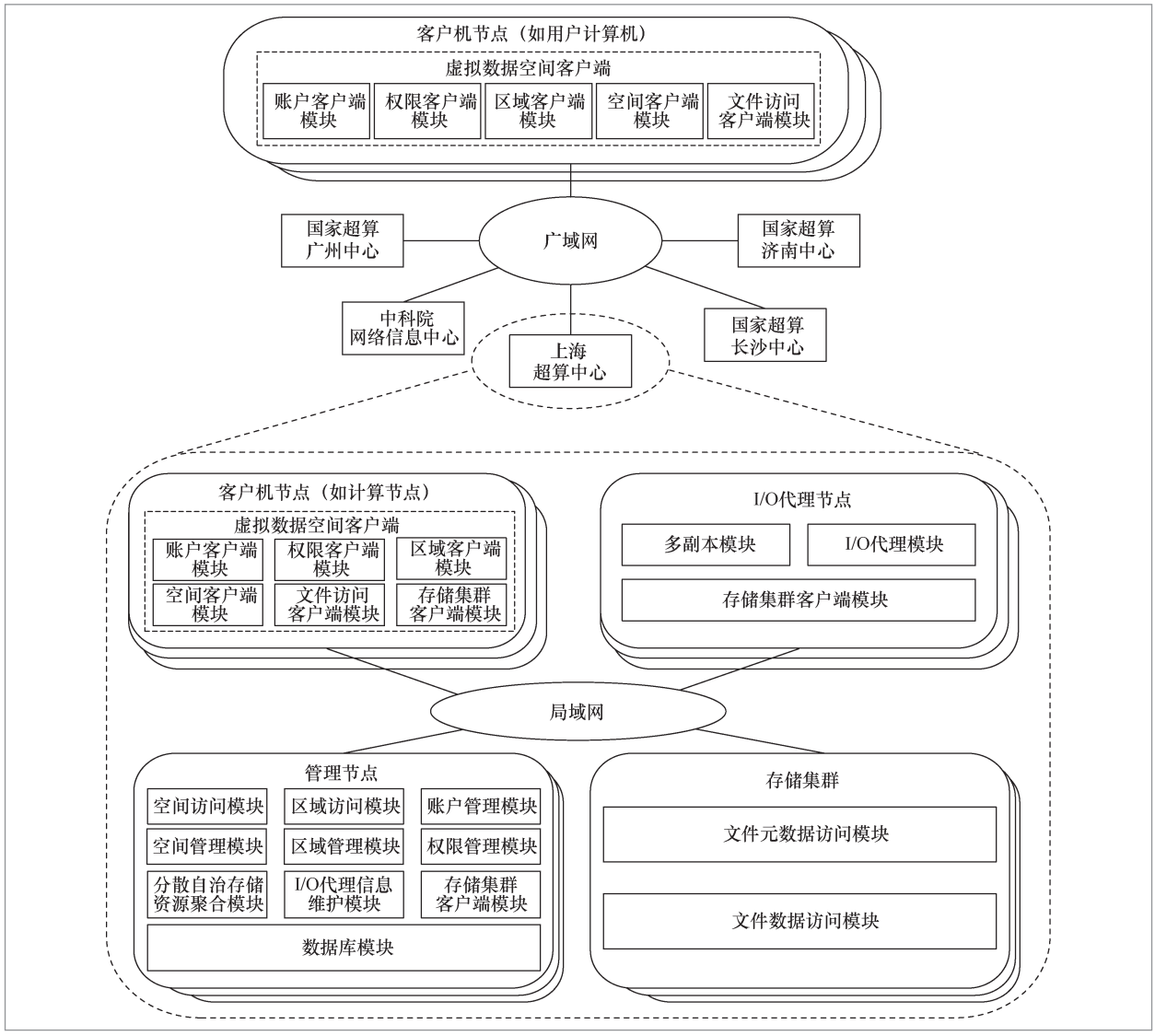

图1 GVDS的部署情况

网络边缘，可称其为边缘客户端。边缘客户端一般部署于PC上，是用户访问广域虚拟数据空间系统的入口；伴随着网络技术的发展，其还可部署在边缘侧的计算节点、数据采集节点和移动设备上。用户可通过客户端直接访问广域虚拟数据空间系统，进行跨超算中心的数据管理、共享和访问。然而，同一研究机构内的用户在一段时间内对相关的数据进行共享和访问时，广域网环境中会多次传输冗余数据，当数据量较大且访问量过多时，会造成网络带宽资源的浪费。另外，随着万物互联时代[2]的到来，边缘计算[3]迅速发展，其将计算作业驻留在靠近数据源、靠近用户的计算设备上运行，可减少数据传输量，从而缩短数据传输时延，最终提高云计算中心的可用性和处理作业的能力。

接下来，通过一个实例阐述当前广域虚拟数据空间系统在广域环境中的冗余数据传输问题。在气象预测场景中，气象数据大多来自网络边缘的采集设备，经过多种处理工序，最终上传到超算中心进行分析，整个过程中会将产生的数据进行多次共享和复制。尽管广域虚拟数据空间系统可形成统一的存储视图，方便了数据管理和数据访问，但当同一区域的多个边缘客户端共享和访问数据时，广域网下仍存在大量的冗余数据复制。以**图2**所示的天气预测过程为例，气象数据来源于全国各地的气象监测点，数据主要包括气温、气压、降水、风向、风速、湿度和辐射等，数据通过气象监测站点的边缘设备进行同化处理后，汇总至省市级气象站，之后进一步汇总至区域级和国家级气象信息中心进行气象预测；各级气象预测单位之间通过国家气象计算网格[4-5]和中国国家网格的网络进行数据传输。广域虚拟数据空间系统对全局气象数据实现了广域网环境的统一管理和调度，然而，在广州气象局和福建气象局同时需要山东气象局的数据时，两者会同时从广域网环境中访问并获取数据，从而造成冗余数据传输。假设在广东气象局设

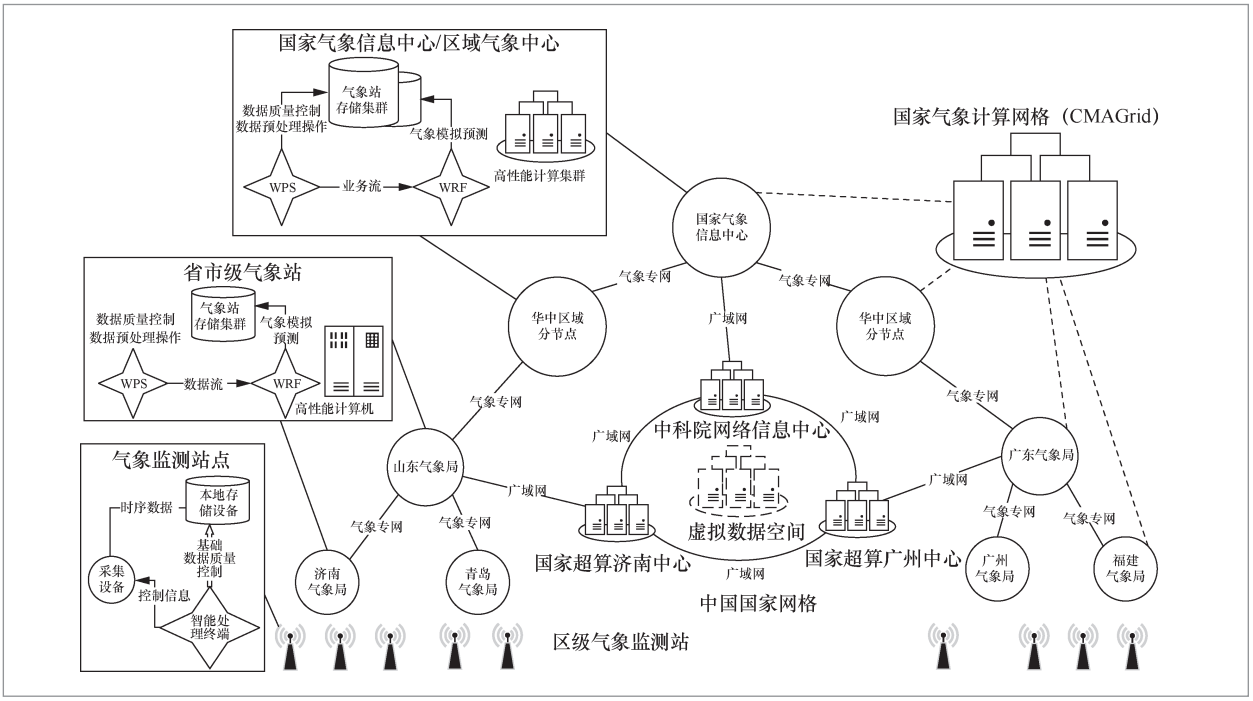

图 2 气象数据处理流程

置边缘存储或者边缘缓存，将气象数据进行缓存，广州气象局和福建气象局从广东气象局获取数据将极大地减少冗余数据传输。另一种情况，当青岛气象局需要济南气象局的数据时，如果在山东气象局设置边缘缓存，济南的气象数据不用到达虚拟数据空间核心空间，即可通过边缘缓存将数据传递给青岛气象局，从而提高广域虚拟数据空间系统的数据共享效率。

通过上述示例可知，尽管国内外仅有的几个跨广域网环境的存储系统已经实现了跨广域环境的存储资源的统一管理，如GVDS、欧洲网格基础设施（EGI）[6]，但仍面临如下数据共享性能的新挑战：缺乏边缘缓存系统，导致数据在广域网环境中进行冗余传输，广域环境存在很高的传输时延，无法充分发挥广域虚拟数据空间系统的存储资源聚合效应的优势，最终会降低高性能计算应用的性能。

因此，靠近客户端的边缘缓存研究是跨广域存储系统研究领域的热点之一，也是亟须解决的问题之一。基于GVDS现有架构构建边缘缓存系统具有十分重要的意义，可提高广域虚拟数据空间数据访问和数据共享的效率，对于促进高性能计算有重要的推动作用。为了解决上述示例中边缘客户端在访问与共享数据时存在的问题，本文在GVDS中设计并实现了云边协同的边缘缓存系统，通过提升共享数据效率来提高边缘用户的整体数据访问性能，具体如下。

（1）设计并实现了边缘缓存架构及其关键技术

本文设计并实现了边缘缓存系统的架构，并将其作为广域虚拟数据空间系统的补充。该架构最大限度地利用了跨域虚拟数据空间中现有的数据访问机制，尽可能地减少了对广域虚拟数据空间基础软件系统的修改；另外，本文提出并实现了一系列边缘缓存的关键技术，具体包括缓存索引机制、缓存替换策略、边缘缓存集群方案设计、缓存数据一致性策略。

（2）在广域虚拟数据空间系统中实现了边缘缓存系统

基于上述边缘缓存的架构和关键技术，在广域虚拟数据空间系统中实现了边缘缓存系统。边缘缓存系统由边缘客户端模块和边缘缓存服务节点组成，其中边缘缓存服务节点包括缓存接口层、缓存组织层和数据服务层。

2 相关研究工作

边缘缓存的起源可以追溯到20世纪90年代，Akamai公司提出了内容分发网络（content delivery network）[7]的概念。内容分发网络是一种基于Internet的缓存网络，其依靠地域分散的内容缓存服务器，将需要分发的文件在广域网上放置多个副本，并通过中心平台的调度和负载均衡策略，将用户的访问发送到距离较近的内容服务器上，从而缓解网络拥塞，提高广域网数据访问的速度。

2006年亚马逊提出了弹性计算云（elastic compute cloud）的概念，在计算、存储和可视化等方面开启了许多新的机遇。Ramaswamy L等人[8]提出了合作式边缘缓存网格，该网格由多个分布式的边缘缓存云组成，如果缓存未命中，边缘缓存节点能够从临近的边缘缓存云中获取文件数据，以缩短用户等待时间。2009年，卡内基梅隆大学的Satyanarayanan M等人[9]引入了微云（Cloudlet）作为边缘计算的形式，Cloudlet部署在网络边缘，并与互联网连接，是一个可信且资源丰富的主机，可以被移动设备访问及为其提供服务。

2014年，Storj Labs提出了一种去中心化的云存储平台STORJ[10]，该平台使用

P2P网络连接存储设备，并借助以太坊区块链技术激励用户将闲置的存储资源充分利用起来，从而以非常低廉的维护和管理成本为边缘端提供存储服务。2017年，Chen B Q等人[11]提出了一种基于D2D（device to device）网络的边缘缓存模型，其按照集群的方式划分边缘用户，将热文件缓存在各个集群中，从而可将D2D缓存网络的吞吐量提高4倍。2019年，Tan H S等人[12]研究了多个边缘服务器的在线协作缓存机制，当本地边缘服务器没有所需数据/服务时，可以选择与周边的边缘服务器关联合作（代价小），或者将服务请求直接发送到云端数据中心（代价大），或者下载云端数据/服务安装到本地，并在必要时替换本地已有内容（代价大），其分别设计了具有确定性和随机性的在线协作机制，优化了服务所需代价，给出了性能的理论保证（具有渐进最优的竞争比），通过真实/基准数据集模拟，验证了该机制的有效性。

