# GVDS：面向广域高性能计算环境的虚拟数据空间

肖利民[1,2]，宋尧[1,2]，秦广军[3]，周汉杰[1,2]，王超波[1,2]，韦冰[1,2]，魏巍[4]，霍志胜[1,2]
1. 北京航空航天大学计算机学院，北京 100191；2. 软件开发环境国家重点实验室，北京 100191；
3. 北京联合大学智慧城市学院，北京 100101；4. 西安理工大学计算机科学与工程学院，陕西 西安 710048

**摘要**
广域高性能计算环境是支撑科技创新和社会经济发展的核心信息基础设施。然而，在广域高性能计算环境中，异构存储资源在地理位置上的分散分布导致无法发挥广域存储资源的聚合效应，难以满足对广域分布数据的统一管理和高效访问需求。为此，提出了虚拟数据空间构建方法及数据访问性能优化方法，并实现了一个面向广域高性能计算环境的全局虚拟数据空间（GVDS）。GVDS可聚合广域分布的异构存储资源，形成统一的虚拟数据空间，有效支撑用户以统一访问模式高效访问广域分散的异构存储资源，实现广域环境中分布数据的跨域共享和协同处理。测试结果表明，与国际领先的面向广域高性能计算环境的OneData、GFFS等存储系统相比，GVDS实现了相当的功能，且数据访问性能明显提升。

**关键词**
全局虚拟数据空间；广域高性能计算环境；高效数据访问；异构存储资源

中图分类号：TP316　　文献标识码：A　　doi: 10.11959/j.issn.2096-0271.2021017

# *GVDS: a global virtual data space for wide-area high-performance computing environments*

XIAO Limin[1,2], SONG Yao[1,2], QIN Guangjun[3], ZHOU Hanjie[1,2], WANG Chaobo[1,2], WEI Bing[1,2], WEI Wei[4], HUO Zhisheng[1,2]
1. School of Computer Science and Engineering, Beihang University, Beijing 100191, China
2. State Key Laboratory of Software Development Environment, Beijing 100191, China
3. Smart City College, Beijing Union University, Beijing 100101, China
4. School of Computer Science and Engineering, Xi'an University of Technology, Xi'an 710048, China

***Abstract***
The wide-area high-performance computing environment is the core information infrastructure to support technology innovation, economic development, and national defense. However, heterogeneous storage resources are geographically distributed in wide-area high-performance computing environments, resulting in the barriers between applications and data. The requirements of unified data management and efficient data access cannot be met. A method of establishing virtual data space and a data access optimization method was presented, and a global virtual data space (GVDS) for

wide-area high-performance computing environments to satisfy the requirements was implemented. GVDS aggregates geographically distributed and heterogeneous storage resources, creating a unified virtual data space to provide unified and efficient data access. Sharing and collaborative processing of geographically distributed data were achieved in wide-area environments. The experimental results indicate that compared with the state-of-the-art wide-area storage system in the field of high-performance computing, such as OneData and GFFS, GVDS has similar functions and improves the read bandwidth significantly.

***Key words***
global virtual data space, wide-area high-performance computing environment, efficient data access, heterogeneous storage resource

## 1 引言

国家高性能计算环境拥有大量的计算与存储资源。根据中国国家网格（CNGrid）发布的《国家高性能计算环境发展报告（2002—2017年）》，截至2017年年底，我国国家网格节点单位达到19家，聚合计算资源超过200 PFlops，总存储资源超过178 PB。然而这些节点地理位置分散，如何发挥资源聚合效应并统筹使用这些广域资源一直是国家高性能计算环境发展面临的一个挑战性问题。网格技术[1]常被用于解决此类广域资源聚合问题。网格技术能够将网络上松散的计算与存储资源聚合起来，隐藏资源的异构性，并最终向用户呈现一个虚拟的超级计算机。当前CNGrid[2-3]中的计算资源已可做到全局调度使用，然而存储资源仍处于广域分散且隔离自治的状态，未能实现统一管理和共享访问。当研究人员需要跨机构协同工作以及分享大规模的数据文件时，通常需要通过网格的数据传输工具将与计算任务相关的输入文件传输到计算中心内部的存储集群中。在计算任务执行完成后，超算用户需要显式指定计算任务的输出文件来完成计算结果的回传。因此，现存的基础设施依然无法提供简单有效的数据访问与共享模式。

科学计算应用通常需要依赖外部数据文件，而其输入数据时常来源于多个超级计算中心（以下简称为超算中心）。以全基因组关联分析应用为例，其不同机构的基因库位于不同的超算中心，主应用位于某一个超算中心，当执行全基因组关联分析任务时，主应用需要访问多个超算中心的基因库，并划分出多个作业。每个作业都只需要访问基因库中所有文件的某一个文件片段，并进行关联分析。这种情况下，数据跨域分散且隔离自治未得到有效汇聚，难以实现对该应用数据的有效管理和共享，无法充分发挥多个超算中心并发处理海量数据的能力，导致各超算中心数据的重复存储以及较低的数据处理效率。此外，现有的跨域存储系统和访问方法不适用于广域高性能计算环境。因此，当前国家高性能计算环境亟须聚合广域分散的存储资源，以形成全局统一的数据空间，在此基础上，针对用户和应用对数据空间的多样化访问需求进行分区管理、共享和隔离，并在广域环境中提供数据空间的高效访问。可统一管理和高效访问的全局数据空间的形成主要面临如下挑战：

- 当前国家高性能计算环境存储资源广域分散且隔离自治，缺乏适用于广域高性能计算环境，能屏蔽底层文件系统异构性，并支持统一管理、分区共享及视图隔离的全局数据空间；

● 在带宽受限且高时延的广域环境下，难以实现高效的远程数据访问，难以满足广域分布数据访问需求。

针对上述挑战，本文提出了虚拟数据空间构建方法及数据访问性能优化方法，并实现了一个面向广域高性能计算环境的、访问模式统一且高效的全局虚拟数据空间（global virtual data space，GVDS）。GVDS是一个跨域分布式存储系统，单个GVDS实例可以直接部署到超算中心，并且通过与其他超算中心中的GVDS实例协作来实现存储资源的聚合。用户或计算节点通过运行在用户空间的GVDS实例，可以以POSIX文件访问接口的模式访问由跨域分散的异构存储资源聚合而成的数据空间。GVDS隐藏了数据传输过程，简化了多超算中心间数据访问与调度的模式。

本文的主要贡献包括以下两个方面。

● 设计了一个面向广域高性能计算环境的全局虚拟数据空间系统——GVDS。GVDS隐藏了底层存储访问接口、广域网链路、数据副本等的差异与复杂性，提供统一接口，以轻松管理、共享和访问跨域分散在多个超算中心的存储资源。

● 设计了高效的广域网远程数据访问机制，通过通信优化、数据预读与缓存、数据副本等技术有效提升了数据文件的实时访问速度。

## 2 相关工作

Lustre[4]、Ceph[5]和GPFS[6]等文件系统能够在本地数据中心内实现统一访问和数据共享，其向上隐藏了分布式存储系统数据调度的复杂性，使得数据访问和共享就像使用本地文件系统一样容易。但是在广域高性能计算环境中，此类问题变得更加复杂，系统的用户往往来自不同的地域和机构，因此需要整合跨域分散的异构存储资源，并对其进行统一管理和访问。

针对跨域异构资源聚合和统一管理及访问问题，目前已有许多关于广域存储系统的研究工作。点对点技术在网络存储系统[7-12]解决广域数据访问和共享问题上发挥了重要作用。这些存储系统通常基于一致性哈希进行数据分区[13]，在分区节点内采用键值结构组织目录树。此类系统去中心化的架构设计使其避免了单点瓶颈，并且还具有较小的集群扩容代价与缩容代价。不过此类架构难以被直接应用于广域高性能计算环境，因其采用一致性哈希随机地决定文件的放置位置，而忽视了文件之间的关联性以及用户的访问成本。WAS（Windows Azure storage）[14]是一种兼具高可用、安全、可伸缩性和可靠性的云存储系统。其基于共享口令的文件访问授权使用户之间极易共享数据。然而WAS不支持差量更新，即使仅覆写了文件的一小部分，也需要将整个文件全量上传到远程数据中心，因此会导致巨大的网络和磁盘开销。近年来，部分研究人员提出了关于面向分布式计算和高性能计算的广域文件系统的构想。Lustre-WAN[15]是尝试将并行文件系统部署到广域的方案。实验结果表明，它可以在100 Gbit/s专线网络上提供较高的聚合访问性能。然而其局限在于要求所有数据中心部署相同的文件系统，即Lustre。Gfarm[16]是一个广域分布式文件系统，由一个元数据节点、多个数据节点组成，同样提供了易用的广域数据访问接口。然而Gfarm存在单点故障问题，即元数据节点出现的网络故障会导致整个系统崩溃。CalvinFS[17]是一种基于广域副本的高可用广域分布式文件系统。CalvinFS利用分布式数据库系统进行元数据管理，并以数据块为建立副本的基本单元。但CalvinFS使用键值的方式组织元数据，

并将文件或目录的绝对路径作为键，因此递归的目录操作可能会导致性能显著下降。OneData[18]是一种用于全局数据存储和共享的解决方案，它是PL-Grid Plus[19]项目的一部分。OneData引入了Space和Provider的概念，以隐藏数据广域分布的复杂性。Space是用户数据存储的载体，Provider是提供存储资源的组织机构，每个Space可以由一个或多个Provider保存。然而OneData的目录树操作会被映射为大量的NoSQL操作，导致其数据库面临极大的负载压力，从而成为性能瓶颈。全局联合文件系统（global federated file system，GFFS）[20]是美国国家科学基金会的极限科学与工程发现环境（extreme science and engineering discovery environment，XSEDE）项目中用于聚合广域分散自治存储资源的全局联合文件系统。与OneData紧密的元数据管理不同，GFFS通过一种松散的顶层元数据组织实现了异构存储资源的聚合，这种资源聚合形式支持个人计算机、大学校园存储集群、云存储等多种来源的存储资源便捷地接入GFFS。GFFS的元数据分多级管理，即顶层元数据集中式管理，存储集群上的元数据分散自治管理。因此，GFFS访问流程中单一的顶层元数据服务器可能会成为瓶颈，导致访问性能受限。

