软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2011,22(10):2372−2384 [doi: 10.3724/SP.J.1001.2011.03925]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

网格虚拟组织副本协作预取机制*

田 田+, 罗军舟, 宋爱波, 伍之昂

(东南大学 计算机科学与工程学院,江苏 南京 211189)

Cooperative Replica Prefetching Mechanism for Virtual Organization in Grid

TIAN Tian+, LUO Jun-Zhou, SONG Ai-Bo, WU Zhi-Ang

(School of Computer Science and Engineering, Southeast University, Nanjing 211189, China)

+ Corresponding author: E-mail: tian_tian@seu.edu.cn

Tian T, Luo JZ, Song AB, Wu ZA. Cooperative replica prefetching mechanism for virtual organization in grid. *Journal of Software*, 2011,22(10):2372−2384. http://www.jos.org.cn/1000-9825/3925.htm

Abstract: Replication is an effective method that improves data access in Data Grids; however, the improvement of the efficiency of repilication is a crucial problem. Previous works on the replication mechanism mostly select high value replicas to replicate, and simply choose the files that have been accessed based on the access records of files. This paper starts with the analysis of the file access characteristics in the virtual organization (VO). After introducing the concept of implicit high-value file (IHVF), it proposes the cooperative replica prefetching mechanism (CoRPM) for virtual organization, which is based on which local grid nodes can obtain the replicas of IHVFs through cooperation with other gird nodes in the same VO. The architecture of CoRPM is presented first, in which prefetching elements running on every gird node work cooperatively to provide a file prefetching service for all the grid nodes in the VO. Then, on the basis of the design of CoRPM, the process of CoRPM is described, whose core algorithms include a job-type centric file prefetching algorithm and a prefetching file selecting algorithm. In the end, the paper evaluates the performance of CoRPM through simulations, and the results show that CoRPM does reduce the file access latency more effectively.

Key words: data grid; replication; virtual organization; file prefetching; cooperative

摘 要: 副本复制是数据网格中提高数据访问效率的有效方法,如何提高副本复制的效率是一个关键性问题.现有的复制策略大多基于文件访问历史选择高价值副本进行复制,但其针对的都是节点已经访问过的文件.通过对虚拟组织文件访问特性进行深入分析,引入隐性高价值文件概念,提出虚拟组织副本协作预取机制(cooperative replica prefetching mechanism,简称 CoRPM),使得本地节点通过与虚拟组织中其他节点进行协作来获取隐性高价值文件副本.该机制首先给出了副本协作预取架构,各个虚拟组织节点上的文件预取模块以协作的方式为虚拟组织内节点提供文件预取服务;然后,在副本协作预取架构的基础上设计了副本协作预取流程,其核心算法包括以作业类型为中心的本地文件预取算法和预取文件选择算法.模拟实验结果表明,CoRPM 与已有的基于文件访问历史的副本复制策略相结合,可以更加有效地降低数据访问延迟.

* 基金项目: 国家自然科学基金(60903161, 60903162, 61070161, 61003257, 61103229); 国家重点基础研究发展计划(973)(2010CB328104); 高等学校博士学科点专项科研基金(200802860031); 江苏省自然科学基金(BK2008030); 江苏省网络与信息安全重点实验室(BM2003201); 东南大学计算机网络和信息集成教育部重点实验室(93K-9)

收稿时间: 2009-11-23; 定稿时间: 2010-07-16

关键词: 数据网格;副本复制;虚拟组织;文件预取;协作
中图法分类号: TP393 **文献标识码:** A

数据网格是一种面向数据密集型应用的网格平台,它聚集了大量的计算资源、存储资源和网络资源,确保数据密集型应用的顺利执行,同时对数据密集型应用所产生的海量数据进行存储和管理[1−3].数据网格通常利用虚拟组织(virtual organization,简称 VO)实现分布式资源的发现和共享[4],虚拟组织是由个人和自治域为了共同的目标按照一定的资源共享规则形成的集合.