综上所述，现有边缘缓存面临如下问题。

- 现有边缘缓存研究都是针对互联网环境中的数据共享应用的，没有考虑国家网格中高性能计算应用负载的需求，如跨多个超算中心的存算协同等要求，从而无法满足国家各超算中心跨广域的数据共享需求。因此，不能有效地满足广域虚拟数据空间系统的性能需求。
- 现在国内外没有开源的边缘缓存系统，现有的系统多为商业系统且闭源。通过研究自主可控的边缘缓存系统并将其开源，将弥补国家网格中高性能计算环境的空白，并进一步提升广域虚拟数据空间系统的性能，有效地提高大型高性能计算应用的性能。

3 边缘缓存架构和关键技术

3.1 广域虚拟数据空间边缘缓存架构

作为广域虚拟数据空间系统的补充，边缘缓存系统主要包括边缘客户端和边缘缓存服务节点，具体架构如图3所示。边缘

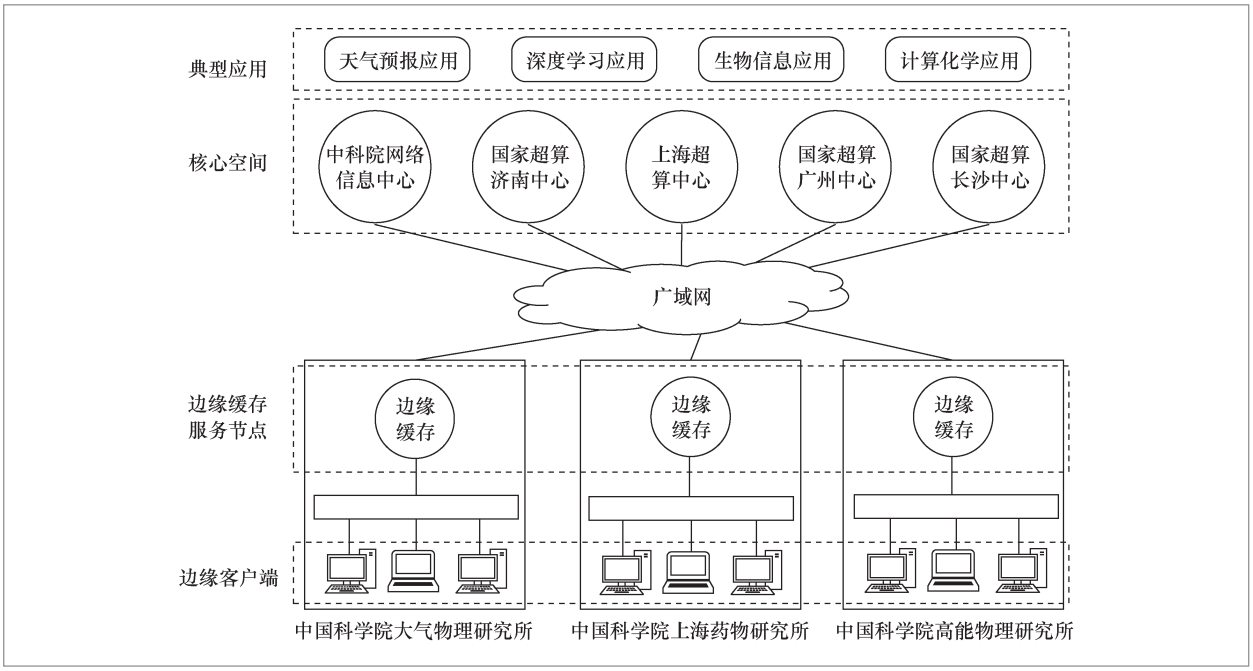

图 3 边缘缓存系统的架构

客户端处于网络边缘，可感知边缘缓存系统的存在；边缘缓存服务节点是边缘缓存的主要组成部分，多个服务节点可以组成边缘缓存集群，并为边缘客户端提供服务。边缘缓存服务节点提供边缘缓存系统的核心功能，包括缓存管理、缓存服务和缓存集群维护等。边缘缓存系统被设计为同构集群。

边缘缓存系统将广域虚拟数据空间系统中的数据以文件粒度进行缓存，整个边缘缓存系统以键值形式的扁平化数据结构进行索引。边缘缓存系统的工作流程如**图4**所示。

边缘客户端请求文件时，首先判断边缘缓存服务节点中是否完整地缓存了该文件及文件是否有效，如果有效，则响应边缘客户端的文件请求，并更新对应缓存文件的请求次数、最后请求时间等相关信息；如果边缘缓存服务节点未缓存该文件，边缘客户端主动向广域虚拟数据空间系统的核心空间请求获取数据，广域虚拟

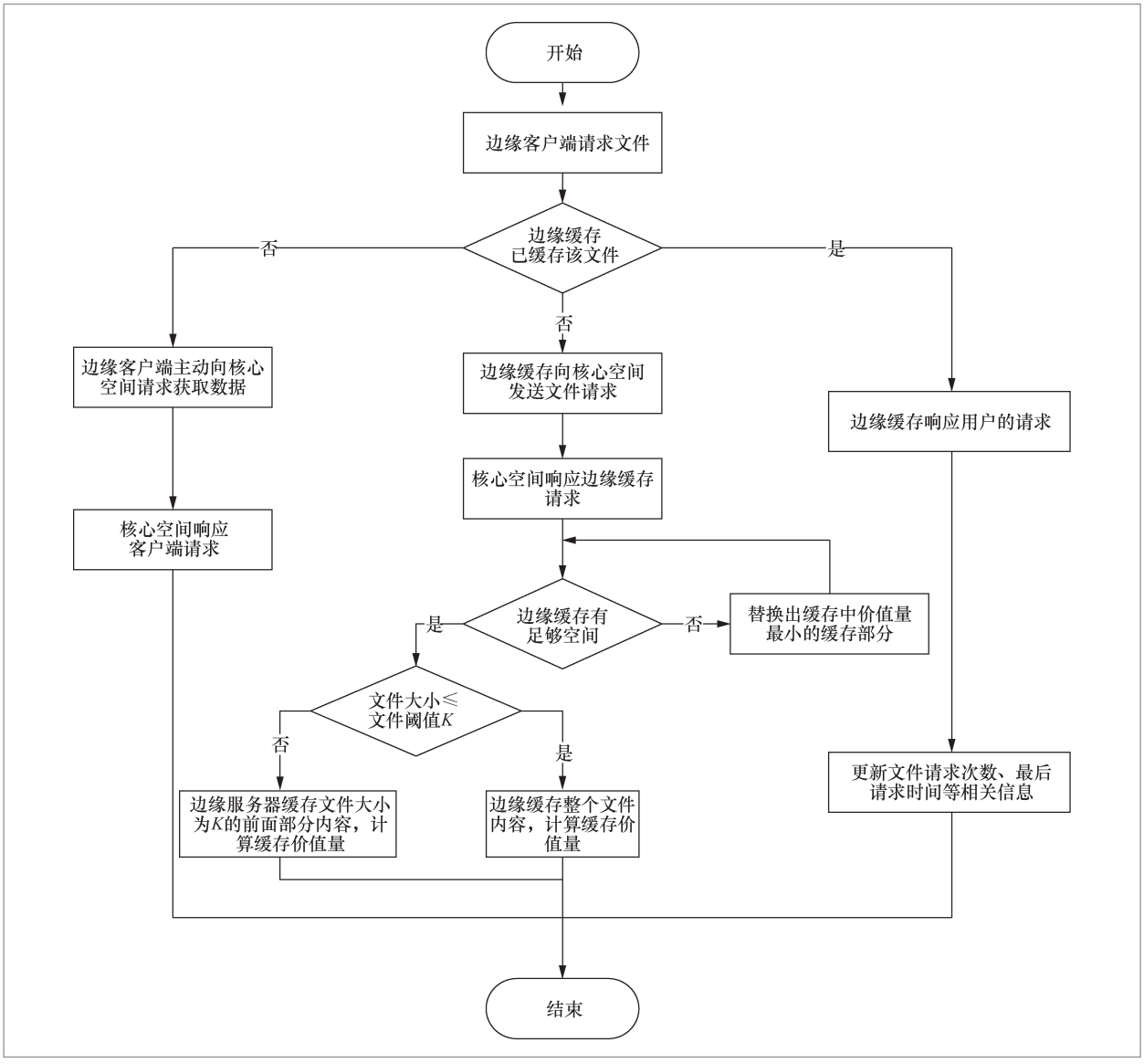

图 4 边缘缓存工作流程

数据空间系统响应边缘客户端的请求，完成整个数据请求流程；边缘客户端向边缘缓存服务节点请求文件，当边缘缓存服务节点未缓存该文件且该文件达到缓存条件时，边缘缓存系统将主动向广域虚拟数据空间系统发送文件请求进行预缓存，边缘缓存服务节点首先判断是否有能力存储该文件，如果无足够空间，则根据缓存替换策略，替换出缓存中价值量最小的文件，直到该文件能够被缓存。为了防止远程文件过大的现象出现，边缘缓存对文件大小设置阈值K（如1 GB），当预缓存文件小于文件阈值时，边缘缓存系统缓存整个文件，否则边缘缓存将缓存文件大小为K的前面部分内容，并计算缓存价值量。

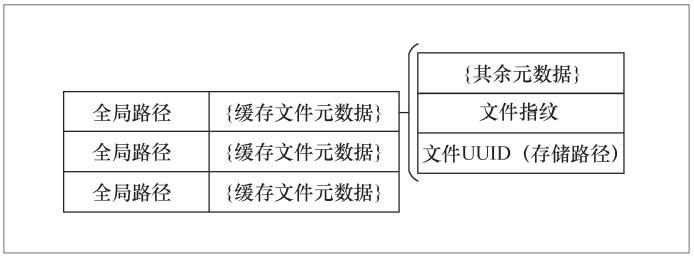

图5 第一层映射：文件全局路径到缓存文件元数据的映射

图6 第二层映射：文件指纹到文件 UUID 的映射

3.2 边缘缓存系统的关键技术

3.2.1 边缘缓存索引机制的设计

边缘缓存系统采用两层索引机制进行缓存文件的查找。整个边缘缓存系统的元数据持久化存储在分布式KV数据库系统中，索引机制具体细节如下。

图5所示为边缘缓存系统的第一层映射机制：文件全局路径到缓存文件元数据的映射。缓存文件元数据除了包含缓存文件的生存时间值（time to live，TTL）、访问频度、文件大小等，还包含文件指纹和文件UUID（universally unique identifier）信息。其中文件指纹和UUID可由算法生成，用于唯一表示数据。文件指纹主要用来向广域虚拟数据空间文件系统发送申请，以校验文件是否过期，文件UUID表示文件在分布式文件系统中的存储路径。

图6所示为边缘缓存中的第二层映射，该映射表示文件指纹到文件UUID的映射。UUID可以被看作磁盘上的一个文件。第二层映射主要用于数据去重，边缘缓存系统中的文件都是通过广域网从虚拟数据空间系统中获取的，当需要获取远程文件时，如果文件指纹与已经缓存的文件匹配，则在底层文件系统中创建软链接即可。

3.2.2 边缘缓存替换策略

缓存替换策略是提高缓存性能的关键技术之一，基于成本/价值模型的替换策略综合考虑了替换对象的大小、访问频率、访问时间、访问时间间隔和获取代价等因素。本文提出了一种基于贪婪对偶大小次数（greedydual size frequency，GDSF）优化策略的替换方法，并将其作为边缘缓存系统的替换策略，具体如下。