针对大数据稀疏问题，参考文献[21]在基于异构众核的超级计算机上，从主存储器访问、进程间/节点间通信、并行化和负载均衡等方向开展了对存储资源访问性能的优化方法研究，但广域环境因其受限的网络带宽和高访问时延为数据访问性能带来了挑战。为了提升对广域环境中存储资源的访问能力，欧洲数据网格（EU data grid，EUDG）[22]、开放网格服务基础设施（open grid services infrastructure，OGSI）[23]等网格计算系统中应用了大量I/O优化手段。遗憾的是，它们最终都没有实现从用户本地环境到网格环境的完全无缝和透明。Globus项目中提供了副本定位服务（replica location service，RLS）[24]、网格文件传输协议（grid file transfer protocol，GridFTP）[25]以及远程I/O库（remote I/O，RIO）[26]等工具供用户高效使用网络资源，然而用户需要通过GridFTP显式地管理和检查文件状态、来回复制文件，便捷性较低。作为广域存储系统中的常见技术，预取和缓存是提升元数据和数据访问性能的重要手段。网络文件系统4.1版本（network file system v4.1，vNFS）[27]以数据块为粒度对数据进行预取，当应用请求的数据量小于一个数据块时，系统预取整个数据块，以减少后续访问的网络开销。但在带宽受限且时延高的广域网环境中，vNFS小粒度地预取数据块会导致较大的访问时延。参考文献[28]在密集型I/O环境中提出了一种面向突发负载的预取性能优化方法，对突发I/O进行在线识别以进行数据预取，从而有效地节省维护数据关联性所需的资源。但此类方法大多面向单一类型的访问负载，在混合负载中无法有效地识别访问特征。数据副本也是广域存储系统中一种常用的访问性能优化手段。SPANStore[29]键值存储系统通过其定位服务器根据每个云平台的存储价格和负载状态决定文件及其副本存放地址，客户端通过本地云存储平台的元数据索引到最近的目标副本地址，然后访问该副本，以提升访问性能。然而SPANStore的解决方案是面向云环境的，其面向云存储成本的副本策略无法直接用于高性能计算环境。参考文献[30]针对边缘计算与云计算协作场景提出了一种数据副本布局策略，在存储空间有限及最长数据访问时间约束等限制条件下，通过启发式算法计算副本放置位置，以降低数据访问时延。但是其中心化、

全信息感知的静态副本放置策略会带来较大的信息采集开销。而参考文献[31]采用了一种局部信息感知的副本策略，其中WARP算法基于对副本一致性维护开销的分析，通过感知副本的数据写入操作，决定副本的最优数量与位置。但只针对写操作的感知导致该副本动态布局在读性能优化方面仍存在提升空间。

综上所述，在广域高性能计算场景中，现有系统都无法完全满足对广域分布数据的统一管理和高效访问需求，因此笔者设计并实现了全局虚拟数据空间，以聚合广域分散异构存储资源，并支持统一且高效的访问模式。

## 3 方法和关键技术

为了构建适用于广域高性能计算环境的全局数据空间，并实现对广域分布数据的统一管理和高效访问，本文针对存储资源广域分散且隔离自治的国家高性能计算环境中全局虚拟数据空间构建问题，研究了一套适用于广域高性能计算环境的虚拟数据空间构建方法，以实现跨域异构存储资源的统一管理。此外，本文针对带宽受限且时延高的广域网环境下的数据高效访问问题，提出了一套数据访问性能优化方法，以实现对广域分布数据的高效访问。

### 3.1 虚拟数据空间构建方法

为了构建适用于广域高性能计算环境，能屏蔽底层文件系统异构性，并支持统一管理、分区共享及视图隔离的全局数据空间，本节提出了基于广域分散自治存储资源的虚拟数据空间构建方法。该方法需聚合已部署在多个超算中心中的异构存储资源，形成数据空间，并根据高性能计算环境用户及应用对数据空间的多样化访问需求进行分区管理、共享和隔离。因此，针对广域分散存储资源聚合问题、数据的多样化共享问题及用户访问控制问题3个子问题，研究并建立了一套虚拟数据空间构建方法。通过分散自治存储资源的聚合方法，实现存储资源的全局统一逻辑视图；通过可定制数据区域划分和空间分配方法，实现区域虚拟视图，以隐藏数据分布的复杂性，并提供统一的数据分区管理；通过视图隔离和访问控制机制，实现用户及用户组之间的数据共享和隔离。

#### 3.1.1 分散自治存储资源聚合方法

针对高性能计算环境存储资源广域分散、无法充分发挥广域资源聚合效应的问题，GVDS使用基于全局路径的名字空间，以实现广域分散、自治异构存储资源的聚合。

GVDS对广域分散的存储资源从本地、局域、广域3个层级进行逐层聚合，形成全局统一的名字空间，以支撑分散异构存储资源的聚合管理和统一访问。GVDS基于单机文件系统和并行文件系统对存储资源进行本地聚合和局域聚合，并将所有超算中心的存储资源以存储集群的粒度映射到管理节点维护的存储资源名字空间中，以进行广域层级资源聚合。如**图1(a)**所示，每个并行文件系统的根目录都链接到全局管理节点维护的全局名字空间中的逻辑节点，逻辑节点的信息和组织关系被管理节点记录在数据库中，底层存储集群的几个重要属性，包括资源的通用唯一识别码（universally unique identifier，UUID）、超算中心的UUID、已分配容量、已使用容量、可用容量和资源状态等，进而聚合所有存储资源，形成全局数据空间。

GVDS被部署在多个超算中心内，并

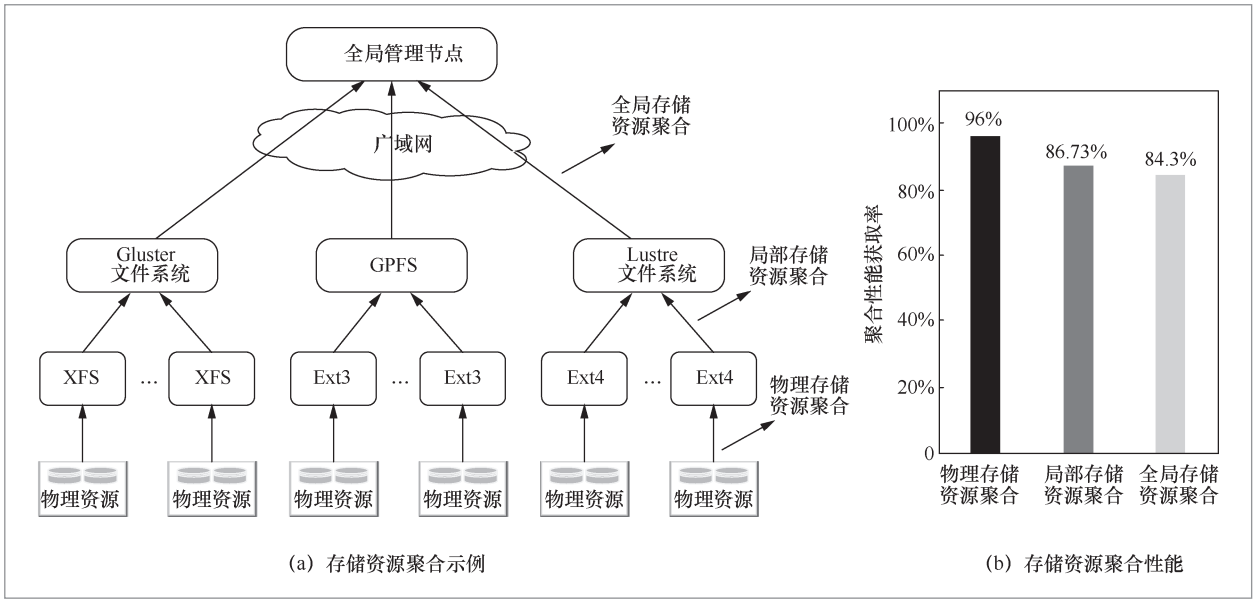



图1 分散自治存储资源聚合方法

聚合了Lustre、Gluster、MooseFS等多种并行或分布式文件系统。如**图1(b)**所示，GVDS的3层资源聚合方法表现出了良好的聚合性能，以较低的性能损失聚合了广域分散且自治异构的存储资源，实现了GVDS的聚合资源管理和统一访问接口。

### 3.1.2 可定制数据区域划分和空间分配方法

广域分散存储资源聚合后，针对数据的便捷管理和共享问题，GVDS引入了“区域”与“空间”的概念，以隐藏数据分布的复杂性，并实现数据的统一管理和共享。区域是对分散数据进行统一管理和共享的平台，空间是承载数据文件的逻辑实体，是存储空间分配、集群映射等的基础控制单元。一个区域通常由多个空间组成。GVDS区域划分和空间分配如**图2**所示，GVDS中可构建个人私有区域、群组共享区域、全局共享区域等不同类型的区域，以作为定制化数据存储和共享的基础，而作为分散数据的集合体，每个区域下设多个空间，这些空间被分配至不同超算中心内的存储集群上，作为承载分散数据的实体。例如**图2**中，用户1可访问个人私有区域“用户1”、群组共享区域“高能物理”以及全局共享区域，而其个人私有区域“用户1”内的数据则被分散存储于超算中心1内的存储集群1-1和超算中心$m$内的存储集群$m$-1上的不同空间内。

GVDS通过构建区域，让用户视图在区域粒度实现隔离，从而隐藏全局数据空间的管理复杂性，而空间创建时的资源分配被映射为逻辑节点创建子节点的操作，进而被映射为不同类型底层存储集群的具体操作。如**图2**所示，用户可以根据需要自定义多个区域，不同科研领域的数据集、用户的私有数据集等均可以存储至不同区域中。同时，GVDS允许用户将新创建的空间映射到存储集群的基础存储容器中，例如并行文件系统的数据目录。如果用户在创建空间时未指定映射的目录路径，则系统会自动根据存储集群剩余容量、区域关联度、用户关联度等信息选择合适的存储集群节点，并创建一个新的映射目录。

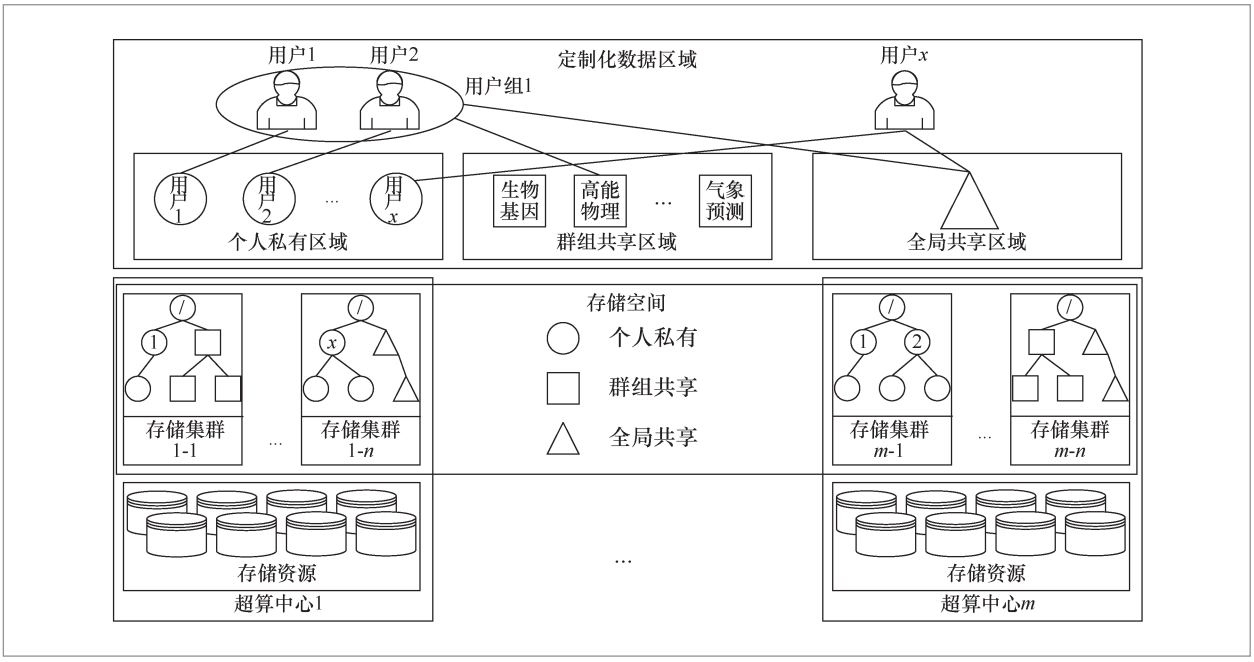



图 2 区域划分和空间分配

GVDS能较好地支持相同类型的数据存储在不同的超算中心的场景。例如在气象预测场景中，异地站点收集的测量数据需要被存储于就近的超算中心，并且进行预处理，因此相同类型的数据需要被分配到不同的超算中心。区域和空间的引入可以有效地管理这些地理分布的数据，如**图2**中，用户可以为气象预测数据创建一个区域，然后在该区域中创建多个空间，并将空间各自映射到距离站点较近的超算中心，以存储相关数据。