当虚拟组织中的计算节点接受用户提交的作业请求时,若作业计算需求的文件不在本地,就需要从其他节点获取副本并传输到本地.在这种情况下,数据传输的快慢决定了作业执行时间的长短,文件过大或者网络带宽过小都容易引起数据传输时间过长,这无疑将延长作业执行时间.因此,为了降低数据访问延迟,数据网格通常采用副本复制的方式将文件以多副本的形式分布存储于多个节点.然而,网格节点的存储资源是有限的,只能存储数量有限的文件副本,因此需要尽可能地复制价值较高的副本.副本的高价值指的是副本在未来一段时间内具有较高的访问次数.目前的数据网格副本复制策略大多通过查询文件访问历史记录来决定存储哪些文件的副本[5−7].一些研究工作提出基于经济模型的复制策略,节点通过对自身已有的文件访问记录和对未来的预测来决定创建副本的内容[5].然而,上述所有策略都是基于文件访问历史的,即副本复制的所有候选者都是在本地节点被访问过或者正在访问的文件,经济模型中对未来的预测也只是针对访问过的文件来预测它们在未来一段时间内的访问需求.但是,还有一部分文件是本地节点从未访问过的,但同样可能在未来被多次访问,本文称这类文件为隐性高价值文件.基于文件访问历史的副本复制策略并没有考虑隐性高价值文件的存在对副本复制的影响,其针对的是具有高价值且本地节点已经访问过的文件,具有一定的滞后性.

虚拟组织的重要特性就是其成员具有共同目标[4],例如某个虚拟组织专用于蛋白质序列分析,也有虚拟组织专用于空间粒子探测.文献[8]指出,兴趣相同的用户倾向于访问相同的或相关性较大的文件.因此,同属于一个虚拟组织的成员所提交的作业更容易访问相同的或相关性较大的文件.文献[9]已经表明,在面向高能物理科学计算的网格环境中,确实存在对相关性较大的文件聚集请求的特性.由于虚拟组织中每个节点都可以接受来自该虚拟组织不同成员提交的作业请求,因此,当某个节点收到一个文件访问请求后,该节点在随后一段时间内很可能收到更多与此文件相关性较大的文件访问请求.这些相关文件无疑具有较高的价值,而且由于本地节点还未体现出对它们的需求,所以它们是隐性高价值文件.显然,若将这些隐性高价值文件的副本预取到本地,将来需要访问这些文件时就可以直接从本地读取,从而降低数据访问延迟.然而,由于隐性高价值文件在本地节点还未体现出访问需求,没有访问记录,因此很难通过查询文件访问记录来确认.目前,很少有研究工作给出虚拟组织中隐性高价值文件的获取方式.为此,本文针对虚拟组织的特点及其文件访问特性,提出虚拟组织中的副本协作预取机制(cooperative replica prefetching mechanism,简称 CoRPM),将副本协作预取与已有的基于文件访问历史的副本复制策略相结合,以进一步降低数据访问延迟.本文的贡献主要包括以下几个方面:

(1) 分析虚拟组织的文件访问特性,以说明虚拟组织中确实存在隐性高价值文件;
(2) 提出虚拟组织副本协作预取架构,各个虚拟组织节点上的文件预取模块以协作的方式为虚拟组织内节点提供文件预取服务;
(3) 在上述提出的副本协作预取架构的基础上,设计了副本协作预取流程,其核心算法包括以作业类型为中心的本地文件预取算法和预取文件选择算法.

本文第 1 节介绍网格环境下副本复制和预取的一些相关工作.第 2 节详细介绍虚拟组织的文件访问特性.在此基础上,第 3 节提出虚拟组织副本协作预取架构.第 4 节给出具体的副本协作预取流程.第 5 节模拟实验并分析结果.第 6 节总结当前工作.

1 研究现状

副本复制主要解决 3 个问题:(1) 副本复制的内容;(2) 副本复制的触发时间;(3) 副本的存储地点.确定副本

复制的内容,是解决问题(2)和问题(3)的前提.文献[5]提出一种基于经济模型的副本复制模型,该模型对所复制副本的价值进行预测,提出确定副本复制内容的基本原则:优先复制高价值的副本.本文提出副本协作预取机制的目的,也是为了获得隐性高价值文件副本.

预取技术被应用到许多领域,并取得了良好的效果,其中颇具代表性的领域是文件预取[10]和 Web 预取[11],两者存在着明显的差别:在 Web 预取中,大部分文件以网页内容形式存在,文件较小,网页间的相关程度可以通过网页之间的链接进行计算,比如 Google 的 PageRank 算法.但是在文件预取中,由于文件之间没有任何链接,且常常以非结构化的形式存在,使得文件间相关性的计算较为困难,影响了文件预取的准确率.