（1）初始化文件的价值参数$L=0$，边缘缓存空间的总大小为Total，已经使用的存储空间大小为Used，初始Used=0。

（2）如果请求的文件i在缓存中已经有副本，则Used值不变，文件访问频率Fr(i)加1，如式（1）所示：

$$\mathrm{Fr(i)}=\mathrm{Fr(i)}+1 \quad (1)$$

同时根据目标函数，重新计算文件i的价值$H(\mathrm{i})$。

（3）如果请求的文件i在缓存中不存在副本，那么边缘客户端将向广域虚拟数据空间系统获取文件数据。当文件请求表中对应的文件超过阈值时，边缘缓存系统获取文件的前K个部分（Size(i)），并进行缓存。文件访问次数Fr(i)=1，计算对象的价值，并将文件保存到缓存中，Used值会发生变换，如式（2）所示：

$$\text{Used}=\text{Used}+\text{Size}(\text{i}) \quad (2)$$

这时，根据边缘缓存系统的剩余空间是否足够，分为两种情况：

① 剩余空间足够，即Used≤Total，此时不需要进行缓存替换，把新文件i存放在边缘缓存中；

② 缓存空间不足，即Used>Total，此时必须进行对象替换，在缓存中选择k个具有非零最小代价H(i)的文件组$\text{i}_1,\text{i}_2,\cdots,\text{i}_k$，满足如式（3）、式（4）所示的2个条件：

$$\text{Used}-\sum_{j=1}^{K}\text{Size}(\text{i}_j)\leqslant\text{Total} \quad (3)$$

$$H(\text{i}_1)\leqslant H(\text{i}_2)\leqslant\cdots\leqslant H(\text{i}_k) \quad (4)$$

然后根据请求的文件i是否在替换文件中分成如下两种情况：

- 如果请求的对象i不在这k个文件$\text{i}_1,\text{i}_2,\cdots,\text{i}_k$中，那么$L$的值就是这$k$个文件中价值$H$最大的那个值，然后按照式（5）、式（6）计算得到$L$和Used的值；

$$L=\max_{j=1,\cdots,k}H(\text{i}_j)=H(\text{i}_k) \quad (5)$$

$$\text{Used}=\text{Used}-\sum_{j=1}^{k}\text{Size}(\text{i}_j) \quad (6)$$

- 如果请求的文件i在这k个文件中，表明新文件i的价值不足以被缓存，因此不需要替换任何对象，只要把Used值恢复到原来的大小即可，Used=Used-Size(i)。

（4）在清除文件时，要考虑其是否被其他软链接索引，再进行实际删除。

整个边缘缓存替换流程如**图7**所示。

3.3 边缘缓存集群方案的设计

随着单一科研机构中客户端数量的增多，单个边缘缓存服务节点将成为边缘缓存系统的瓶颈。本文采用集群的方式来提高边缘缓存系统的服务能力。本文提出的边缘缓存系统的集群方案设计主要包括同构集群结构设计[13]、基于一致性哈希算法的请求路由方法[14]、基于可伸缩可传导的弱一致性进程组成员资格（SWIM）协议[15]的集群成员维护方法。

3.3.1 同构集群结构设计

边缘缓存系统是广域虚拟数据空间系统的补充软件，为了降低系统的复杂性，本文将边缘缓存系统设计为同构集群，**图8**所示为边缘缓存系统的集群架构。如**图8**所示，边缘缓存系统由同构的边缘缓存服务节点构成，每个边缘缓存服务节点都具有缓存数据服务、缓存数据管理和集群状态维护等功能，最上层为部署在研究所内各类终端上的虚拟数据空间客户端，这些客户端在启动时可以选择是否使用边缘缓存系统。当不使用边缘缓存系统时，客户端可直接从广域虚拟数据空间系统进行数据访问，客户端后续不能感知到边缘缓存系统的存在；当选择使用边缘缓存系统时，客户端在访问文件时，首先查询想访问的文件是否在边缘缓存系统中，如果已缓存，则通过Proxy方式直接从边缘缓存系统获取文件；如果文件未在边缘缓存系统中缓存，则通过Bypass方式直接从广域虚拟数据空间系统访问数据。

3.3.2 基于一致性哈希算法的请求路由方法

当集群中存在多个同构的边缘缓存服务节点时，边缘客户端访问任意一个节点即可获取边缘缓存服务。尽管各个边缘

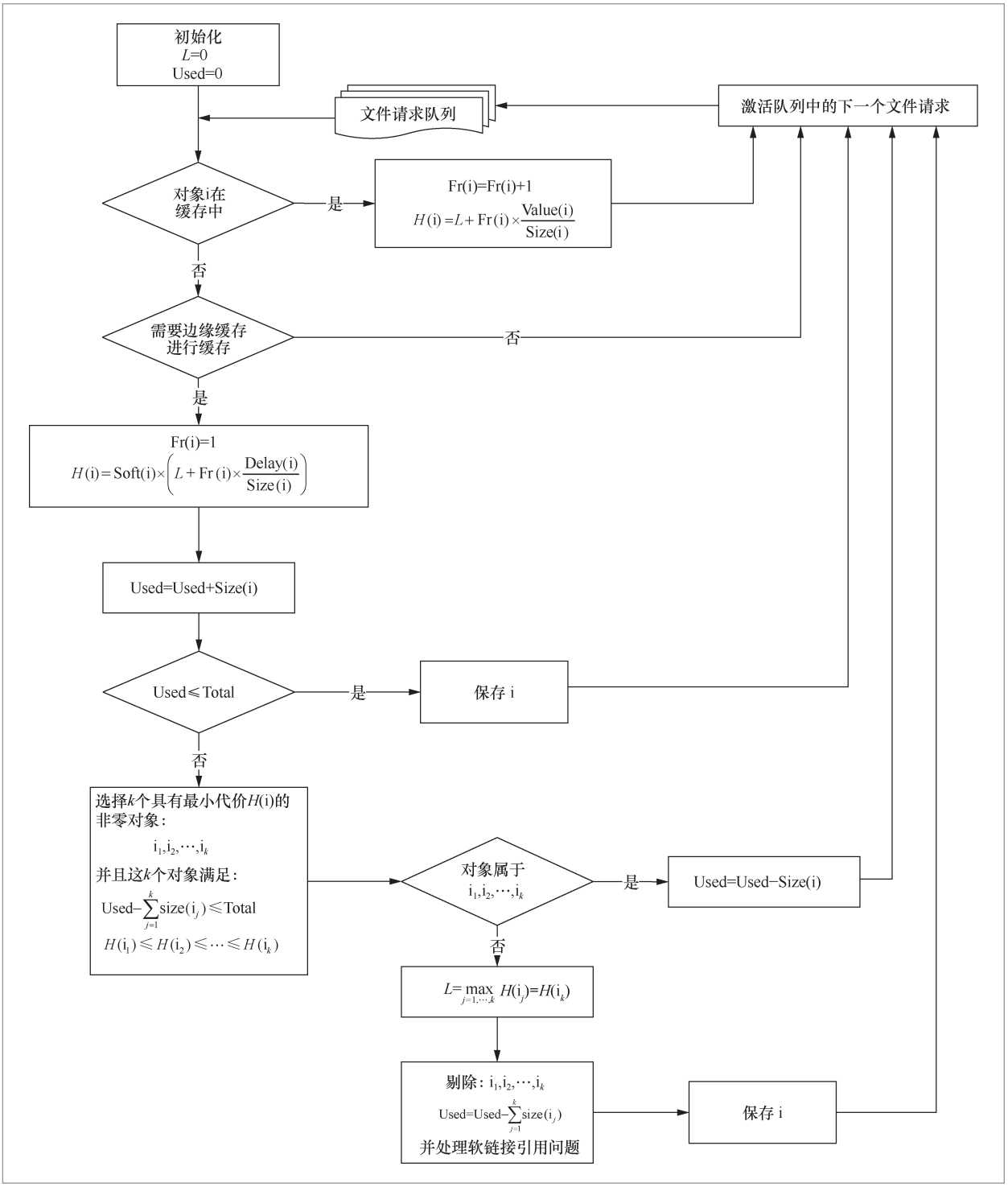

图 7 边缘缓存文件对象替换策略流程

缓存服务节点提供的功能相同，但由于节点间性能不同，向外提供的服务能力也不同。因此，需要研究相关负载均衡（load-balancing）方法把请求路由到合适的边缘缓存服务节点上。尽管一致性哈希算法可将边缘缓存服务节点加入和退出的影响降到最

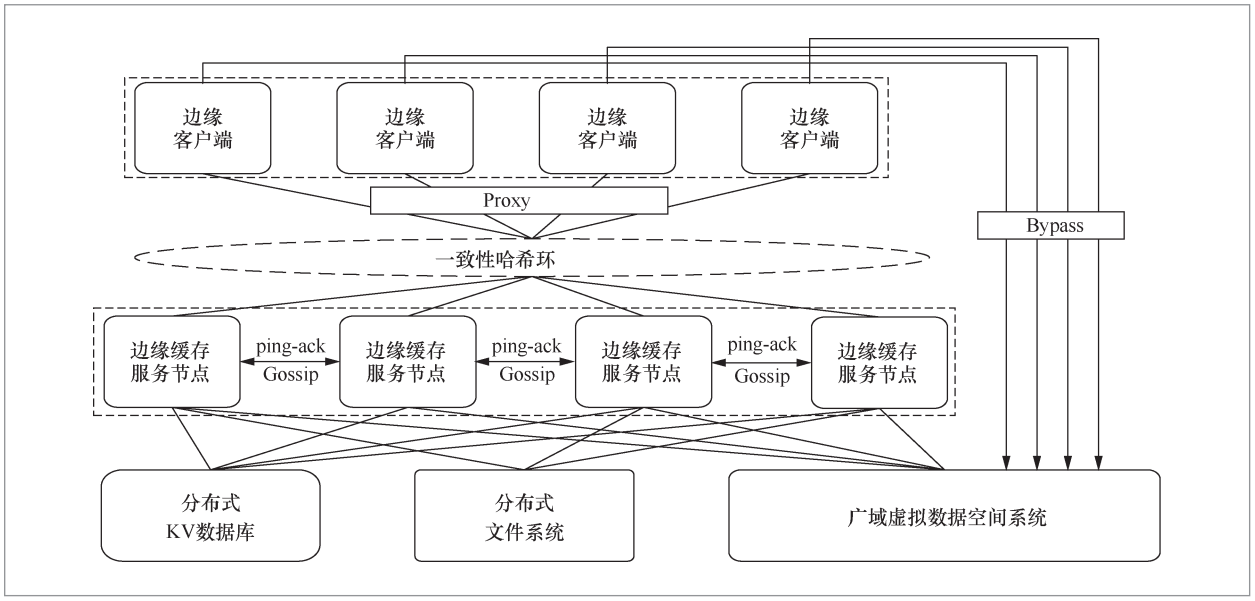

图 8 边缘缓存系统的集群架构

低，但在服务器数量较少时会导致映射不均匀，因此，本文提出了基于虚拟节点机制的一致性哈希策略的请求路由算法。

具体地，引入虚拟节点的机制，即根据每个边缘缓存服务节点自身的性能进行多次哈希计算，再映射到圆环上。如**图9**所示，以1个边缘缓存服务节点（**图9**中简称为缓存服务节点2）为例，分别增加1A、1B、1C、2B和2C虚拟节点，并将这些虚拟节点映射到哈希环上。通过这种方式，节点的分布更加均匀，文件访问仍然根据先前的方法进行路由，并增加虚拟节点到物理节点的映射，文件访问请求被路由到相应的边缘缓存服务节点中，可解决请求路由在边缘缓存服务节点中分布不均匀的问题。

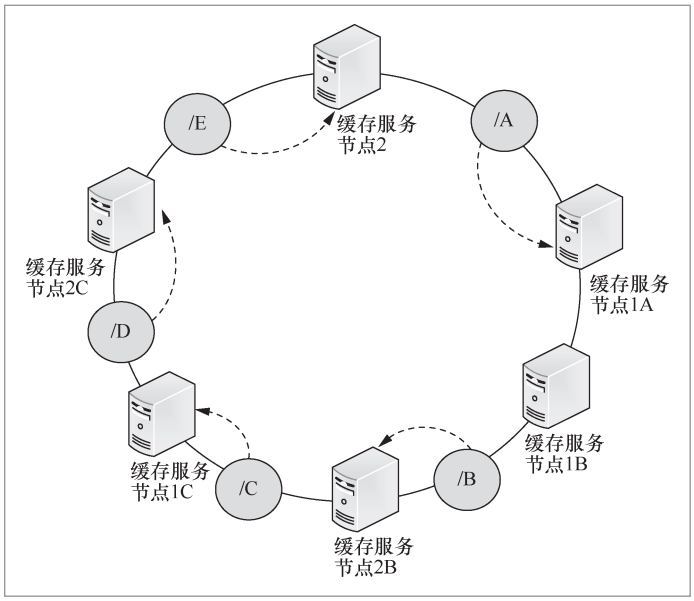

图 9 一致性哈希的虚拟节点

3.3.3 基于SWIM协议的集群成员维护方法

边缘缓存系统采用SWIM协议维护边缘缓存服务节点的状态信息。每个边缘缓存服务节点在内存中都维护一个状态表，该表记录了当前集群中活跃节点的状态信息，状态信息包括当前活跃节点的IP地址、开放的服务端口和节点的负载信息等。各节点周期性地从状态表中选择节点进行网络探包确认（ping-ack）操作。当有节点加入、离开或检测到节点失效时，通过Gossip协议[16]以点对点的方式向集