GVDS可定制化的数据区域划分和空间分配满足了用户对数据管理和共享的多样化需求，可支撑个人、群组、全局等不同层级的数据管理和共享。此外，GVDS以拓展属性接口形式提供了扩展应用程序接口（application programming interface，API），用于获取空间实际映射位置信息。这允许高性能计算环境中的作业调度程序有感知地将作业调度到数据文件所在的计算中心，以最大化数据局部性，从而降低远程数据访问或迁移的开销。

### 3.1.3 视图隔离与访问控制机制

为了支持GVDS中数据的安全共享和隔离，针对用户访问控制问题，本节研究了GVDS中视图隔离和访问控制机制。如第3.1.2节所述，GVDS中的区域可以在多用户之间共享。创建区域后，该区域的所有者可以将区域的访问权限授予其他用户或用户组。例如，用户可以创建一个区域，并向一个用户组授予其对该区域的只读权限，然后该用户组的成员将继承用户组的只读权限，并且能够访问该用户的区域。

GVDS管理节点的数据库中记录了每个区域对用户和用户组的访问控制列表权限。当用户登录GVDS客户端时，将获取其可访问的区域列表，管理节点将使用N1QL在数据库中搜寻所有该用户可见的区域。并且客户端会定期向管理节点发送区域请求，以获取最新的区域信息。

GVDS使用一对多映射模型来控制访问权限，一个GVDS账号将会被映射到多个超算中心的专属本地账号或者GVDS统一账号。如果用户账号被映射到超算中心专属本地账号，则该账号名下的所有映射到该超算中心的空间在存储集群中的所有者等信息都会变更为专属账户用户，基于此，本地超算中心计算节点上的GVDS客户端即可绕过I/O代理直接访问空间中的文件。如果用户账户被映射到GVDS统一账户，则其权限将采用GVDS的区域空间权限管理机制，通过管理节点鉴权产生的凭证经由I/O代理访问相应数据空间。

基于视图隔离和访问控制机制，GVDS建立了全局一站式权限控制和认证体系，实现了广域分散存储资源的共享和隔离访问。

### 3.2 数据访问性能优化方法

GVDS聚合了跨域分散且异构的存储资源，形成了统一的虚拟数据空间，并面向用户和应用提供统一的访问接口及便捷的管理和共享。然而，仍需进行大量的性能优化工作才能满足用户对广域分布数据高性能、低时延的访问需求。因此，本节研究了虚拟数据空间中的数据访问性能优化方法。针对带宽受限且时延高的广域网环境下的高效数据读、写及数据优化布局3个子问题，通过基于负载特征的数据预读方法，实现高效的数据读取；通过基于请求合并的回写缓存方法，有效提升数据写性能；通过副本布局及一致性同步方法，实现数据块副本的广域优化布局及低开销同步。在本节中，笔者将详细描述这些性能优化措施。

#### 3.2.1 基于负载特征的数据预读方法

针对广域带宽受限且数据广域分布场景下的数据高效读取需求，GVDS实现了基于负载特征的自适应增量预读方法，通过感知访问的带宽负载、文件I/O等特征，基于缓存的命中概率动态调整预读块的长度，实现了广域环境下多文件海量数据的高效访问。

在读取数据时，如果在客户端缓存中找到请求的数据，即可确认读取完成。而当发生缓存未命中时，则需要通过与I/O代理联系来读取请求的数据。如**图3（a）**所示，GVDS使用文件缓存控制块来记录文件访问特征，并决定是否需要进行预读以及确定预读的步长。当用户提交文件读请求时，文件缓存控制块会记录该文件的历史访问信息，并根据其访问的数据块大小、存储区地址、历史预读块及其命中率等信息分析其访问特征。文件访问的随机读和顺序读特征是GVDS预读机制是否发挥作用的关键，当GVDS的预读机制发现当前文件的历史顺序访问数据量已经超过预设的启动阈值之后，将会启动预读机制，分配预读缓存，并行预读若干个数据块。在预读机制启动后，GVDS预读机制则会根据预读缓存的命中率进行增量预读，即在判定预读缓存被访问后，自适应地增加预读数据量。此外，预读的数据量与预读步长正相关，而预读步长则由带宽压力和文件I/O特征等负载特征综合决定，以平衡系统负载和数据访问性能。其中，带宽压力由I/O请求调度器提供，其与I/O请求调度器中等待队列中的请求数量成正比，较大的带宽压力意味着较大的I/O负载，此时预读步长会自适应地减小，以减轻负载；文件I/O特征则包括文件的预读命中率、I/O请求最大吞吐量等，较高的预读命中率和较大的I/O请求吞吐量都会增加预读步长，以提升访问性能。

如**图3（b）**、**图3（c）**所示，GVDS的数据预读方法可有效地提升顺序读时的数据访问性能，而随机读取时，面对读放大问

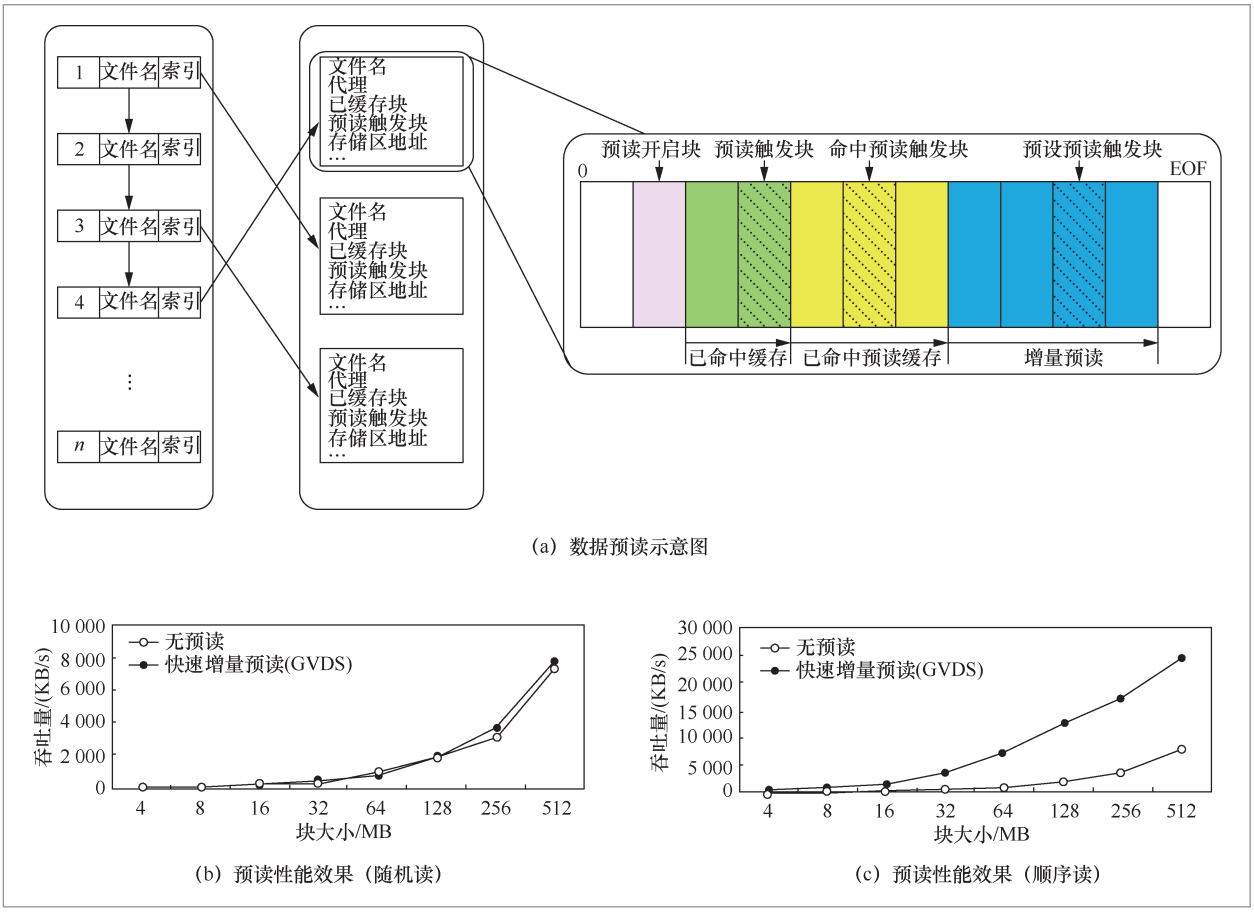



图 3 基于负载特征的数据预读方法

题，GVDS通过动态访问模式检测动态调整预读机制，从而获取较低的性能损失。综上所述，GVDS通过基于负载特征的数据预读方法实现了广域环境中的数据高效读取。

### 3.2.2 基于请求合并的回写缓存方法

在数据写入方面，针对广域环境中的数据高效写入需求，GVDS实现了基于请求合并的回写缓存方法。GVDS客户端聚合可乱序执行的数据I/O请求，并通过回写缓存将虚拟数据空间客户端提交的数据暂时缓存在本地，由后台线程异步提交，进而提升远程数据的写入性能。

数据写入请求将数据写入客户端缓存而非数据源，客户端不断将缓存数据刷新回数据源。由于可同时将多个写入请求发送到远程超算中心内部署的I/O代理上，因此GVDS采用异步写方式显著提高写性能。此外，每个请求都需要通过高时延广域网络通信，这会带来较大的网络开销，并且对文件的多个不连续部分的访问请求经常被交叉提交，因此如**图4(a)**所示，在将写请求发送到I/O代理之前，用户级别的守护程序会合并这些请求，以提高I/O性能。

为了提高并发请求处理的性能，GVDS客户端允许存在多个线程，以便每个线程可以独立选择和处理队列中的并发请求。当从I/O代理接收到响应时，用户级

别的守护程序会将其写回到用户空间文件系统(filesystem in userspace, FUSE)[32]的内核模块，然后驱动程序将请求标记为已完成，并唤醒用户进程。

如**图4(b)**所示，GVDS请求合并方法能有效地提升小数据访问块时的写入性能，因为其对请求的合并能有效地降低高时延网络通信开销。但是受到FUSE请求切割机制的约束，在数据访问块大于64 KB时，请求合并方法将无明显的性能优化。此外，GVDS基于请求合并的回写缓存方法能从整体上保持较高的远程数据写入性能。

### 3.2.3 副本布局及一致性同步方法

除了对数据读取和写入进行优化外，数据布局也是提升系统跨域访问性能的重要方法之一。针对广域环境中的数据优化布局问题，GVDS中设计并实现了一种数据块粒度的多副本机制及副本间的一致性同步方法，以优化数据在广域环境中的布局，并实现低开销的副本同步。