文献[12]采用 VSM(vector space model)模型和 LSI(latent semantic indexing)技术从文件内容上来判定文件间的相关性,然而这种方式需要消耗大量存储资源和计算资源,并消耗大量时间.文献[13]提出通过文件标识符之间的差值来计算文件的相关性,这种方法试图根据已经访问文件的标识符的分布,预测未来可能要访问文件的标识符.然而在真实系统中,随机产生的文件标识符难以反映文件的相关性.

文件预取算法已经得到了广泛的研究,然而已有的文件预取算法大多没有考虑文件动态流行度对预取性能的影响[11].文件的流行度在它的生命期内是动态变化的,文件预取的对象应该是近期较为流行的文件,本文提出基于动态支持度的预取规则来解决这个问题.另外,本文还针对网格下不同作业类型文件访问的特点,提出以作业类型为中心的文件预取算法,提取出适合该特定作业类型的预取规则以提高预取准确率.

虚拟组织中的用户对文件访问具有相似的兴趣,更容易访问内容相似或相关性较大的文件,这种文件访问特点可以通过文献[6]提出的网格下文件访问的局部性特征而得到体现.据此,本文将在第 2 节分析虚拟组织文件访问的局部性特征.目前,网格环境下文件预取方面的研究还不是很多,文献[14]研究内存网格的内存预取技术,它启发了本文的工作.但是,本文研究的是数据网格虚拟组织中的文件预取;更明显的区别是,本文的文件预取是协作式的,本地节点通过与虚拟组织中其他节点进行协作来获取隐性高价值文件副本.

2 虚拟组织文件访问特性

虚拟组织中的每个网格节点在功能上是对等的,都可以贡献计算资源(computing element,简称 CE)和存储资源(storage element,简称 SE).存储资源用于存放数据,目前成熟的网格数据管理方法都是基于文件的.虽然副本目录的组织方式既可以是全局目录,也可以是分布式目录,但无论目录组织方式如何,在一个虚拟组织中,都需要提供副本定位服务来定位文件的实际存储位置,副本定位服务会根据请求节点的情况(包括带宽、地理因素等)择优返回副本存储地址.一旦有新副本建立时,需要通过相应的副本注册服务进行注册.

由于虚拟组织是由个人和自治域为了共同的目标协作建立的,因此虚拟组织中的成员对文件具有相似的兴趣,有更大的可能性去访问相关性较大的文件.文件的相关性既可以体现在文件内容的相似上,也可以通过特定的文件访问方式来反映.例如,一些总是被集体访问的文件就有很大的相关性.虚拟组织中这种倾向于访问相关性较大文件的特点可以通过文献[6]提出的一些文件访问的局部性特征体现出来,包括:

(1) 时间局部性:最近被访问过的文件可能被再次访问;
(2) 地理局部性:也可称其为用户局部性,是指一个用户最近访问的文件可能被与该用户邻近的用户再次访问;
(3) 文件局部性:也可称其为空间局部性,即与刚被访问的文件邻近的文件很可能会被访问.

时间局部性已经在很多的缓存管理或副本管理策略中得以运用,典型的如 LRU 算法.地理局部性在虚拟组织中可以得到很好的体现.两个用户是否是“邻近”用户并不是由他们之间的物理跳数或者物理连接距离决定的,而是由他们是否具有相似的兴趣来决定.因此,邻近用户是一个具有特殊兴趣的成员集合,集合中的成员可能是地理分布的,但是在逻辑上却由于相似的兴趣而“聚”在一起.比如,P2P 中的“swarm”就是由下载相同的一个或多个文件的所有下载者组成的一个集合[15].虚拟组织也是因为其成员具有相似的兴趣而建立的,因而虚拟组织中的成员可以看成是邻近用户,被虚拟组织中某个用户访问过的文件很可能被其他邻近用户访问.而从网格节点的角度来看,当一个文件在某个节点被某个虚拟组织用户访问过后,该文件在未来很可能会在其他节点上

被其他虚拟组织用户所访问.

文件局部性的关键在于判定文件间的“邻近”关系,“邻近”不是物理上的邻近,即存储介质上存储地址的邻近,而是指文件间有较大的相关性.文献[13]根据文件的 ID 号来判定文件的相关性,认为两个文件的整数 ID 号越相近,相关性就越大.然而真实系统中的文件标识符大多随机产生,不仅包括数字,还包括字母,很难反映文件的相关性.本文提出一种新的判定“邻近”文件的规则:在一个虚拟组织中,相关性较大的文件很可能被用户连续访问,而连续访问的文件在文件访问序列中必然位置靠近,由此定义“邻近”文件为文件访问序列中位置邻近的文件.