群中其他节点通知状态的变动。SWIM协议满足最终一致性，一定周期后集群状态将达到一致。

3.4 缓存数据一致性的设计

边缘缓存系统一致性主要包括边缘缓存系统内、边缘缓存系统与广域虚拟数据空间系统间两个层次的数据一致性。

3.4.1 边缘缓存系统内的数据一致性机制

边缘缓存系统内的数据一致性由锁机制来保证。锁模型中包含3种锁：共享锁（S，shared）、排他锁（X，excluded）和共享排他锁（SX，shared excluded）。S和X模式是经典的两种读写锁模式，SX模式是对X模式的优化，它与S模式是兼容的。**表1**描述了3种锁的兼容性关系。

表1 锁兼容关系表

类型	共享锁（S）	共享排他锁（SX）	排他锁（X）
共享锁（S）	兼容	兼容	冲突
共享排他锁（SX）	兼容	冲突	冲突
排他锁（X）	冲突	冲突	冲突

图10所示为在分布式锁和文件锁加入后边缘客户端EA访问缓存文件的过程。EA在访问边缘缓存文件F1时，首先访问边缘缓存服务节点（在**图10～图14**中简称为缓存服务节点）ES1，ES1获取F1的缓存元数据的共享排他锁，EA先通过ES1获取F1的文件共享排他锁，之后才能获取F1文件数据。

如**图11**所示，边缘客户端EA首先通过边缘缓存服务节点ES1获取F1元数据的共享排他锁，并继续后续访问。此时边缘客户端EB也通过边缘缓存服务节点ES2获取F1元数据的共享排他锁，因为锁的兼容性，此时只能获得共享锁。在ES1对F1元数据进行维护时，ES2只能对元数据进行查询操作。

图12所示为文件加锁过程，边缘客户端EA访问缓存数据F1，首先通过边缘缓存服务节点ES1获取F1元数据的共享排他锁，需要获取文件的共享排他锁才可修改F1。边缘客户端EB也访问F1时，只能获取F1的共享锁，并进行只读操作。

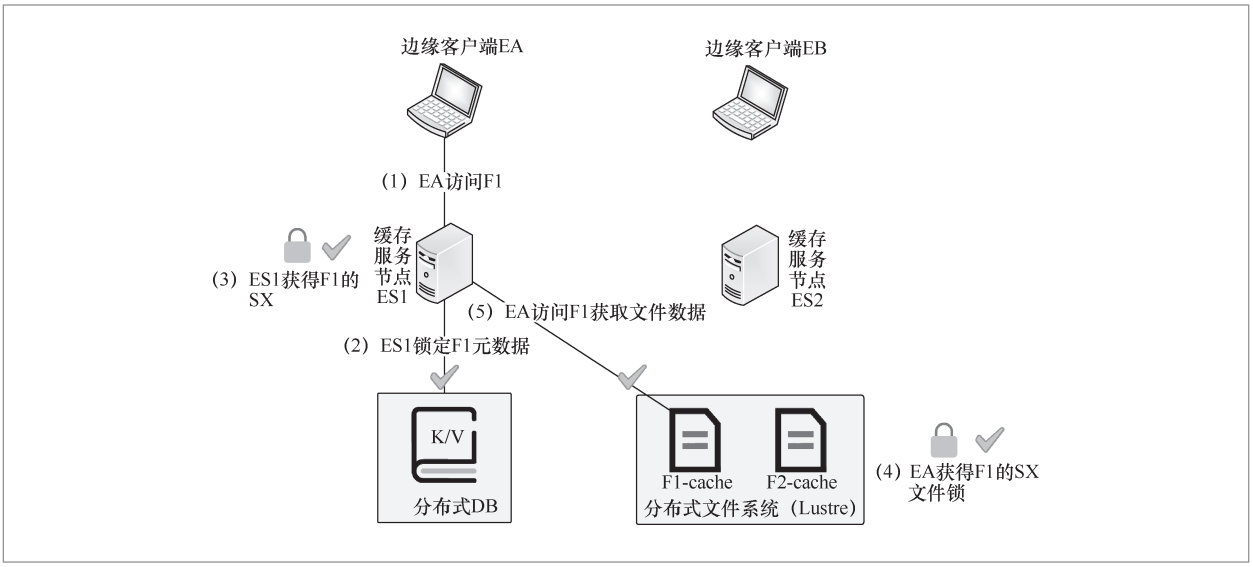

图 10 在一致性机制下 EA 访问边缘缓存文件的过程

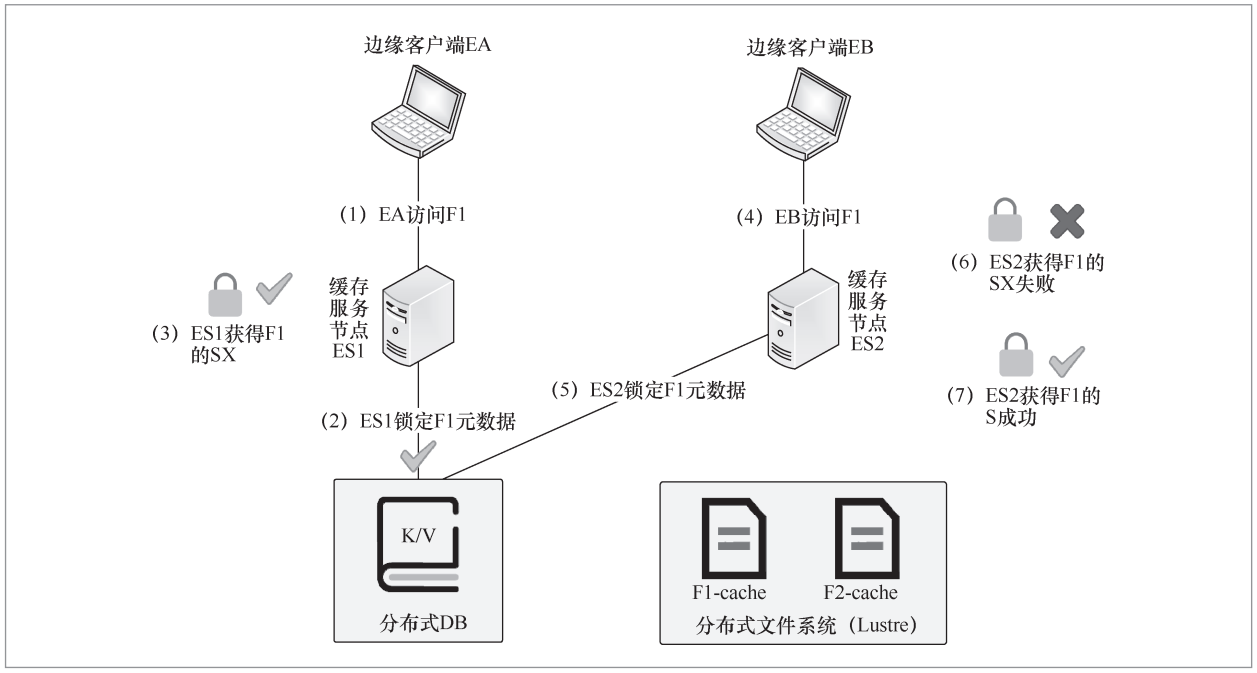

图 11 元数据加锁过程

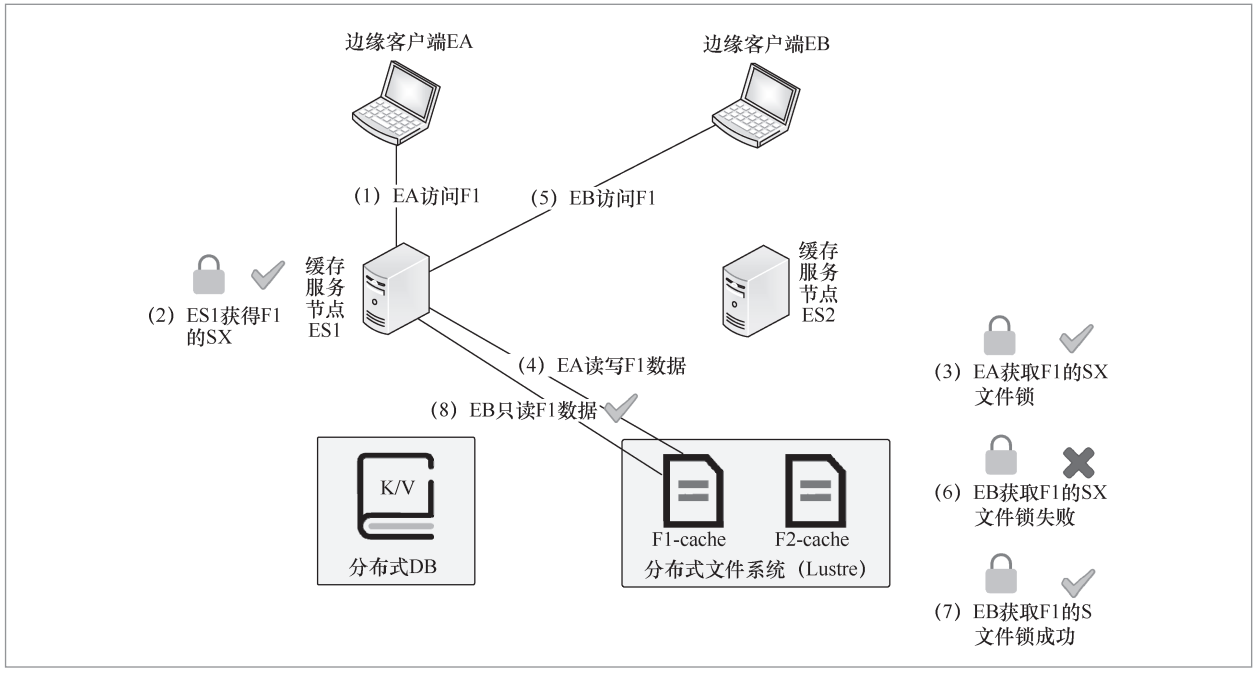

图 12 文件锁冲突过程

图13所示为缓存维护文件数据时的加锁过程，边缘缓存服务节点ES1在获取F1的元数据排他锁和F1的文件排他锁后进行缓存数据的维护。此时，边缘缓存系统正在维护F1数据，其他边缘缓存服务节点访问F1的元数据时加锁都不能成功。

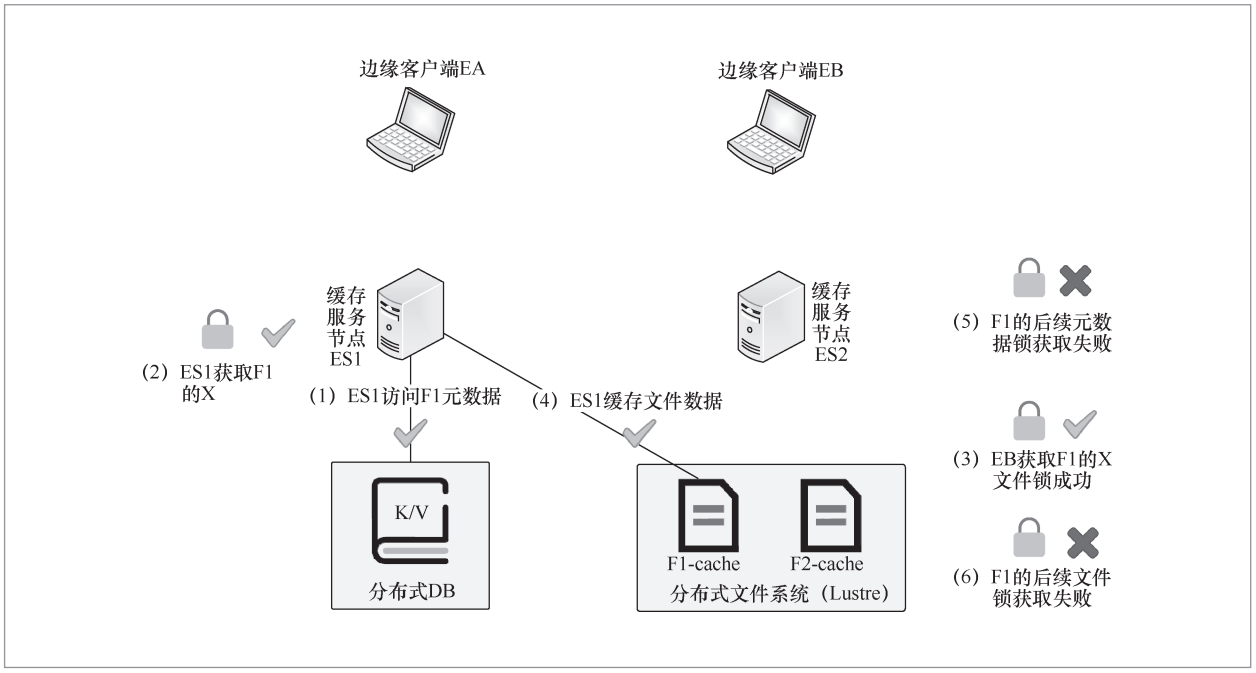

图 13 缓存服务节点维护文件数据时的加锁过程

3.4.2 边缘缓存系统与广域虚拟数据空间系统之间的一致性机制

边缘缓存系统采用类网络文件系统（network file system，NFS）的语义模型进行一致性控制，实现“打开-关闭”的一致性，并定期回写数据，以达到边缘缓存系统与广域虚拟数据空间系统的最终一致性；同时，边缘缓存系统会设置缓存文件的TTL和文件指纹。当文件被边缘客户端再次访问时，首先会检查文件指纹是否匹配，如果匹配，则通过边缘缓存系统访问文件数据，否则绕过边缘缓存系统直接访问广域虚拟数据空间系统。边缘缓存系统周期性地通过文件的TTL检查文件是否与广域虚拟数据空间系统一致，如果文件过期，则通过比较文件指纹来决定是否重新同步文件，并更新TTL。