首先，GVDS基于负载特征感知实现副本的动态布局。如**图5(a)**所示，GVDS基于本地超算采集的I/O负载特征来预估并构建以访问时延为收益的计算模型，并结合对副本异地更新I/O负载的感知来决策副本的创建与布局。为了避免负载特征的采集给I/O代理带来过大的开销，客户端维护其本地打开的远程文件的历史访问请求统计信息，并周期性地提交给I/O代理的负载特征度量子系统。GVDS副本动态布局策略能够凭借有限时间窗口内的局部数据访问信息、历史积累的目录及文件访问模式统计量对文件的负载特征进行识别，并根据识别到的数据负载特征实现副本动

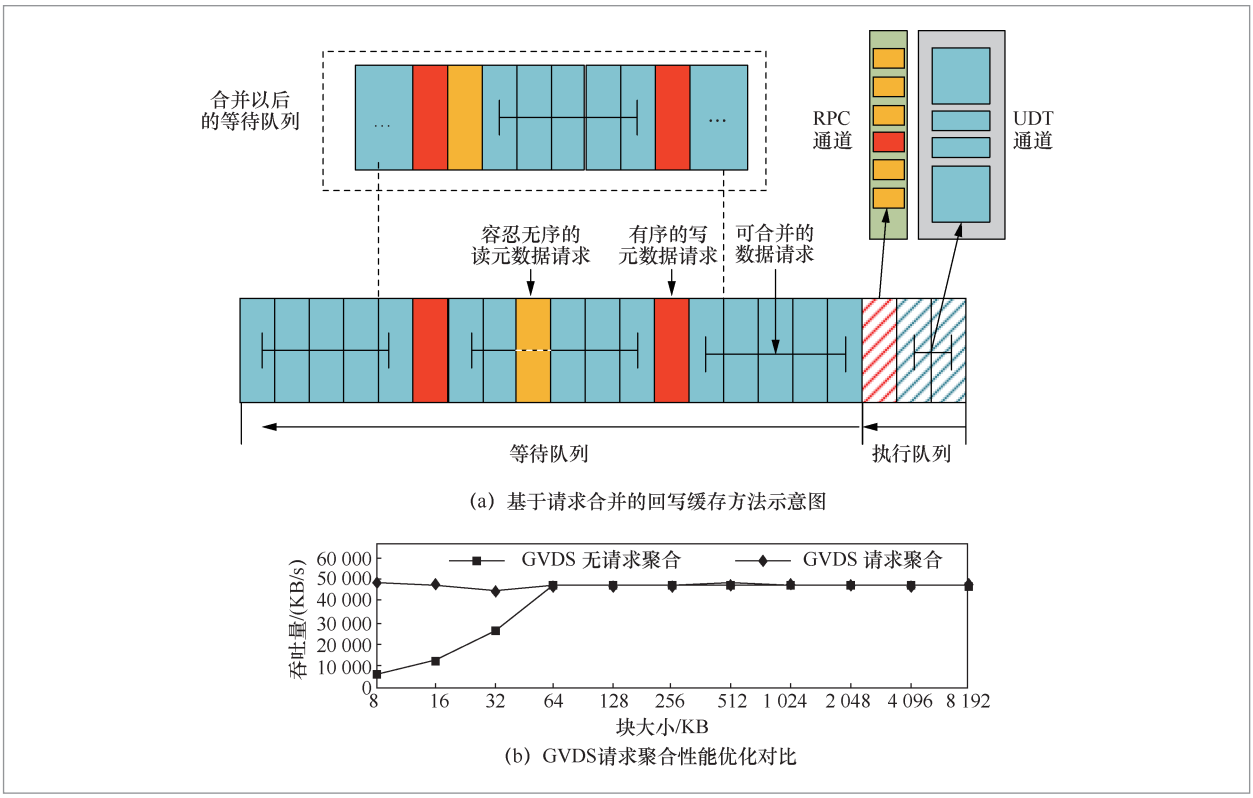
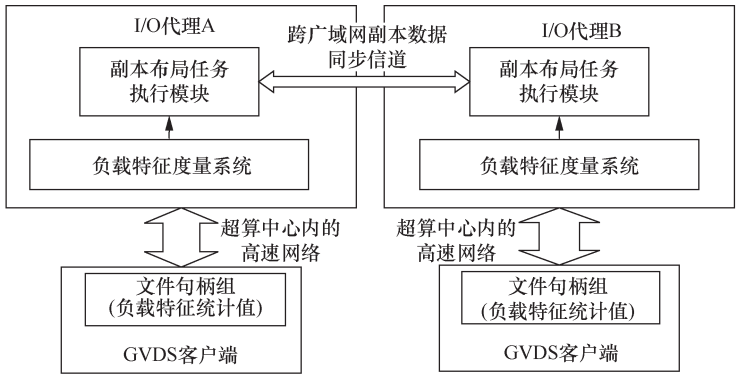



图 4 基于请求合并的回写缓存方法

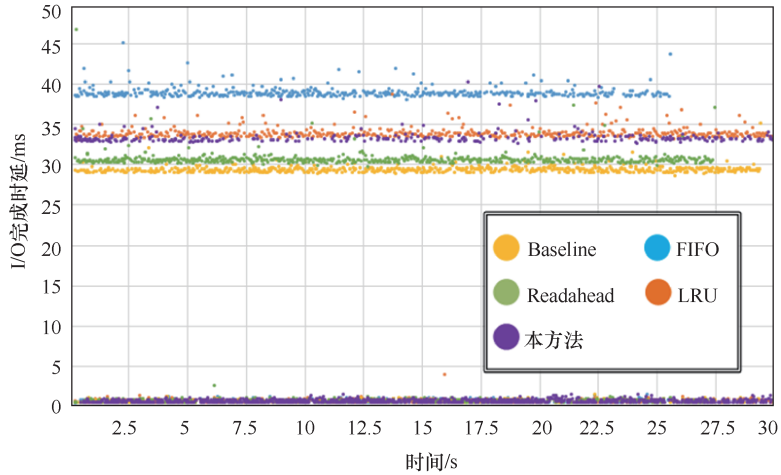
态布局。图5（b）表明，GVDS中的副本布局方法能有效地提升广域环境中的文件访问性能。

数据布局执行后，如何以较低的同步开销保证副本数据间的一致性是GVDS面临的重要挑战之一。GVDS针对广域环境中副本空间的目录树同步问题和并发写的一致性问题，分别采用了渐进一致性保证的目录树同步方法和因果一致的副本数据索引同步方法。

副本空间内目录树的一致性是存储系统需要满足的基本属性和首要保证。如图6（a）所示，GVDS采用了一种适用于广域网环境的副本空间目录树同步架构。其通过冲突检测识别或预判广域网络中目录树节点的并行冲突状态，通过为节点消息同步选择渐进一致性保证的同步模型中强一致或者弱一致的同步信道，最终达成了较强的一致性以及高性能访问的目的。GVDS中目录树的渐进一致性同步模型将I/O代理划分到两种不同一致性强度的共识组，将单中心内的I/O代理节点划分为域内共识组，负责中心内的目录树副本状态同步。域内共识组中的一个特殊节点会被指派为边缘I/O代理，参与到GVDS系统的跨域共识组中。跨域共识组负责维护跨广域网多个超




（a）负载特征感知的副本动态布局架构




（b）访问时延测试

图 5　负载特征感知的副本动态布局方法

算中心的副本空间内的目录树的一致性，所有对跨域副本空间的目录树修改操作都会由域内共识组的边缘节点发起并提交。

对于数据并发写的一致性问题，GVDS采用一种因果一致的副本数据索引同步方法。如图6(b)所示，GVDS通过I/O请求逻辑时间戳对访问请求进行定序，并基于多版本区段树的数据索引方法解决索

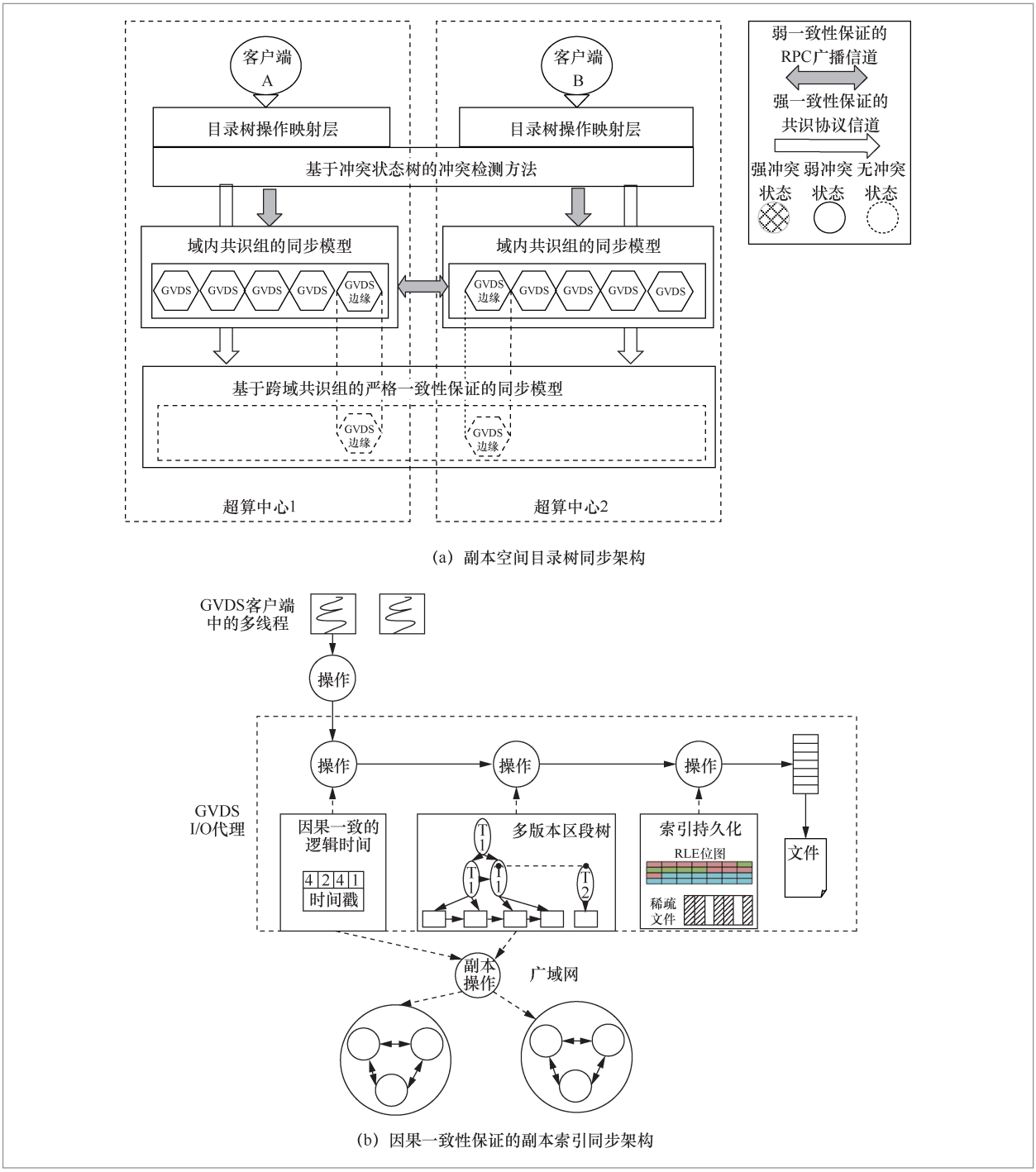



图 6 副本空间一致性保证

引更新请求乱序到达的冲突问题，实现了副本数据并发写的因果一致性。保证因果一致的I/O请求逻辑时间戳主要由版本矢量、机器时间戳以及源超算中心3个值构成。其中，版本矢量用于标记请求之间的因果依赖关系，机器时间戳以及源超算中心的ID号用于为I/O请求提供全序关系，以在因果一致性模型的并行更新时解决冲突。此外，I/O代理节点会为目标副本文件构建多版本区段树，并通过树的版本回滚来解决网络包乱序到达导致的并发冲突。