为了下文描述方便起见,首先给出两个定义:

定义 1(显性高价值文件 EHVF(explicit high-value file)). 具有较高价值,且在本地节点已经有访问需求的文件.

定义 2(隐性高价值文件 IHVF(implicit high-value file)). 具有较高价值但目前还未在本地节点体现出访问需求(即本地节点从未访问过)的文件.本地节点的所有隐性高价值文件组成的集合称为隐性高价值文件集合(implicit high-value files set,简称 IHVFS).

显然,EHVF 的存在利用了文件访问的时间局部性,优先复制 EHVF 副本就是基于文件访问历史副本复制策略的核心所在.在复制 EHVF 副本的基础上,为了进一步降低本地数据访问延迟,本文利用协作式文件预取获取 IHVF 副本,而虚拟组织中,3 种文件访问局部性特征为本地节点获取 IHVF 副本提供了可能.例如,虚拟组织中某节点 a 最近一段时间经常收到文件请求 f,若文件 f 不在本地,节点 a 就需要从远程节点获取 f 的副本.若文件 f 请求次数超过了阈值,则触发副本复制.根据时间局部性,f 显然是 EHVF.不仅如此,由于文件局部性的存在,节点 a 在未来一段时间内可能会收到更多与文件 f 相关性较大的文件访问请求.因此,节点 a 在获取文件 f 的副本时,可以引进一些文件 f 的“邻近”文件的副本,进一步提高未来一段时间内文件访问请求的响应速度.未存储在本地节点的文件 f 的“邻近”文件属于 IHVFS.IHVFS 中的文件在本地节点从未被访问过,很难通过本地文件访问记录预测并确认,因此,对这些“邻近”文件的获取是一个难题.本文引入地理局部性来解决这个问题,在一个虚拟组织中,文件 f 虽然在本地没有被访问过,但很可能在其他远程节点已被访问过;而且由于文件局部性的存在,这些远程节点可能储存了文件 f 的“邻近”文件.

综上所述,对于不在本地的文件访问请求,本地节点在获取该文件副本时,可以主动地与其他远程节点进行协作,选择性地预取该文件的“邻近”文件副本,以进一步提高未来一段时间内文件请求的响应速度.据此,本文在第 3 节提出了虚拟组织副本协作预取架构,给出了文件预取功能模块的组成和协作方式.第 4 节则详细阐述了副本协作预取流程及其核心算法.

3 虚拟组织副本协作预取架构

本节提出如图 1 所示的虚拟组织副本协作预取架构.图 1 左侧是一个包含多个网格节点的虚拟组织,每个虚拟组织节点可以抽象为 CE 和 SE 两部分.PE(prefetching element)是文件预取功能模块,PE 作为一个独立的功能模块运行在 SE 之上.然而,各个虚拟组织节点上的 PE 之间并不是相互独立的,而是以一种协作的方式为虚拟组织内节点提供文件预取服务.某个节点的 PE 可以向单个或者多个节点的 PE 发送文件预取请求 FPR(file prefetching request),相关节点接受请求后,运行文件预取算法并将预取文件信息发送回预取请求节点,请求节点则根据收到的预取文件信息选取一部分文件进行预取.

图 1 右侧所示为 PE 的组成模块图,PE 由 3 个子模块组成,包括文件访问历史数据库、预取规则库、预取处理引擎.文件访问历史数据库记录了所在节点所有的文件访问历史记录,它可以以文件的形式存在.随着时间的增多,该数据库的数据量将会越变越大,可以采用合适的数据压缩算法来降低其存储空间.预取规则库对文件访问历史数据库中的历史数据进行预处理,得到文件访问模式,并从文件访问模式中提取出文件预取规则,只有符合预取规则的文件才可以作为预取对象.预取处理引擎负责向其他节点发出文件预取请求,并接收返回的预取文件信息,同时也接受其他节点的文件预取请求并返回相关的预取文件信息.预取处理引擎的核心是本地文

件预取算法和预取文件选择算法.