边缘缓存系统会检查文件是否过期，如果未过期，则认为缓存文件可用；如果文件过期且文件指纹与远程文件指纹不同，则重新同步文件。

图14所示为同步过程，边缘缓存服务节点ES1首先获取F1的元数据排他锁，检查缓存的文件和F1文件指纹是否相同或检查数据是否达到同步周期。如果达到同步周期，ES1获取F1的文件排他锁进行数据同步；同时，当边缘客户端完成对缓存文件数据的读写时，将缓存的文件同步到广域虚拟数据空间系统中，并可设置写回间隔，以满足类NFS语义。

3.5 边缘缓存系统与广域虚拟数据空间系统接口设计

为了实现边缘缓存系统与广域虚拟数据空间系统之间的交互，需要设计合理的交互接口。本节将介绍边缘缓存系统中的缓存管理接口和数据访问接口，从而实现边缘缓存系统与广域虚拟数据空

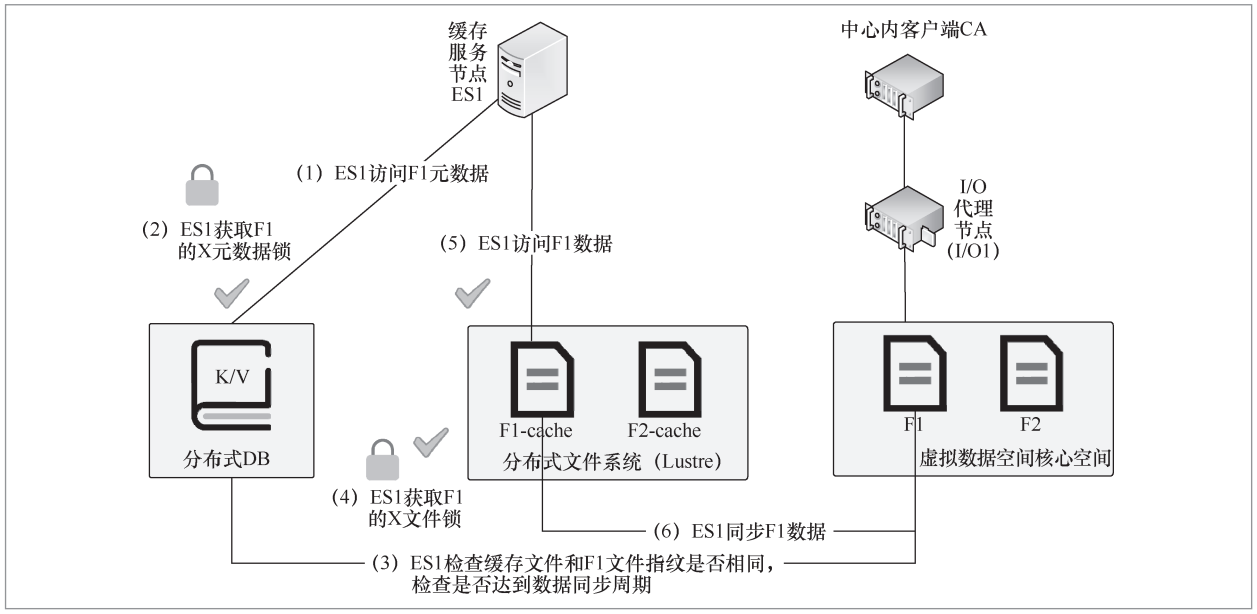

图 14 边缘缓存系统与广域虚拟数据空间系统的同步过程

间系统的对接。**表2**和**表3**描述了缓存管理RESTful接口和数据访问接口。

4 边缘缓存系统在广域虚拟数据空间系统中的实现

图15所示为边缘缓存系统在广域虚拟数据空间系统中的具体实现。边缘客户端和边缘缓存服务节点共同组成了边缘缓存系统。边缘缓存服务节点提供边缘缓存系统的核心功能，包括缓存管理、缓存服务和缓存集群维护等功能。边缘客户端包括虚拟数据空间客户端模块，使得虚拟数据空间客户端能够感知边缘缓存系统的存在。

图16所示为边缘缓存系统的组成。边缘缓存系统包含5层，其中，最上层为客

表 2 缓存管理 RESTful 接口

接口	接口规范	返回值	接口说明
缓存文件添加接口	POST /cache Json Value /r/n	Value 200 OK; 500 Internal Server Error	管理接口，表示向边缘缓存添加指定全局路径对应的文件
缓存文件查询接口	POST /cache Json Value /r/n	Value 200 OK; 500 Internal Server Error	管理接口，表示向边缘缓存查询指定全局路径对应的文件
缓存文件更新接口	POST /cache Json Value /r/n	Value 200 OK; 500 Internal Server Error	管理接口，表示同步边缘缓存中指定全局路径对应的文件
缓存文件删除接口	POST /cache Json Value /r/n	200 OK; 500 Internal Server Error	管理接口，表示删除边缘缓存中指定全局路径对应的文件
边缘缓存集群状态查询接口	GET /cluster	Value 200 OK; 500 Internal Server Error	管理接口，表示获取边缘缓存服务节点集群状态信息
边缘缓存容量状态查询接口	GET /storage	Value 200 OK; 500 Internal Server Error	查询边缘缓存、存储容量状态

表3 缓存管理数据访问接口

接口	接口规范	参数说明
缓存元数据查询接口	CacheMetaEdgeCacheSearch(string global_path, string ioproxy_info, string relative_path, string token)	global_path：文件全局路径； ioproxy_info：表示负责该文件的I/O代理信息； relative_path：I/O代理相对路径； token：客户凭证； return：缓存元数据信息；失败
缓存文件写入接口	Int64 EdgeCacheWrite(string global_path, string ioproxy_info, string relative_path, byte[] buf, int64 offset, int64 length)	global_path：文件全局路径； ioproxy_info：表示负责该文件的I/O代理信息； buf：文件数据存储缓冲； offset：文件写入起始位置； length：文件写入数据长度； return：写入的数据量值（写入成功），-1（写入失败）
缓存文件读取接口	Int64 EdgeCacheRead(string global_path, string ioproxy_info string relative_path, byte[] buf, int64 offset, int64 length)	global_path：文件全局路径； ioproxy_info：表示负责该文件的I/O代理信息； relative_path：I/O代理相对路径； buf：文件数据存储缓冲； offset：文件读取起始位置； length：文件读取数据长度； return：读取的数据量值（读取成功），-1（读取失败）
缓存文件关闭接口	Int64 EdgeCacheClose(string global_path, string ioproxy_info, string relative_path)	global_path：文件全局路径； ioproxy_info：表示负责该文件的I/O代理信息； relative_path：I/O代理相对路径； return：0（关闭成功），-1（关闭失败）

户端层，客户端层主要包括缓存管理命令行工具、虚拟数据空间边缘客户端；中间3层组成了边缘缓存服务节点，其包含缓存接口层、缓存组织层和数据服务层，缓存接口层包括缓存管理接口模块、缓存数据访问接口模块，缓存组织层包含集群维护模块、缓存管理模块、缓存服务模块，数据服务层包含KV数据库模块、本地I/O模块及远程I/O模块；最下面一层为资源层，包含分布式KV数据库、网络文件系统和虚拟数据空间核心空间，边缘客户端可以绕过中间3层，直接与处于资源层的广域虚拟数据空间系统进行通信，并获取数据。**图17**所示为边缘缓存系统中各模块间的具体交互。

5 实验验证

5.1 测试目标

测试主要包括功能测试和性能测试。功能测试包括对边缘客户端和边缘缓存服务节点功能的测试。在缓存服务节点侧，包括对缓存服务节点的集群功能和边缘缓存管理功能的测试。在边缘客户端侧，包括边缘客户端POSIX接口兼容和请求路由方法测试等，其中POSIX接口兼容测试主要包含文件元数据查询、修改，文件创建、写、读、复制及截断等，目录创建、查询、