如图7所示，相对于访问广域中的数据空间，在应用副本布局及一致性同步方法时，GVDS能实现低时延的副本空间访问。综上所述，GVDS通过优化的副本布局实现了广域数据的高效访问，并在多副本之间形成了低开销的有效同步。

## 4 系统设计与实现

### 4.1 GVDS软件体系结构

GVDS中引入了用户、用户组、区域、空间等实体，以实现对广域分散存储资源的聚合、共享与隔离。GVDS实体关系如图8所示。

每个GVDS用户都对应一个包含基本权限信息、组关系和区域空间列表的GVDS账户，例如在图8中，用户3与区域1和区域2关联，即区域1、区域2对用户3可见。用户组由多个用户组成，用于进行批量的数据共享和访问权限控制，组的权限将被其成员用户继承，例如在图8中，用户1和用户2是同一用户组的成员，因此用户1和用户2将继承用户组对区域1的访问权限。区域可以隐藏数据访问的复杂性，并简化数据共享过程，用户数据在区域级别共享，用户通过加入区域来获取某些领域或者科研机构所共享的数据集。空间用于隐藏数据位置映射的复杂性，并简化对数据的访问，是保存用户数据的逻辑容器及数据调度和副本控制的一级单位。创建空间时，调度程序会根据存储集群剩余容量、区域关联度、用户关联度等信息将空间映射到合理的超算中心。空间的存储资源由超算中心的I/O代理提供。如在图8中，区域1中含有空间1和空间2两个空间，空间1和空间2分别存储于超算中心1和超算中心2中，并分别由部署于两个超算中心的I/O代理1和I/O代理2提供访问支持，同时I/O代理2还支持区域2下的空间3。

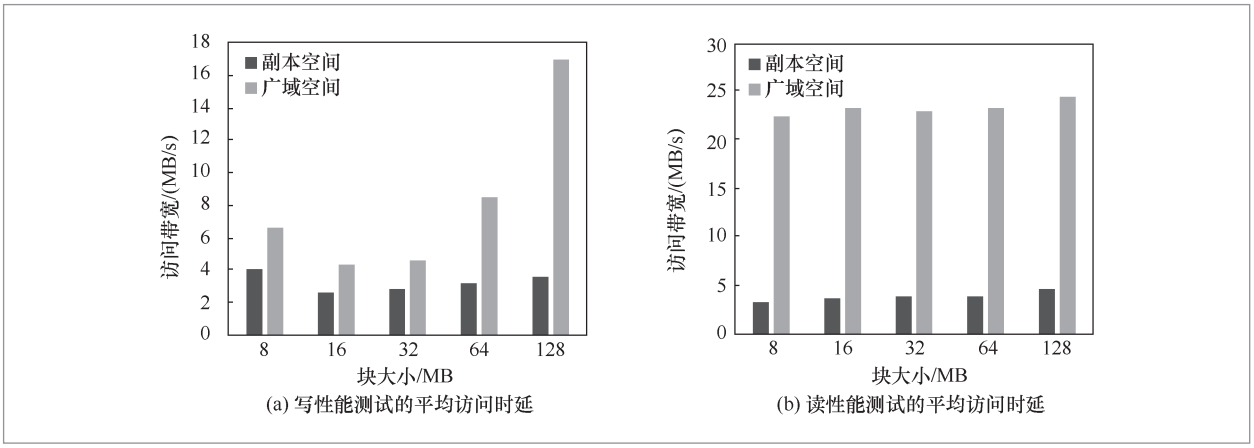



图 7 副本空间与广域空间的访问性能对比

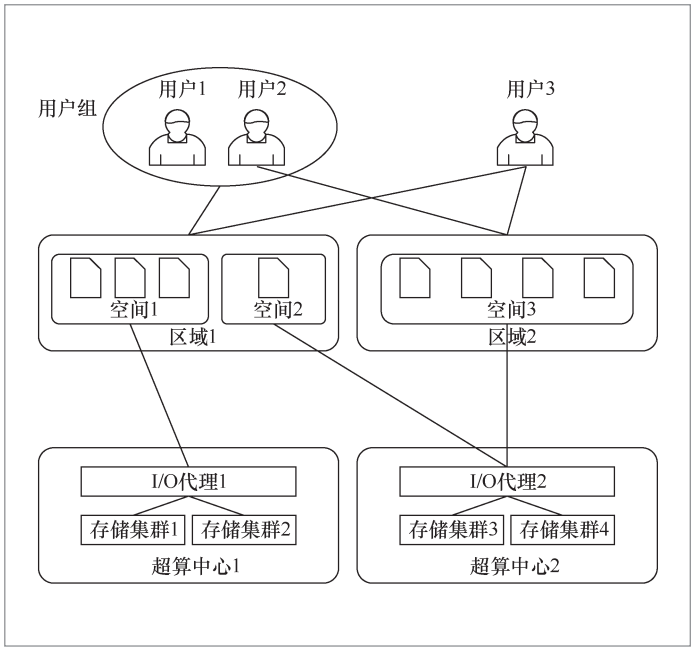



图 8 GVDS 实体关系

GVDS软件体系结构如图9所示。GVDS管理节点中的模块提供对存储资源的管理功能，并使用分布式数据库维护控制信息，包括存储资源状态、用户与组成员信息、区域信息与空间信息、I/O代理拓扑图等信息，该数据库部署在每个超算中心，并且彼此同步。I/O代理节点向广域网暴露存储资源的访问接口。GVDS客户端从GVDS入口点获取超算中心的拓扑图，并且以网络连接质量为标准，对所有超算中心进行排序。GVDS客户端会优先选择同一超算中心的GVDS管理节点实例，其次选择网络距离最近的超算中心管理节点实例。GVDS客户端向管理节点拉取用户身份认证、用户可访问的区域及空间等信息。经由管理节点身份验证的

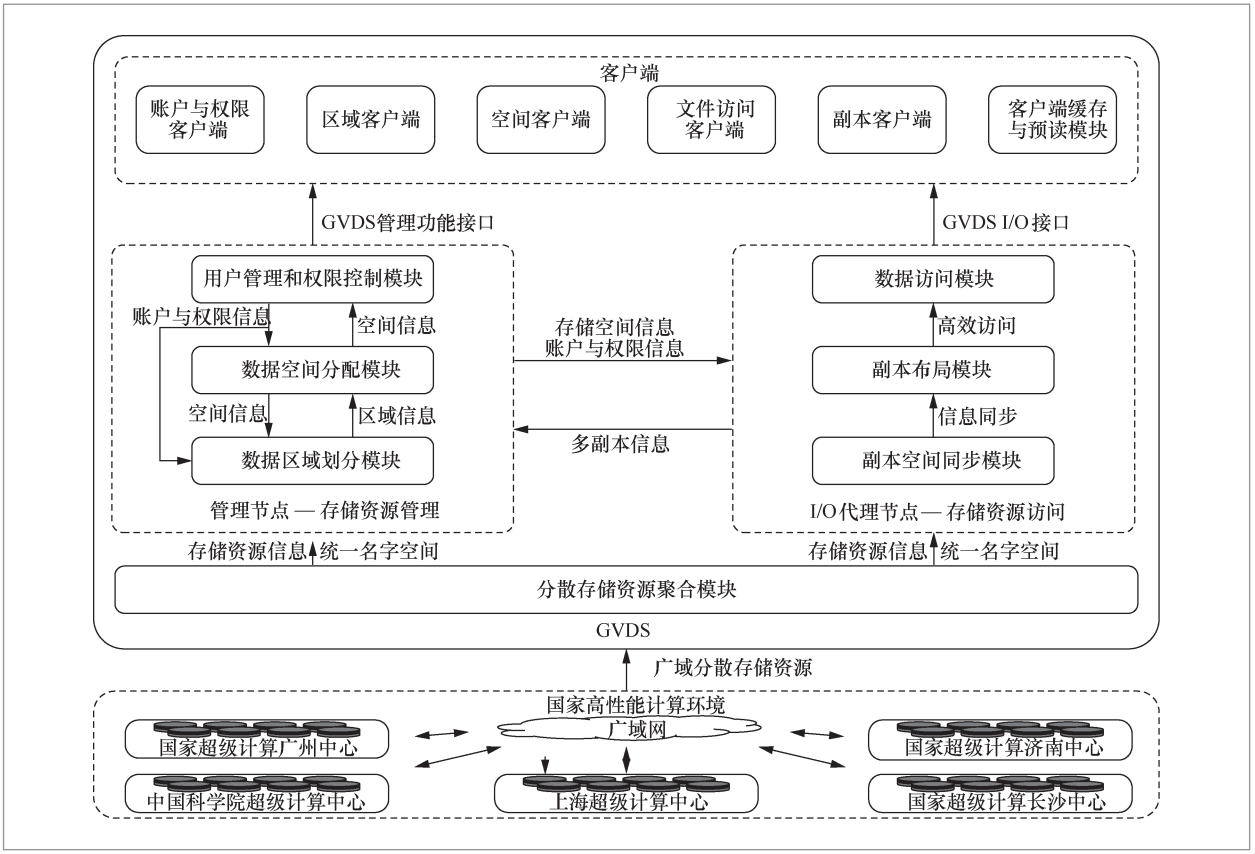



图 9 GVDS 软件体系结构

客户端有权访问GVDS广域环境中的所有I/O代理节点，并且从其名下的空间读写文件数据。在数据访问过程中，客户端会将客户端视图的逻辑文件路径转换为GVDS全局资源路径，该全局资源路径最终在I/O代理中被转化为实际资源存储路径。GVDS客户端基于FUSE构建POSIX访问接口，将POSIX语义的请求转换为GVDS语义的请求，并发送到I/O代理端数据访问模块处理，实现统一访问接口。此外，GVDS客户端的预读和缓存机制及I/O代理节点中的副本机制将优化广域环境中的数据访问性能，以实现高效数据访问。

## 4.2 系统实现

GVDS系统架构如**图10**所示，GVDS系统可以部署在多个超算中心内，基于每个超算中心内的存储资源构建若干个管理节点和I/O代理节点。GVDS客户端通常被部署于超算中心内的计算节点上，也可以被部署在超算中心之外，并通过广域网访问超算中心的存储资源。

### 4.2.1 管理节点

单个超算中心需要至少一个管理节点，管理节点通过使用全局知识构建的全局名字空间，极大地简化了存储资源聚合、空间放置决策和I/O代理管理。为了降低管理节点的负载，避免管理节点成为文件访问性能瓶颈，尽量避免其参与到主I/O路径中。GVDS管理节点仅维护用户视图中第一级区域和第二级空间的元数据信息，空间下文件的元数据全部由空间所映射超算中心的I/O代理维护。GVDS客户端仅在初次访问或者区域空间的元数据缓存超时时，才会向管理节点发起数据访问请求。