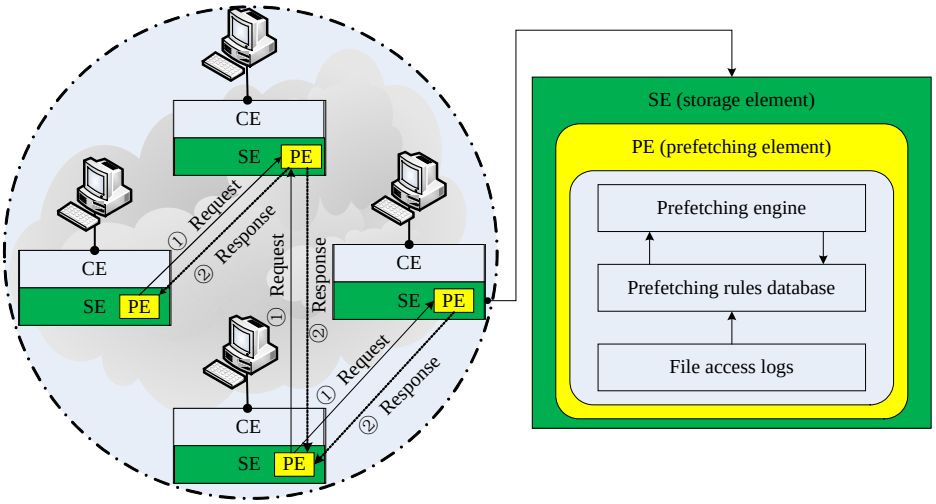


Fig.1 Architecture of cooperative replica prefetching mechanism

图 1 副本协作预取机制架构

4 副本协作预取流程

副本协作预取流程通过各个虚拟组织节点上的 PE 模块协作实现,其核心算法包括独立运行在各个节点上的本地文件预取算法和预取文件选择算法.本文提出的本地文件预取算法具有两大优点:(1) 考虑文件动态流行度对预取性能的影响;(2) 针对不同作业类型文件访问的特点,提出以作业类型为中心的预取算法,以进一步提高预取效率.本节首先给出本地文件预取算法,随后给出预取文件选择算法,最后给出副本协作预取流程.

4.1 基于动态支持度的预取规则

本地文件预取算法接收其他节点的文件预取请求,通过查询本地预取规则库决定需要预取哪些文件,并将这些文件的相关信息返回给预取请求节点.

预取规则库对本地存储的文件访问历史记录进行预处理形成文件访问模式,再从文件访问模式中提取出预取规则.采用与文献[14]类似的方法来生成文件访问模式.每个虚拟组织节点在一段时间内都会形成一个文件请求序列,记为$(f_1,f_2,\ldots,f_n)$,$f_i(1\leqslant i\leqslant n)$是文件唯一标识符,同时记录文件 f_i 的访问时间 $t_i(1\leqslant i\leqslant n)$.定义连续文件访问请求:设定阈值 t_s,则文件请求序列中任何小于 t_s 的两个相邻的文件访问请求是连续文件访问请求.如果两个相邻的文件访问请求其间隔时间大于等于 t_s,则将序列一分为二,即当 $t_{i+1}-t_i\geqslant t_s$ 时,由于 f_{i+1} 与 f_i 已经不属于连续文件访问请求,则将序列$(f_1,f_2,\ldots,f_n)$分为两个子序列:$(f_1,f_2,\ldots,f_i)$和$(f_{i+1},f_{i+2},\ldots,f_n)$.经过上述预处理后的文件请求序列被称为文件访问模式,可以看作是一段时间内对于文件的连续访问请求,能够反映出一定的文件关联性.若生成相同的文件访问模式,需要记录其在文件访问历史记录中出现的总次数.

文件访问模式形成后,需要从中提取出预取规则.目前,主流的预取方法包括基于 Markov 模型的预取算法和基于数据挖掘的预取方法.前者的典型代表有基于高阶 PPM 模型的算法[11],后者则有 CloSpan[16]等.这些预取方法准确率较高,但预取模型复杂且时间复杂度较高.为简化起见,本文在结果可接受的情况下提出一种轻量级的预取规则提取方式,在确保本地文件预取算法具有较低执行时间的同时,还降低了本地节点计算资源和存储资源的耗费.不仅如此,本文提出的本地文件预取算法还具有本节开头所描述的两个优点,即考虑了文件动态流行度对预取性能的影响,并针对不同作业类型文件访问的特点提出以作业类型为中心的预取算法.

假设已经给定文件访问模式$(f_1,f_2,\ldots,f_n)$,定义从中提取的预取规则如下.

定义 3(预取规则 PR(prefetching rule)). $PR=\{f_i\&f_j\rightarrow f_k\}$,满足 $1\leqslant i<j<k\leqslant n,(j-i)\leqslant m_1$,且$(k-i)\leqslant m_2$.