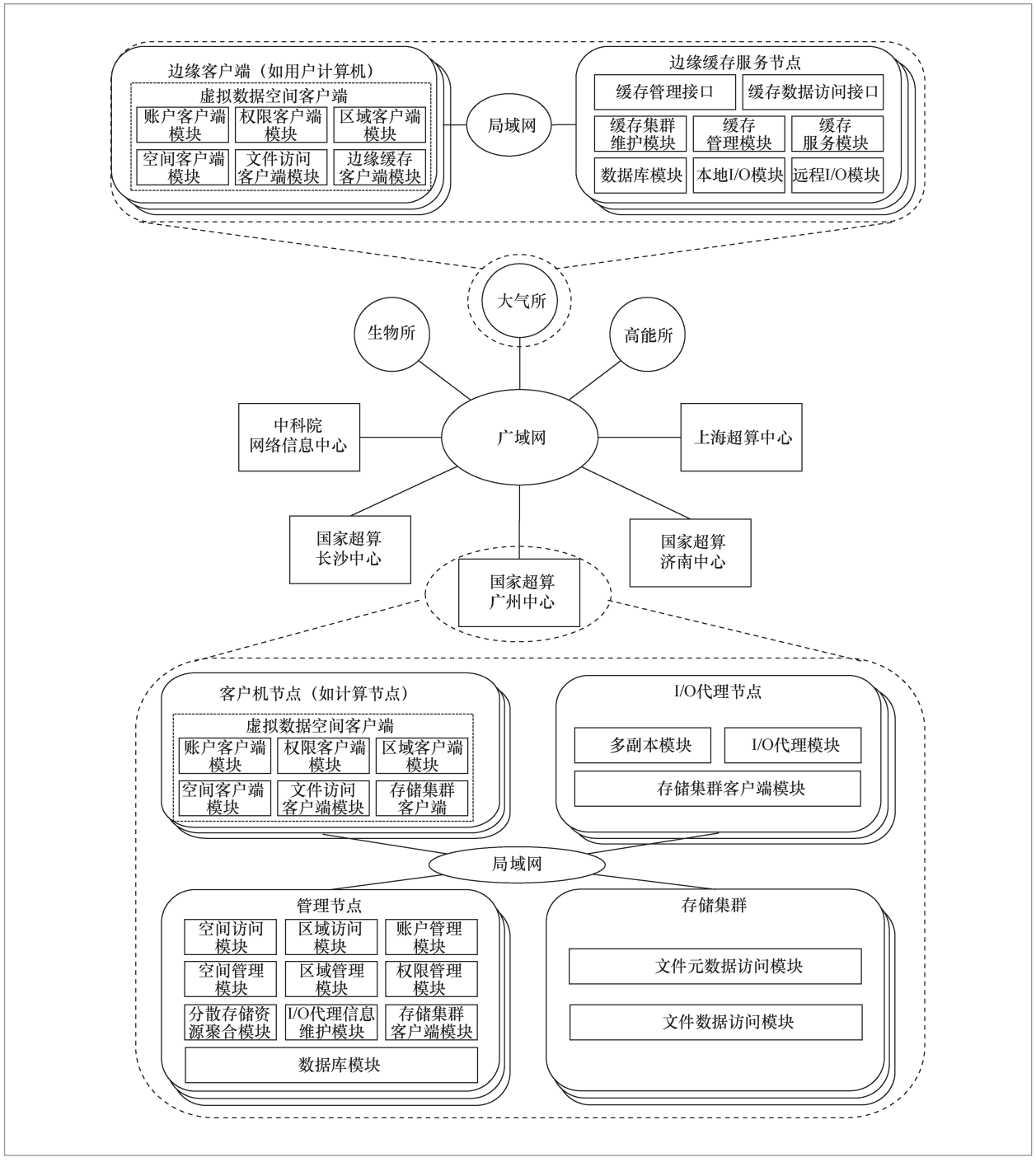

图 15 边缘缓存系统在广域虚拟数据空间中的实现

删除等，以及第三方应用运行的测试。关于性能测试，在广域网真实环境中，在有无边缘缓存的情况下，分别对边缘客户端的元数据与数据读写等操作的性能进行对比测试。

5.2 测试环境

在整个测试过程中，在PC上通过虚

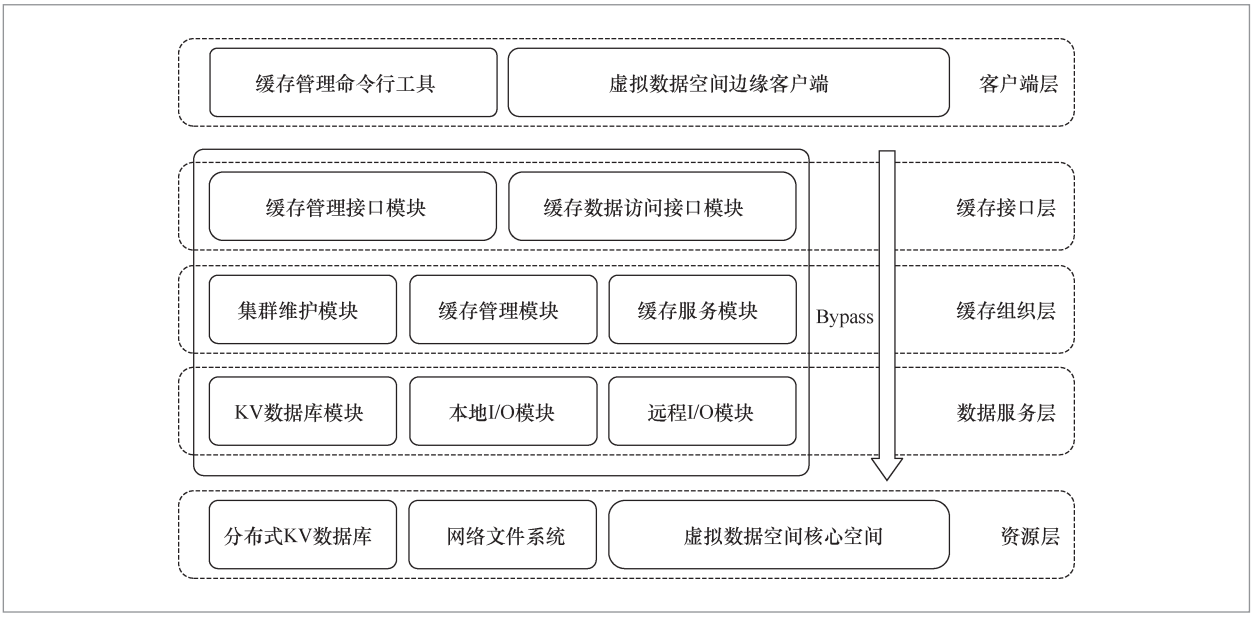

图 16 边缘缓存系统的模块组成

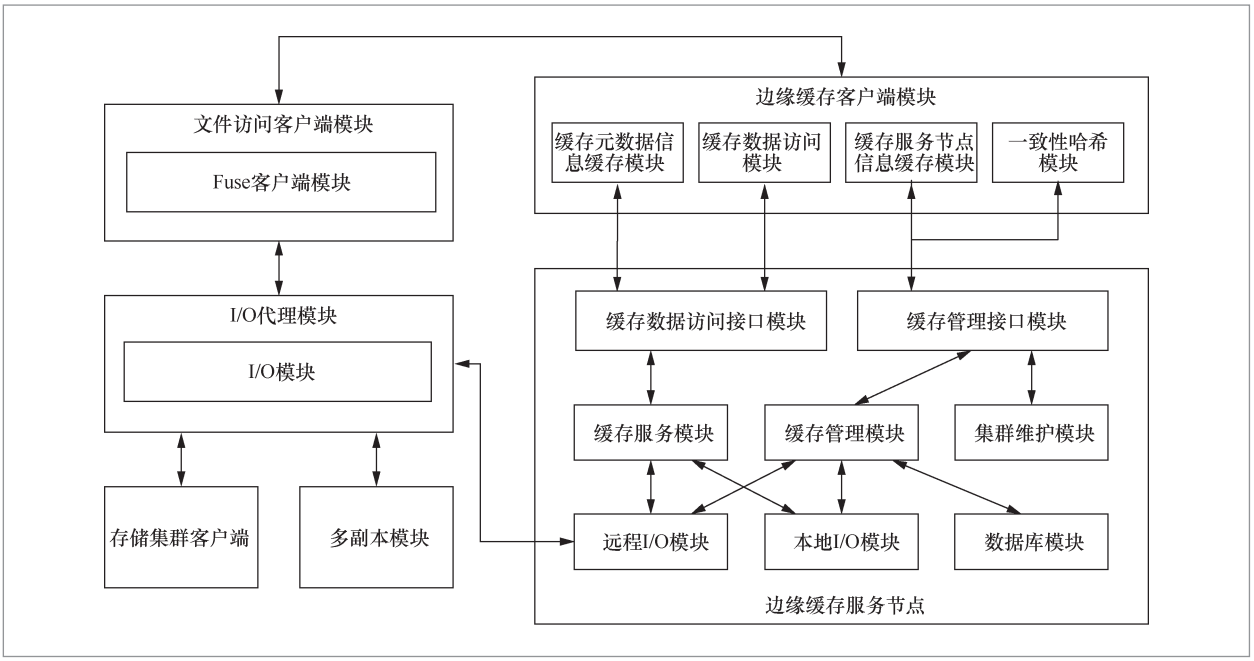

图 17 边缘缓存系统模块交互

拟机对边缘缓存系统的功能进行测试。笔者在真实广域网环境中对边缘缓存系统的元数据和数据访问的性能进行测试。功能测试的PC配置为1颗i7 9700 CPU，32 GB DDR4 3 200 MHz内存，Windows 10 Professional 10.0.19041 x86_64操作系统；局域网内边缘客户端配置为1颗i7 9700 CPU，32 GB DDR4 3 200 MHz

内存，Ubuntu 18.04.3操作系统；局域网内服务器配置为2颗Intel至强金牌5118 CPU，128 GB内存，Ubuntu 18.04.5操作系统；国家超算济南中心测试环境服务器配置为2颗Intel至强金牌51167 CPU，64 GB内存，Ubuntu 18.04.5操作系统。

为了保持软件环境一致，测试使用的主要软件、工具及其对应版本为：GCC 7.5.0、libfuse 3.8.0、Python 3.6.9、C++ 11、gRPC 1.27.2、Protobuf 3.1.3.0、Go 1.14.7、VirtualBox 6.1.16、WANem 3.0 Beta、fio 3.1和iperf 3.1.2。

中科院网络信息中心、国家超算济南中心、国家超算长沙中心、上海超算中心和国家超算广州中心5个节点上部署了广域虚拟数据空间系统。

5.3 功能测试

如**图18**所示，为了利用虚拟机对边缘缓存系统的功能进行测试，实验配置了虚拟机拓扑结构，所有虚拟机通过VirtualBox运行在PC上。边缘客户端由两台虚拟机模拟，边缘缓存服务节点由3台虚拟机模拟，边缘缓存服务节点还共同连接同一个分布式KV数据库和Lustre文件系统，Lustre文件系统也部署在独立的虚拟机上。虚拟数据空间系统由一台性能较高的虚拟机模拟，该虚拟机运行虚拟数据空间中的所有服务，功能测试对虚拟数据空间系统规模没有要求。在边缘缓存服务节点与虚拟数据空间之间设置广域网模拟器，边缘缓存服务节点所有请求要通过广域网模拟器才能到达虚拟数据空间。**图18**中的交换机由VirtualBox的Host-Only网络模拟，所有虚拟机配备由虚拟机模拟的千兆网卡，提供局域网千兆带宽的物理链路。

边缘客户端和边缘缓存服务节点处于同一网段中，虚拟数据空间系统在另一网段中；广域网模拟器配置两个网卡，分别处于两个网段中，并且通过命令“echo “1” > /proc/sys/net/ipv4/ip_forward”开启内核对IP数据包转发的功能，使得一个网卡接收的数据包可转发到另外一个网卡上；

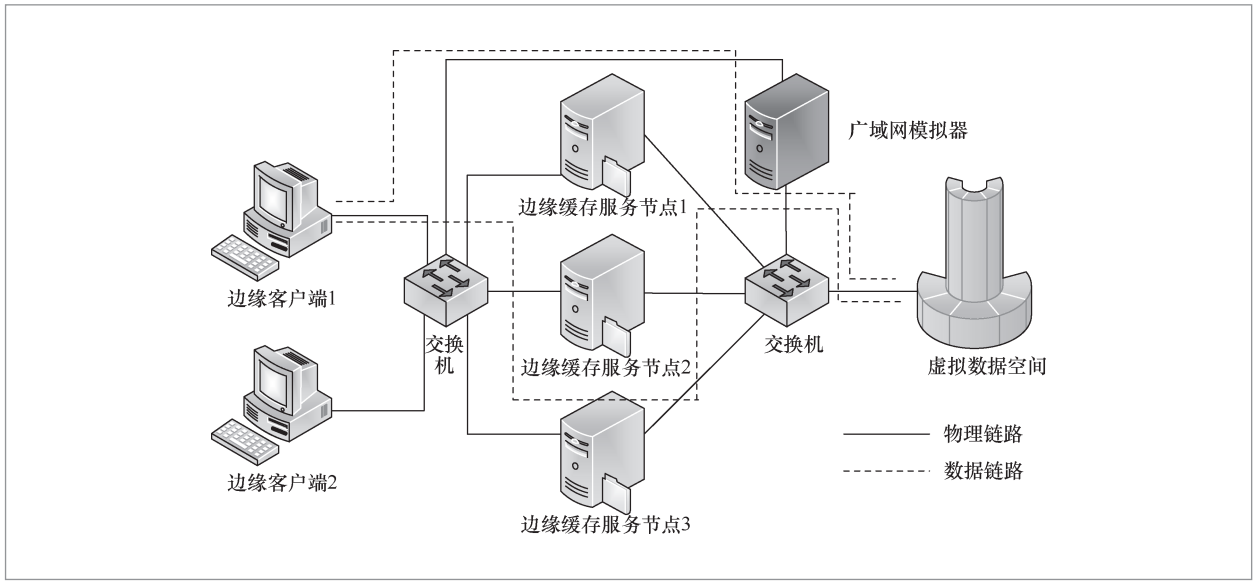

图 18 功能测试虚拟机拓扑

虚拟数据空间所在的虚拟机和边缘缓存系统所在的虚拟机都配置上通往对方网络的静态路由，网关为对应广域网模拟器的网卡IP地址，使得网络互联互通；在广域网模拟器上设置网络带宽和时延参数，完成实验环境的配置。

（1）边缘缓存服务节点功能测试

在IP地址为10.1.3.106的节点上部署并启动了虚拟数据空间系统。该部分主要进行边缘缓存文件管理接口功能测试。边缘缓存系统提供POST/cache类型的REST管理接口，包括文件查询接口、文件添加接口、文件同步接口和文件删除接口；同时，在边缘客户端实现了对应的管理命令行管理工具。边缘缓存系统命令行工具与ge-cluster命令一样，都要先将请求发送给边缘客户端，然后利用边缘客户端的全局信息将请求转发给边缘缓存服务节点，并由边缘缓存服务节点处理请求。

① 边缘缓存文件查询

边缘缓存文件查询由ge-stat命令实现。为了测试的顺利进行，提前在虚拟数据空间中存储了多种类型的数据，包括目录、文本数据、图片数据、音频数据、视频数据和二进制应用等。最初所有这些数据都存储在虚拟数据空间，在此条件下进行边缘缓存文件查询命令的测试。实验结果显示，边缘缓存系统的查询功能正常。

② 边缘缓存文件添加

边缘缓存文件添加由ge-add命令实现。ge-add通过边缘客户端将请求转发到边缘缓存服务节点，边缘服务器节点将文件同步到本地，初始化边缘缓存元数据信息，返回文件添加成功并提示用户。重新使用ge-stat命令查询该文件，返回结果显示该文件已被边缘缓存系统缓存。