客户端在访问文件时，会与管理节点和I/O代理节点产生交互。客户端请求由管理节点维护的区域和空间信息，获取与当前登录用户关联的区域和空间信息、集群拓扑图等。基于从管理节点获取的信息，客户端将构建一个由区域逻辑节点和空间逻辑节点组成的两级目录树。进入区域目录时，客户端需要向管理节点发送带有区域ID的请求，以获取有关区域内的空间信

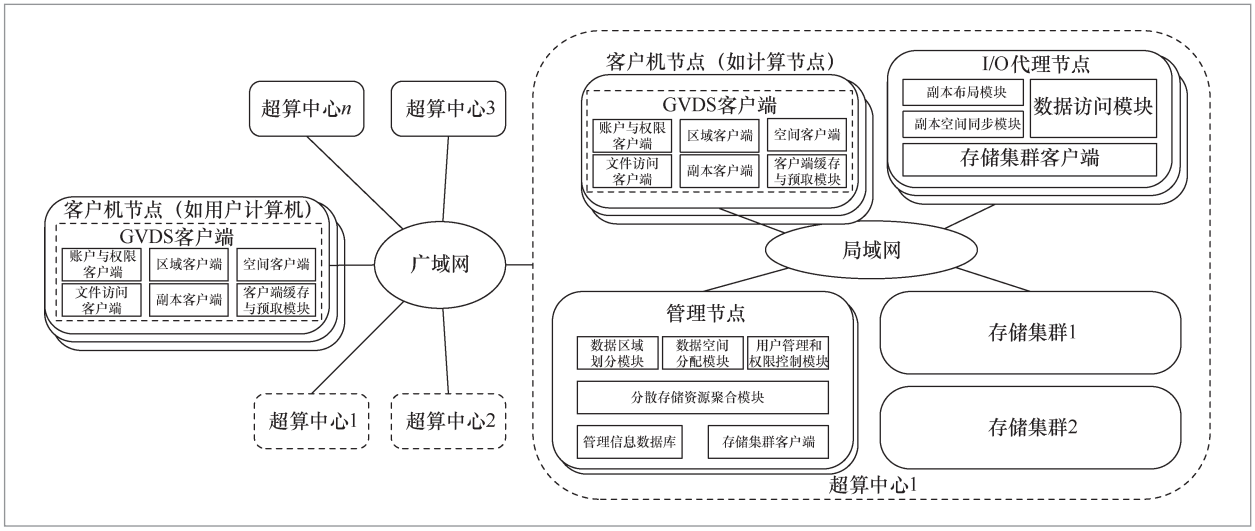



图 10 GVDS 系统架构

息。空间信息则包含空间全局映射统一资源标识符（uniform resource identifier，URI）、空间权限信息等，其中空间全局映射URI是存储用户文件的I/O代理入口点。管理节点动态采集I/O代理的负载信息，监控其健康状态，并且向客户端反馈I/O代理的负载信息，不可用的I/O代理将会被管理节点屏蔽，避免客户端数据访问超时对计算任务产生影响。客户端可以基于缓存局部性、最低负载等策略选择同一超算中心内的任意一个I/O代理访问其空间。

为了提高可靠性与自治性，GVDS环境中的每个超算中心均部署了一个分布式数据库实例。主机使用Couchbase[33]数据库系统存储用户、区域、空间、I/O代理和存储资源等信息，并且依赖其跨数据中心复制机制实现超算中心间跨域的最终一致的控制信息复制。客户端和I/O代理通常与位于同一超算中心的管理节点联系，以避免广域网通信的开销。当同一超算中心中的管理节点发生故障时，客户端和I/O代理会联系较近的超算中心的管理节点。

#### 4.2.2 I/O代理节点

I/O代理节点为客户端提供空间分配的存储资源的访问接口。I/O代理节点的挑战在于保证实际应用的文件操作与客户端POSIX语义请求的一致性。例如当客户端打开文件时，需要获取文件句柄，以进行后续的文件读取/写入。如果选择让I/O代理节点执行同样的打开文件操作、返回文件句柄，并且让客户端在随后的文件读写中使用文件句柄作为文件访问入口点，这种操作的一一对应反而将造成与实际效果的不一致。例如，当I/O代理负载平衡、I/O代理句柄过多自动删除、句柄号长时间未用被回收并被其他客户端使用时，都会发生因句柄失效而产生的不一致操作。

而生成GVDS环境中全局唯一句柄号的开销过于庞大，因此I/O代理使用文件路径打开文件，然后执行实际的读写操作，最后关闭文件。并且，为了减少文件打开带来的开销以及文件关闭引发的不必要的缓存冲刷操作，I/O代理将文件路径作为键来缓存文件句柄，采用最近最少使用（least recently used，LRU）策略驱逐长期未访问的句柄，并实际关闭该文件。

#### 4.2.3 GVDS客户端

用户应用程序使用GVDS客户端访问文件。GVDS客户端基于FUSE向用户应用程序提供POSIX文件访问接口。它一般被安装在超算中心内的计算节点上，用于获取执行计算时所需的数据；或者被安装在用户PC上，用于上传输入文件或者浏览计算结果。GVDS客户端在用户空间中提供访问服务，与大多数本地文件系统相似，GVDS提供读取、写入、删除文件的操作，以及创建、移动和删除目录的操作。用户应用程序使用客户端文件视图中的逻辑文件路径来访问文件数据，而无须知道文件实际所处的超算中心。访问文件时，客户端将逻辑文件路径转换为GVDS全局资源路径，并且根据文件的全局资源路径从GVDS环境拓扑图中寻找适合的I/O代理，将POSIX语义的文件访问请求翻译为GVDS文件访问请求格式，并最终向I/O代理入口点提交请求。

## 5 实验验证

为了从各个方面评估本文提出的系统，笔者在国家高性能计算环境以及阿里云环境中对GVDS进行了实验验证。在国家高性能计算环境中广域分布的5个超算

节点部署了GVDS，并测试了GVDS的功能及性能。此外，在阿里云中的5个跨域分布的数据中心中部署了GVDS以及流行的网络存储系统，以进行性能对比测试，并验证了GVDS系统在不同环境中的适用性。

## 5.1 功能验证

在GVDS功能验证方面，在部署的实验平台中开展了GVDS典型使用场景测试验证工作。通过数据区域的定制化共享、远程大数据集的按需实时访问、广域分布数据的多中心协同处理等GVDS使用场景，验证GVDS系统全局统一视图、广域数据共享、远程数据按需访问、跨域数据协同处理等重要功能。

（1）典型使用场景1：数据区域的定制化共享

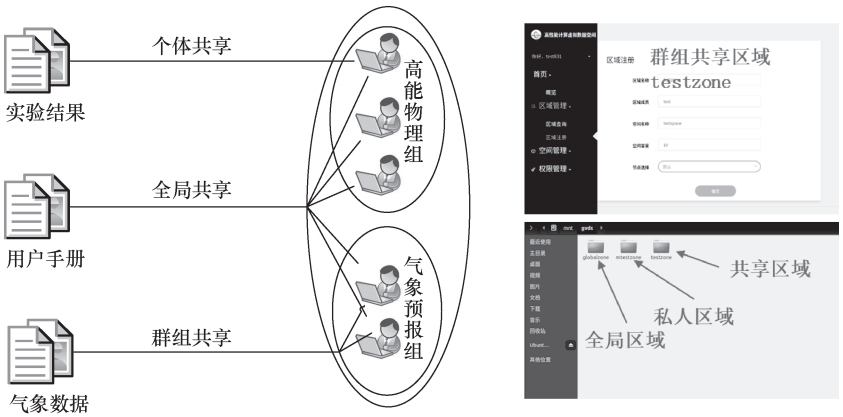
当前高性能计算环境中，在数据广域分散、各中心隔离自治的情况下，GVDS提供了一种能满足用户多样化共享需求的灵活数据共享模式。如**图11（a）**所示，利用GVDS数据区域管理和权限管理等功能，可将数据区域定制化共享给个体用户、群组用户、全部用户等多层级，获得可靠安全且灵活的数据分区能力。验证情况如**图11（b）**所示。

（2）典型使用场景2：远程大数据集的按需实时访问

针对国家高性能计算环境中用户对远程大规模数据集的访问需求，现有的访问模式在进行跨域访问和管理时较为困难，甚至需要远程迁移整个数据集。GVDS系统可通过统一文件虚拟视图，以标准文件访问接口直接跨域访问大数据集中的任何数据，如**图12（a）**所示。GVDS系统可按需实时访问所需数据块，而不必传输整个大数据集。验证情况如**图12（b）**所示。

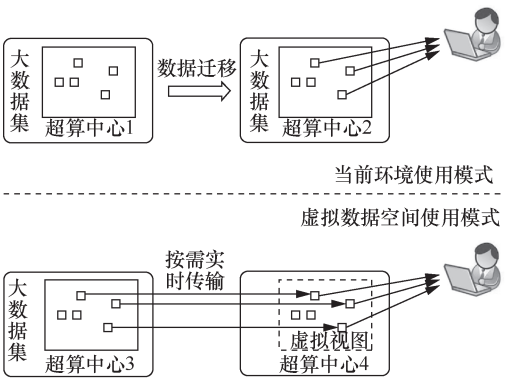
（3）典型使用场景3：广域分布数据的多中心协同处理

随着科学和工程问题的复杂化，对跨域分布数据的多中心协同处理已成为未来大型应用重要的发展趋势。为了满足上述需求，如**图13**所示，GVDS系统基于广域分布数据的全局统一视图支撑数据多中心协同处理。计算任务可以分布于不同的超算中心，并可直接访问分布于不同地点的数据，数据的分布和迁移由GVDS透明地完成。

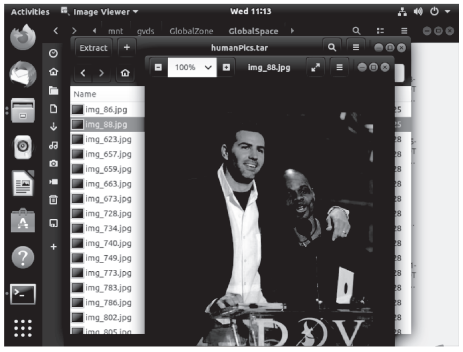



（a）数据区域共享使用场景　　（b）功能验证界面

图 11　数据区域的定制化共享




(a) 远程数据按需实时访问



(b) 通过文件视图跨域访问数据集中的部分数据

图 12 远程大数据集的按需实时访问

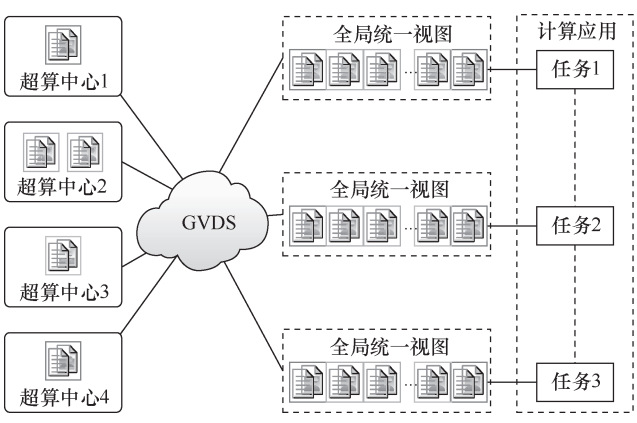



图 13 广域分布数据的多中心协同处理

## 5.2 性能测试

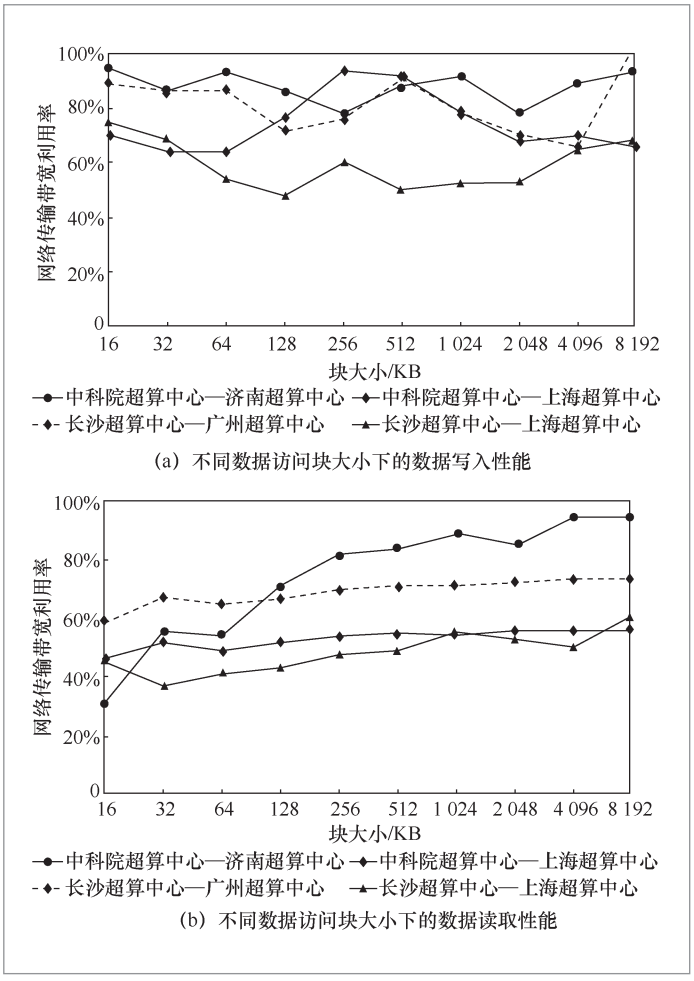
笔者在国家高性能计算环境中进行了GVDS系统访问性能和网络传输带宽的对比测试，采用了4种不同平均带宽的跨超算中心链路进行文件读写压力测试，并记录了GVDS的文件读写吞吐速率与网络传输带宽比值。各超算节点间的平均网络传输带宽分别为中国科学院超级计算中心（简称中科院超算中心）到国家超级计算济南中心（简称济南超算中心）112 125 KB/s、中科院超算中心到上海超级计算中心（简称上海超算中心）82 083 KB/s、国家超级计算长沙中心（简称长沙超算中心）到国家超级计算广州中心（简称广州超算中心）3 887 KB/s、长沙超算中心到上海超算中心118 KB/s。**图14**表明，在多种广域网络链路条件下，GVDS文件写入性能平均能达到约80%的最大网络传输速率，读取性能平均能达到约50%的最大网络传输带宽速率。

此外，在阿里云的5个数据中心进行了GVDS与3个网络存储系统的性能对比测试。5个数据中心分别位于北京、上海、青岛、深圳和成都，每个中心各创建3~4台2xLarge的云服务器（elastic compute service，ECS）实例，每台实例拥有广域网的32 Mbit/s带宽。将GVDS与以下3个存储系统进行比较：安全外壳协议文件系统（secure shell file system，SSHFS）[34]、网络文件系统（network file system，NFS）[35]、OneData[19]。笔者使用Fio[36]生成所需类型的I/O操作，以评估所有系统的数据访问性能。

**图15**展示了多种存储系统在广域环境中的元数据访问性能对比。可以看出，基

于GVDS在元数据访问方面的优化工作，GVDS的文件创建速度仅次于NFS，且文件删除速度快于其他网络文件系统。此外，GVDS为了优化元数据访问性能，对目录项元数据进行了预读优化。这种优化在创建或删除文件等元数据操作时会有较好的优化效果，在列文件目录操作时会导致不必要的开销，同时OneData实现了元数据的自管理推送更新且SSHFS对目录项有缓存优化，因此GVDS的性能略低于OneData和SSHFS。

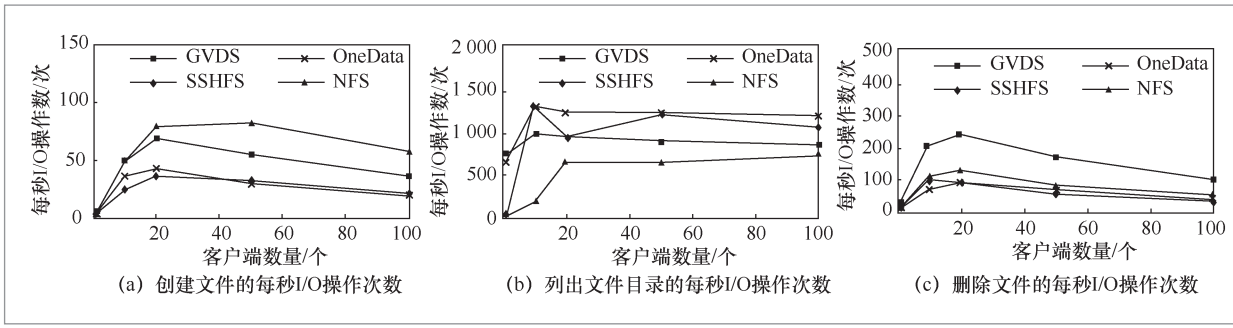
对于顺序读、写压力测试，单个压力器线程以不同的数据块大小顺序刷入10 GB的数据，而单次实验中，压力器读取或写入的数据块的大小是相同的。如**图16（a）**所示，对于数据顺序写入，当单次写入数据块小于256 KB时，GVDS、SSHFS、OneData的吞吐量要远远高于NFS。原因在于，写入请求在NFS中串行提交、同步写入，数据块越小时，相同I/O量的网络请求数量越多，而每次写入都需要串行通过高时延的广域网通信，从而导致了NFC较差的写入性能。SSHFS、GVDS和OneData均选择了回写机制刷写缓存到远程数据中心，这有益于提升性能，但是违背了客户端同步写入的语义，引入了数据丢失的风险。为了解决这个风险，GVDS只对远程数据中心或者访问数据块大小小于“带宽×时延÷2”的请求进行回写缓存，并且当应用程序开启同步标志时，还会优先选择磁盘文件作为持久缓存，尽量保证数据持久性。与此同时，在块大小从512 KB增加到4 096 KB的过程中，NFS的吞吐量显著增



（a）不同数据访问块大小下的数据写入性能

（b）不同数据访问块大小下的数据读取性能

图 14 不同数据访问块大小下的数据访问性能



（a）创建文件的每秒I/O操作次数

（b）列出文件目录的每秒I/O操作次数

（c）删除文件的每秒I/O操作次数

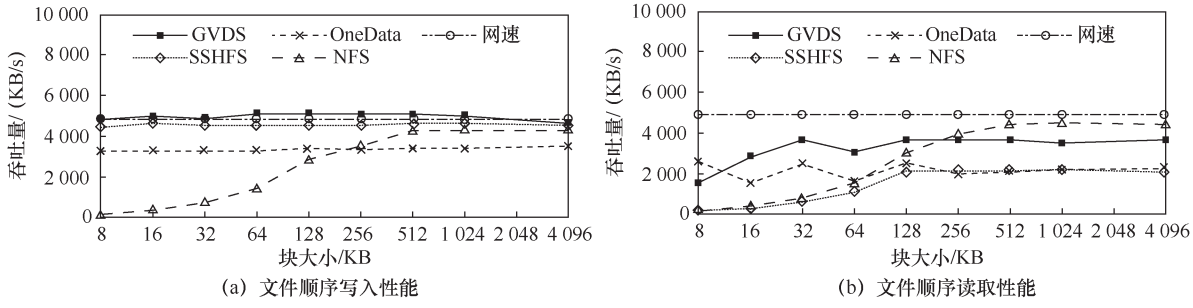
图 15 不同数据访问块大小下的数据顺序访问性能

加，并且基本达到网络带宽瓶颈。从实验结果可以看到，GVDS基本具有最佳的写入性能，相比SSHFS和OneData，分别平均高9.94%和49.18%。

**图16（b）**显示了顺序读取数据的实验结果。GVDS顺序读性能相比SSHFS、OneData和NFS，分别平均高131.22%、52.84%和35.89%。当单次读取数据块小于256 KB时，GVDS的文件读取性能一直有较高的优势，这是因为GVDS针对广域网环境特性设计了链式预读机制。GVDS依赖文件的历史访问负载特征进行预读判断，相比SSHFS以及OneData的固定块预读机制，能更早地触发预读，并且两次预读之间没有明显的停顿时间。当块大小超过256 KB时，NFS展现出比所有基于FUSE接口的网络存储系统更高的吞吐量。这是由于FUSE会将大块数据读过程切分为对多个128 KB大小固定块的串行子过程，成倍提高了请求的总响应时间以及请求总数。而NFS的大块数据读子过程为并行执行，没有该缺陷。

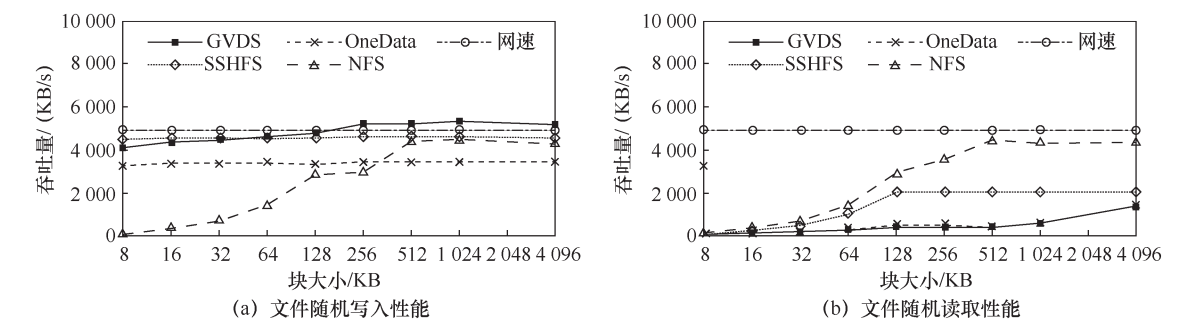
对于随机读、写压力测试，单个压力器将1 GB数据随机写入具有不同块大小的文件中，或从具有不同块大小的文件中随机读取数据。如**图17（a）**所示，随着块大小的增加，GVDS、SSHFS与OneData的随机写性能没有明显变化，其瓶颈均在于广域网链路带宽。其原因与顺序写入相同，三者均采用了回写缓存以加速写入过程，并且GVDS提供了更强的数据持久性。块大小增加到512 KB后，NFS与其他网络存储系统的写入性能基本持平，达到广域网链路瓶颈。在随机写过程中，由于数据块相互覆盖的情况以及缓存导致的误差，GVDS的写入性能在部分测试项中略微超过了链路最大带宽。