③ 边缘缓存文件同步

边缘缓存文件同步由ge-sync命令实现，当虚拟数据空间中的文件被添加到边缘缓存系统时，边缘缓存系统将维护边缘缓存本地文件和远程文件的一致性。ge-sync命令使用户可以主动同步边缘缓存系统中存储的文件，这种操作有利于那些对文件时效性要求高的用户。测试证明，边缘缓存文件同步功能运行正常。

④ 边缘缓存文件删除

边缘缓存文件删除由ge-del命令实现。边缘客户端将ge-del请求转发到边缘缓存服务节点，边缘缓存服务节点再异步删除文件。边缘缓存文件的删除是指将边缘缓存系统中的文件删除。测试证明，边缘缓存文件删除功能运行正常。

（2）边缘缓存客户端功能测试

边缘客户端功能测试包括边缘客户端挂载、基于一致性哈希算法的请求路由方法、边缘客户端POSIX接口兼容和第三方应用运行。其中POSIX接口测试主要包含文件元数据查询、修改，文件创建、写、读、复制、截断、删除，目录创建、目录查询、目录删除等功能测试。

① 边缘客户端挂载测试

边缘客户端挂载测试主要用于展示边缘客户端是否能够真实挂载和成功运行。边缘客户端是基于用户空间文件系统（filesystem in userspace，FUSE）实现的，且为用户态文件系统。

通过mount命令输出的信息可观察到，client程序被挂载到/mnt/gvds节点上。因此，边缘客户端挂载功能运行正常。

② 请求路由方法测试

为了测试请求路由方法的可用性，启动边缘缓存服务节点1（IP地址为10.0.2.103）和边缘缓存服务节点3（IP地址为10.0.2.105），通过ge-cluster命令获取当前的集群状态。使用边缘缓存文件查询命令（ge-stat）测试路由机制的可用性。为了方便查看实验结果，边缘缓存服务节点每次接收到查询请求时都会输出客

户端查询日志，通过ge-stat命令行依次统计虚拟数据空间中的文件是否在边缘缓存中进行缓存。测试证明，文件请求路由方法功能运行正常。

③ 边缘客户端POSIX接口兼容测试

POSIX接口兼容测试主要包含：元数据查询、修改；目录创建、查询、删除；文件创建、写、读、复制、截断、删除。

通过ls命令查询文件的元数据信息，证明了文件元数据查询和修改功能可用；通过mkdir命令创建文件夹，查询其元数据信息，并删除目录，证明了边缘客户端目录创建、目录查询和目录删除功能可用。

文件操作验证实验证明，边缘客户端文件操作功能运行正常。

5.4 边缘缓存性能测试与分析

在真实的广域网环境中，在有无边缘缓存的情况下，对边缘客户端的读写数据与元数据的访问等方面进行性能对比测试。在中科院网络信息中心、国家超算济南中心、国家超算长沙中心、上海超算中心和国家超算广州中心5个节点上部署了测试环境。

笔者进行了3个性能对比实验：

- 虚拟数据空间系统在有无边缘缓存系统情况下的性能对比试验，对比实验从元数据访问性能、文件访问性能、文件并发访问性能等方面进行；
- 在局域网内搭建NFS，在相同局域网环境中，分析通过边缘缓存系统访问底层文件与通过NFS访问底层文件系统的性能差异；
- 为了测试边缘缓存系统本身带来的性能损失，将边缘客户端与边缘缓存服务节点部署在同一节点上，相比于直接访问底层存储，测试通过边缘缓存系统访问底层存储带来的性能损失。

图19所示为广域网环境的节点拓扑。边缘客户端的配置、边缘缓存服务节点配置以及国家超算济南中心的服务器配置与功能测试环境配置相同。交换机为千兆网交换机，可提供局域网千兆带宽的物理链路。

在广域网环境节点的网络配置中，边缘客户端和边缘缓存服务节点处于同一局域网中。**表4**描述了本地局域网与广域网的基本网络情况测试结果。性能测试采用了Linux命令集和fio等测试命令。

（1）在有无边缘缓存系统的情况下广域虚拟数据空间系统的性能

在客户端未使用边缘缓存模块的情况下，数据通过原始的数据通路访问广域虚拟数据空间系统，是广域虚拟数据空间原始系统通路，记为GVDS；在客户端使用边缘缓存模块的情况下，该客户端被称为边缘客户端，数据通过边缘缓存通路访问广

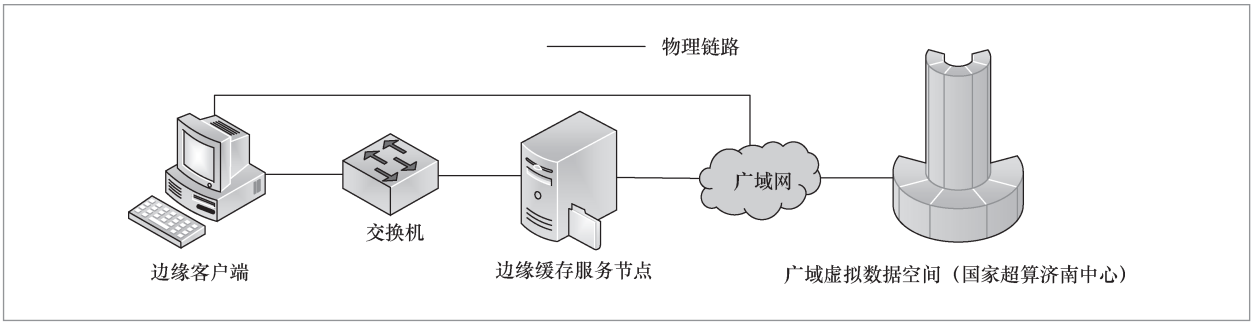

图 19 广域网真实环境下节点拓扑

域虚拟数据空间系统，是边缘缓存系统通路，记为GeCache。对比实验从元数据并发访问性能、数据访问性能和数据并发访问性能3个方面进行。

图20所示为GVDS与GeCache在多进程并发情况下元数据的访问性能结果。**图20(a)**显示了文件创建操作的性能对比结果，随着并发进程数的增加(从0增加到100)，GeCache的整体性能(最高为75 IOPS)高于GVDS(最高为55 IOPS)。**图20(b)**显示了文件查询操作的性能对比结果，并发进程数超过20时GVDS(并发进程数为50时吞吐量为7 600 IOPS)高于GeCache(并发进程数为50时吞吐量为5 500 IOPS)；随着并发进程数量的增长两者趋于相同(并发进程数为100时，GVDS吞吐量为7 800 IOPS，GeCache吞吐量为7 600 IOPS)。**图20(c)**显示了文件删除操作的性能测试结果，随着并发进程数量的增长(从0增加到100)，GeCache的性能(最高吞吐量为250 IOPS)一直高于GVDS(最高吞吐量为200 IOPS)。原因是在边缘缓存存在的情况下，文件是直接在边缘缓存中创建的，并异步在广域虚拟数据空间系统中同步；文件删除时，如果文件在边缘缓存中，则先从边缘缓存中删除，再异步从广域虚拟数据空间系统中删除，因此文件创建和删除时GeCache性能高于GVDS。文件状态查询时，因为边缘缓存不包含文件元数据，元数据需先从GVDS中获取，再从GeCache中获取被缓存的文件元数据，导致通信次数增加，所以GeCache元数据查询性能低于GVDS系统。后期可以采用元数据聚合和预取手段提高GeCache元数据访问性能。

表4 本地局域网与广域网网络基本情况测试结果

网络环境	上行带宽	下行带宽	时延	抖动	丢包率
回环网络	7.8 GB/s	7.9 MB/s	0.04 ms	0.01 ms	0
局域网	117.8 MB/s	117.8 MB/s	0.12 ms	0.02 ms	0
广域网	4.6 MB/s	7.3 MB/s	21.22 ms	2.29 ms	1%

图21所示为GVDS与GeCache的单进程文件读写性能测试结果，包括顺序写、顺序读、随机写和随机读4种。如**图21**所示，随着数据块的增大(从8 KB增加到4 096 KB)，GeCache的顺序读写和随机读写的性能远高于GVDS。例如，在块大小为4 096 KB时，GeCache的随机写的吞吐率为90 000 KB/s，而GVDS的随机写的吞吐率只有10 000 KB/s。接下来分析实验结果的原因，用户在边缘缓存系统中读写文件，在带宽较高的情况下，GeCache的性能远远高于GVDS，且随着数据块增大，吞吐率差异巨大，由此证明边缘缓存系统的有效性。另外，如**图21**所示，GVDS

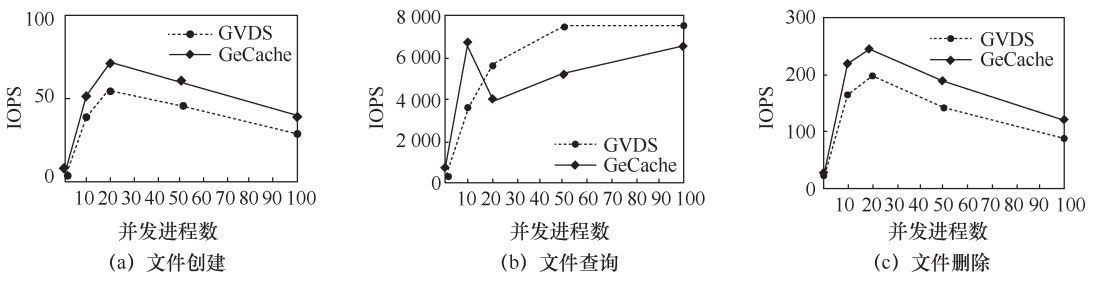

图20 在多进程并发情况下GVDS与GeCache的元数据访问性能

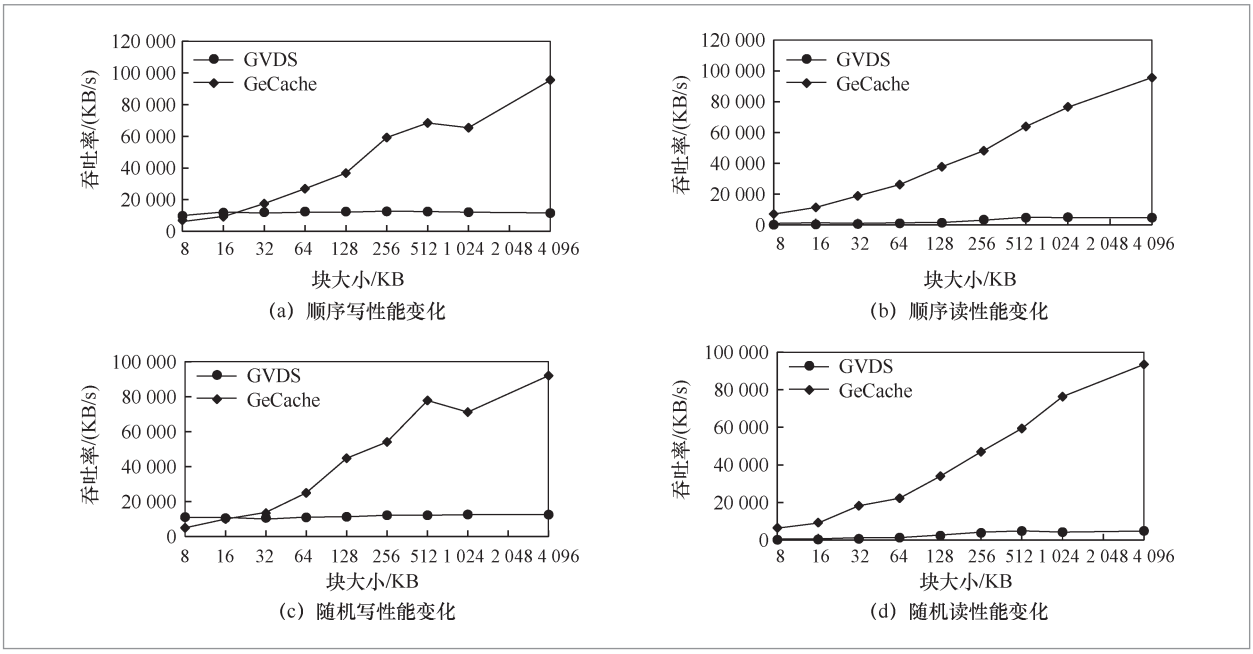

图 21 GVDS 与 GeCache 单进程情况下文件读写性能

在块大于256 KB后，系统吞吐率趋于稳定，这是因为此时已经达到广域网带宽，而GeCache系统因为局域网千兆网优势，吞吐率逐渐上升。

图22所示为GVDS与GeCache在多进程并发情况下的文件读写性能测试结果。如**图22**所示，GeCache的性能在4种情况下都远高于GVDS。例如，在多进程多文件写场