如**图17（b）**所示，当数据访问块大小在128 KB以下时，NFS性能与SSHFS相




（a）文件顺序写入性能　　（b）文件顺序读取性能

图16　不同数据访问块大小下的数据顺序访问性能




（a）文件随机写入性能　　（b）文件随机读取性能

图17　不同数据访问块大小下的数据随机访问性能

差不大，访问块大小超过128 KB以后，由于FUSE的串行子过程机制，基于FUSE访问接口的3个网络存储系统（SSHFS、OneData以及GVDS）表现均劣于NFS。并且在随机读负载下，预读机制并没有发挥作用，反而挤占了OneData以及GVDS的正常访问流量，导致这两者的随机读访问性能低于同类型的SSHFS。

综上所述，在对广域分布数据的访问性能方面，GVDS在顺序读写和随机写性能上优于其他主流网络文件系统，而在随机读性能方面仍存在进一步的优化空间。

## 6 结束语

GVDS是为了满足国家高性能计算环境中对广域分布数据的统一管理和高效访问需求而设计并实现的跨域分布式存储系统。GVDS不仅是国家高性能计算环境中提供基础存储服务的工具，还能与CNGrid基础架构进行结合，进而丰富其数据层面的跨域协作与资源共享。

GVDS具有以下优点。

● 位于异地超算中心的异构存储资源可以被统一管理、共享和访问。由抽象的区域、空间以及空间内部实际目录与文件所构建的文件系统视图可以提供透明的跨域数据共享与隔离。

● 支持文件数据按需加载。通过数据块粒度的数据迁移可以从远程超算中心加载文件的任意数据片段，而不必下载整个文件，大大减少了通过网络传输的数据量。

● 支持高效的数据访问。通过预读、副本等数据访问性能优化方法可以支撑高效的远程数据访问，而且简化了现有的显式数据迁移和部署过程。

今后笔者将进一步优化GVDS的广域数据访问性能，针对随机读方面存在的问题开展相关的优化方法研究工作，进而使GVDS系统能有效地支撑大型计算应用的高效访问和跨域协作。

## 参考文献:


[1] SLOTA R, DUTKA L, KRYZA B, et al. Storage systems for organizationally distributed environments-PLGrid plus case study[C]// International Conference on Parallel Processing and Applied Mathematics. Heidelberg: Springer, 2014: 724-733.

[2] XIE X, XIAO N, XU Z W, et al. CNGrid software 2: service oriented approach to grid computing[C]//The UK e-Science All Hands Meeting. [S.l.:s.n.], 2005: 701-708.

[3] DEPEI Q. CNGrid: a test-bed for grid technologies in China[C]//The 10th IEEE International Workshop on Future Trends of Distributed Computing Systems. Piscataway: IEEE Press, 2004: 135-139.

[4] Cluster File Systems, Inc. Lustre: a scalable, high-performance file system[Z]. 2002.

[5] WEIL S A, BRANDT S A, MILLER E L, et al. Ceph: a scalable, high-performance distributed file system[C]//The 7th Symposium on Operating Systems Design and Implementation. New York: ACM Press, 2006: 307-320.

[6] SCHMUCK F B, HASKIN R L. GPFS: a shared-disk file system for large computing clusters[C]//The Conference on File and Storage Technologies. New York: ACM Press, 2002: 231-244.

[7] DABEK F, KAASHOEK M F, KARGER D, et al. Wide-area cooperative storage with CFS[C]//The 18th ACM Symposium on Operating Systems Principles. New York: ACM Press, 2001: 202-215.

[8] DABEK F, BRUNSKILL E, KAASHOEK M F, et al. Building peer-to-peer systems with chord, a distributed lookup service[C]//The 8th Workshop on Hot Topics in Operating Systems. Piscataway:

IEEE Press, 2001: 81–86.

[9] GRAFFI K, GROSS C, STINGL D, et al. Lifesocial.KOM: a secure and p2p–based solution for online social networks[C]//2011 IEEE Consumer Communications and Networking Conference. Piscataway: IEEE Press, 2011: 554–558.

[10] CHAN Y W, HO T H, SHIH P C, et al. Malugo: a peer–to–peer storage system[J]. International Journal of Ad Hoc and Ubiquitous Computing, 2010, 5(4): 209.

[11] TOKA L, DELL'AMICO M, MICHIARDI P. Data transfer scheduling for P2P storage[C]//2011 IEEE International Conference on Peer–to–Peer Computing. Piscataway: IEEE Press, 2011: 132–141.

[12] SHEN H Y, LI Z, LI J. A DHT–aided chunk–driven overlay for scalable and efficient peer–to–peer live streaming[J]. IEEE Transactions on Parallel and Distributed Systems, 2012, 24(11): 2125–2137.

[13] 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]//The 29th Annual ACM Symposium on Theory of Computing. New York: ACM Press, 1997: 65–663.

[14] CALDER B, WANG J, OGUS A, et al. Windows Azure storage: a highly available cloud storage service with strong consistency[C]//The 23rd ACM Symposium on Operating Systems Principles. New York: ACM Press, 2011: 143–157.

[15] SIMMS C K, PIKE G G, BALOG D. Wide area filesystem performance using Lustre on the TeraGrid[R]. 2007.

[16] TATEBE O, HIRAGA K, SODA N. Gfarm grid file system[J]. New Generation Computing, 2010, 28: 257–275.

[17] THOMSON A, ABADI D J. CalvinFS: consistent WAN replication and scalable metadata management for distributed file systems[C]//The 13th USENIX Conference on File and Storage Technologies. New York: ACM Press, 2015: 1–14.

[18] WRZESZCZ M, TRZEPLA K, SOTA R, et al. Metadata organization and management for globalization of data access with OneData[C]//International Conference on Parallel Processing and Applied Mathematics. Cham: Springer, 2015: 312–321.

[19] DUTKA U, SOTA R, WRZESZCZ M, et al. Uniform and efficient access to data in organizationally distributed environments[M]//eScience on Distributed Computing Infrastructure. Cham: Springer, 2014: 178–194.

[20] GRIMSHAW A, MORGAN M, KALYANARAMAN A. GFFS–the XSEDE global federated file system[J]. Parallel Processing Letters, 2013, 23(2): 1340005.

[21] 胡正丁, 薛巍. 面向异构众核超级计算机的大规模稀疏计算性能优化研究[J]. 大数据, 2020, 6(4): 40–55.
HU Z D, XUE W. Research on performance optimization for large–scale sparse computation over many–core heterogenous supercomputer[J]. Big Data Research, 2020, 6(4): 40–55.

[22] KUNSZT P, BADINO P, FROHNER A, et al. Data storage, access and catalogs in gLite[C]//2005 IEEE International Symposium on Mass Storage Systems and Technology. Piscataway: IEEE Press, 2005: 166–170.

[23] FOSTER I, CZAJKOWSKI K, FERGUSON D E, et al. Modeling and managing state in distributed systems: The role of OGSI and WSRF[J]. Proceedings of the IEEE, 2005, 93(3): 604–612.

[24] CHERVENAK A, FOSTER I, KESSELMAN C, et al. The data grid: towards an architecture for the distributed management and analysis of large scientific datasets[J]. Journal of Network and Computer Applications, 2000, 23(3): 187–200.

[25] ALLCOCK W, BRESNAHAN J, BESTER J. GridFTP protocol specification[Z]. 2002.

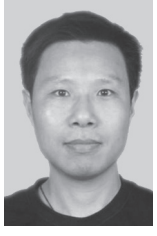
[26] FITZGERALD S, FOSTER I, KESSELMAN C, et al. A directory

service for configuring high-performance distributed computations[C]//The 6th IEEE International Symposium on High Performance Distributed Computing. Piscataway: IEEE Press, 1997: 365-375.
[27] CHEN M, BANGERA G B, HILDEBRAND D, et al. vNFS: maximizing NFS performance with compounds and vectorized I/O[J]. ACM Transactions on Storage, 2017, 13(3): 1-24.
[28] RYU J, LEE D, SHIN K G, et al. ClusterFetch: a lightweight prefetcher for intensive disk reads[J]. IEEE Transactions on Computers, 2017, 67(2): 284-290.
[29] WU Z, BUTKIEWICZ M, PERKINS D, et al. SPANStore: cost-effective geo-replicated storage spanning multiple cloud services[C]//The 24th ACM Symposium on Operating Systems Principles. New York: ACM Press, 2013: 292-308.
[30] SHAO Y, LI C, TANG H. A data replica placement strategy for IoT workflows in collaborative edge and cloud environments[J]. Computer Networks, 2019, 148: 46-59.
[31] CHANG W C, WANG P C. Write-aware replica placement for cloud computing[J]. IEEE Journal on Selected Areas in Communications, 2019, 37(3): 656-667.
[32] The reference implementation of the linux FUSE (filesystem in userspace) interface[Z]. 2019.
[33] CHOPADE R, DHAVASE N S. MongoDB, Couchbase: performance comparison for image dataset[C]//2017 2nd International Conference for Convergence in Technology. Piscataway: IEEE Press, 2017: 255-258.
[34] A network filesystem client to connect to SSH servers[Z]. 2019.
[35] SHEPLER S, NOVECK D, EISLER M. Network file system (NFS) version 4 minor version 1 protocol[Z]. 2010.
[36] Fio - flexible I/O tester rev. 3.16[Z]. 2019.

## 作者简介

**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,国家计算机科学技术名词审定委员会委员,国家科技基础条件平台专家组成员,工业和信息化部电子科学技术委员会委员,中国工程院中国信息与电子工程科技发展战略研究中心专家委员会特聘专家。主要研究方向为计算机体系结构、计算机软件系统、高性能计算、云计算、虚拟化技术等。先后获得国家科技进步奖二等奖、北京市科学技术奖一等奖、中国科学院科技进步奖一等奖、原信息产业部信息产业重大技术发明奖、科技部国家重点新产品奖等国家级和省部级科技奖励。

**宋尧**(1994- ),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、分布式调度系统、存算联动调度等。



**秦广军**(1977- ),男,博士,北京联合大学智慧城市学院讲师,CCF会员,主要研究方向为高性能计算、存储系统、大数据和机器学习等。作为项目骨干参与了国家863计划项目、国家重点研发计划项目、国家自然科学基金项目、北京市自然科学基金项目等。

**周汉杰**（1995– ），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。

**王超波**（1997– ），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、软件工程等。

**韦冰**（1990– ），男，北京航空航天大学计算机学院博士生，主要研究方向为网络存储、数据容错、大数据处理、分布式计算等。

**魏巍**（1975– ），男，博士，西安理工大学计算机科学与工程学院副教授，IEEE、CCF高级会员，*FGCS*、*AHSWN*、*IEICE*、*KSII*等期刊编委会成员，*IEEE TPDS*、*TVT*、*TIP*、*TMC*、*TWC*、*JNCA*和其他多个Elsevier期刊的定期审稿人。作为首席研究员和技术成员，主持了多项研究项目。主要研究方向为无线网络、无线传感器网络应用、图像处理、移动计算、分布式计算、普适计算、物联网、传感器数据云等。

**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，作为项目主持人和项目骨干，主持和参与了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。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。

**收稿日期：** 2021-01-21
**通信作者：** 秦广军，qingj@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)