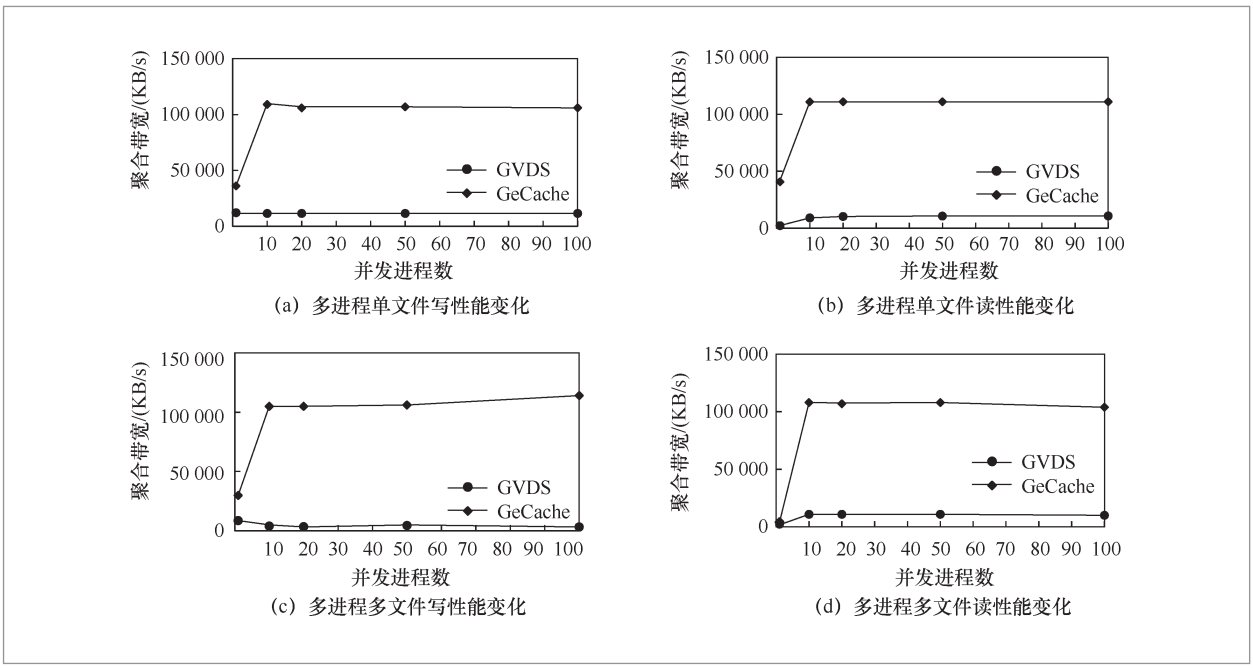

图 22 GVDS 与 GeCache 多进程并发情况下文件读写性能

景中，随着并发进程的数量从0增加到100，GeCache的聚合带宽可以达到110 000 KB/s，而GVDS只有将近100 KB/s。这是因为在多进程并发访问的情况下，GVDS和GeCache的聚合带宽大大提升；在并发进程数超过10之后，聚合带宽变化趋于稳定，这是因为GeCache达到了局域网千兆带宽的瓶颈，GVDS系统达到了广域网带宽的瓶颈。

（2）局域网环境下网络文件系统与边缘缓存系统的性能对比

本实验是在局域网环境下NFS与边缘缓存系统的性能对比，NFS服务器与边缘缓存服务节点运行在相同机器上，两者具有相同的底层文件系统/mnt/lustre；NFS客户端与边缘客户端位于相同机器上，这样可保证实验在相同的软硬件环境下进行。

图23所示是多进程并发情况下NFS与GeCache的文件读写性能测试结果。如**图23**所示，当进程并发量高于10时，NFS与GeCache系统聚合带宽基本相同，并接近局域网带宽。实验结果说明，GeCache不仅可以在广域网环境中提高虚拟数据空间系统的性能，还可以在局域网环境中提高虚拟数据空间系统的性能，并且带来的开销可以忽略不计。

6 结束语

本文针对广域网环境中边缘用户访问冗余数据降低应用性能的问题，通过研究虚拟数据空间系统中的缓存补充技术，优化了数据访问通路，在边缘客户端访问和共享远程数据时，避免了数据冗余传输造成的大量的网络带宽浪费，边缘缓存系统将

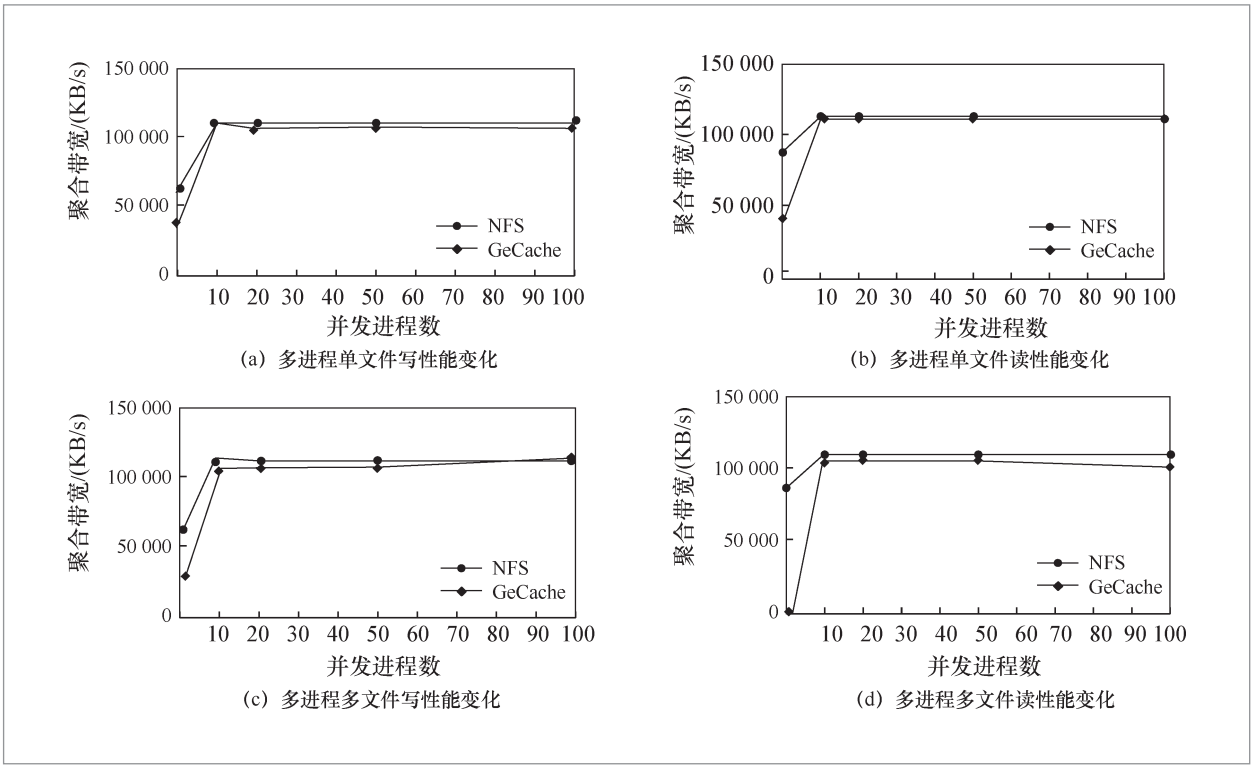

图 23 NFS 与 GeCache 多进程并发情况下文件读写性能

数据以文件粒度缓存在靠近边缘客户端的位置，可以提升上层应用访问和共享数据的性能。

参考文献:

[1] 肖利民, 宋尧, 秦广军, 等. GVDS: 面向广域高性能计算环境的虚拟数据空间[J]. 大数据, 2021, 7(2): 123-146.
XIAO L M, SONG Y, QIN G J, et al. GVDS: a global virtual data space for wide-area high-performance computing environments[J]. Big Data Research, 2021, 7(2): 123-146.

[2] ASHTON K. That "Internet of things" thing[J]. RFID Journal, 2009, 22(7): 97-114.

[3] 施巍松, 张星洲, 王一帆, 等. 边缘计算: 现状与展望[J]. 计算机研究与发展, 2019, 56(1): 69-89.
SHI W S, ZHANG X Z, WANG Y F, et al. Edge computing: state-of-the-art and future directions[J]. Journal of Computer Research and Development, 2019, 56(1): 69-89.

[4] 刘立明, 王彬. 气象网格环境下大数据的端到端传输机制研究[J]. 计算技术与自动化, 2014, 33(1): 122-126.
LIU L M, WANG B. Research of an end-to-end transfer mechanism for big data in CMAGrid environment[J]. Computing Technology and Automation, 2014, 33(1): 122-126.

[5] 王彬, 宗翔, 田浩. 国家气象计算网格的设计与建立[J]. 应用气象学报, 2010, 21(5): 632-640.
WANG B, ZONG X, TIAN H. Design and establishment of a nationwide meteorological computational grid[J]. Journal of Applied Meteorological Science, 2010, 21(5): 632-640.

[6] European open science cloud[J]. Nature Genetics, 2016, 48(8): 821.

[7] DILLEY J, MAGGS B, PARIKH J, et al. Globally distributed content delivery[J]. IEEE Internet Computing, 2002, 6(5): 50-58.

[8] RAMASWAMY L, LIU L, IYENGAR A. Scalable delivery of dynamic content using a cooperative edge cache grid[J]. IEEE Transactions on Knowledge and Data Engineering, 2007, 19(5): 614-630.

[9] SATYANARAYANAN M, BAHL P, CACERES R, et al. The case for VM-based cloudlets in mobile computing[J]. IEEE Pervasive Computing, 2009, 8(4): 14-23.

[10] WILKINSON S. STORJ a peer-to-peer cloud storage network[Z]. 2014.

[11] CHEN B Q, YANG C Y, WANG G. High-throughput opportunistic cooperative device-to-device communications with caching[J]. IEEE Transactions on Vehicular Technology, 2017, 66(8): 7527-7539.

[12] TAN H S, JIANG S H C, HAN Z H, et al. Camul: online caching on multiple caches with relaying and bypassing[C]// Proceedings of IEEE Conference on Computer Communications. Piscataway: IEEE Press, 2019: 244-252.

[13] CHERKASOVA L. Improving WWW proxies performance with greedy-dual-size-frequency caching policy[Z]. 1998.

[14] KARGER D, LEHMAN E, LEIGHTON T, et al. Consistent hashing and random trees: distributed caching protocols for relieving hot spots on the World Wide Web[C]//Proceedings of the 29th Annual ACM Symposium on Theory of Computing. New York: ACM Press, 1997: 654-663.

[15] DAS A, GUPTA I, MOTIVALA A. SWIM: scalable weakly-consistent infection-style process group membership protocol[C]//Proceedings of International Conference on Dependable Systems and Networks. Piscataway: IEEE Press, 2002: 303-312.

[16] BIRMAN K. The promise, and limitations, of gossip protocols[J]. ACM SIGOPS Operating Systems Review, 2007, 41(5): 8-13.

作者简介

霍建同（1985- ），男，北京航空航天大学计算机学院博士生，主要研究方向为高性能计算、分布式存储、系统安全、计算机网络等。

肖利民（1970- ），男，博士，北京航空航天大学计算机学院教授、博士生导师，计算机科学技术系主任，计算机系统结构研究所副所长，中国计算机学会（CCF）大数据专家委员会委员、高性能计算专业委员会常务委员、容错计算专业委员会委员，中国电子学会云计算专家委员会委员，主要研究方向为计算机体系结构、大数据存储、高性能计算等。曾获国家科技进步奖二等奖4项、省部级科技进步奖一等奖4项及其他省部级奖项5项。发表SCI/EI论文230多篇，申请发明专利100多项，其中授权发明专利88项。

霍志胜（1983- ），男，博士，北京航空航天大学计算机学院助理研究员，主要研究方向为大数据存储、分布式存储系统、分布式/并行文件系统等。作为项目主持人和项目骨干，主持和参与了多项博士后基金面上项目、国家重点研发计划项目、国家自然科学基金面上项目等。

徐耀文（1996- ），男，北京航空航天大学计算机学院硕士生，主要研究方向为分布式文件系统、高性能计算、网络安全等。

收稿日期： 2021-05-31

通信作者： 霍志胜，huozs@buaa.edu.cn

基金项目： 国家重点研发计划资助项目（No. 2018YFB0203901）

Foundation Item: The National Key Research and Development Program of China(No. 2018YFB0203901)