对于每条预取规则 $PR=\{f_i\&f_j\rightarrow f_k\}$,$f_i\&f_j$ 称为预取规则的前缀;f_k 称为预取规则的后缀,其含义是:若已有连续文件访问请求 f_i 和 f_j,则 f_k 是下一个要访问的文件.例如,若有文件访问模式$(abcd)$,取 $m_1=2,m_2=4$,则提取出预取规则$(ab\rightarrow c),(ab\rightarrow d),(ac\rightarrow d),(bc\rightarrow d)$.此外,预取规则的支持度指的是从已有所有文件访问模式中提取出该预取规则的总次数.

我们注意到,虚拟组织中的用户对文件具有相似的兴趣.但随着时间的流逝,其兴趣可能会发生变化.兴趣的变化使得文件流行度也发生动态变化,文件预取的对象应当是近期具有较高流行度的文件,一些过去较为流行但近期访问较少的文件不适合作为文件预取的对象.已有的文件预取算法大多没有考虑文件流行度的动态变化,本文通过计算预取规则动态支持度来反映文件动态流行度对预取的影响.文件的流行度可以表示为一段时间内该文件总的访问次数,流行度越大的文件,总访问次数越多,生成的文件访问模式中出现该文件的次数也越多,可能导致最终提取出的与该文件相关的预取规则支持度也越大;反之亦然.我们在每个本地节点设立时间阈值 t_o,对应每个预取规则 PR,分别记录最近支持度 S_{recent} 和过去支持度 S_{old}.设当前时间点为 t_{cur},S_{recent} 记录了距 t_{cur} 时间点过去 t_o 时间内该 PR 出现的次数,S_{old} 记录了距 t_{cur} 时间点 t_o 时间以外该 PR 出现的次数.那么,PR 的逻辑支持度 S_{total} 的计算公式如下:

$$S_{total}=\alpha\cdot S_{recent}+(1-\alpha)\cdot S_{old},\alpha\geqslant 0.5 \tag{1}$$

$\alpha=0.5$ 时,表示最近支持度 S_{recent} 和过去支持度 S_{old} 对逻辑支持度的影响所占的权重一样大,即不采用基于动态支持度的预取规则;$\alpha>0.5$ 时,则表示逻辑支持度受最近支持度的影响较大,且α越大,表明逻辑支持度受最近支持度的影响越大.逻辑支持度更好地反映了文件的动态流行度,对于具有相同前缀但后缀不同的预取规则,按照逻辑支持度大小降序排列,逻辑支持度越大,文件预取的价值越高.图 2 描述了具体的预取规则,每条预取规则都对应记录了 S_{total},S_{recent} 和 S_{old},并且 $S_{total}\{ab\rightarrow r\}>S_{total}\{ab\rightarrow t\}>S_{total}\{ab\rightarrow m\}$.此外,逻辑支持度需要及时更新,更新策略如下:每个节点每隔一段时间 $t_u(t_u<t_o)$,根据原有的 S_{recent} 和 S_{old} 值以及 t_u 时间内该预取规则新出现的次数,计算新的 S_{recent} 和 S_{old},最后按照公式(1)计算新的 S_{total};在删除预取规则或者提取出新的预取规则时,也要重新对与此预取规则前缀相同的所有预取规则按逻辑支持度大小降序排列并及时更新.

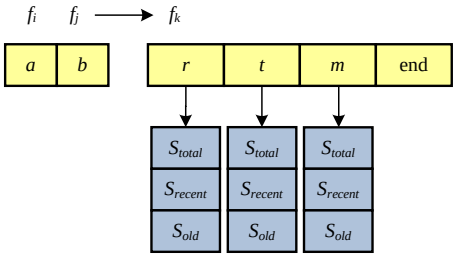


Fig.2 Data structure of prefetching rules

图 2 预取规则的数据结构

在引入预取规则动态支持度后,本地文件预取算法可以简单描述为:接受预取请求 $f_i\&f_j$ 后,查找以 $f_i\&f_j$ 为前缀的逻辑支持度前 N 的预取规则,返回所有预取规则的后缀.显然,预取规则的逻辑支持度越大,对应的文件被预取的可能性就越高.本文若无特殊说明,取 $N=2$.

4.2 作业类型为中心的本地文件预取算法

除了考虑文件动态流行度对预取性能的影响,本地文件预取算法还必须考虑虚拟组织中不同作业类型文件访问的特点,以提高算法的适应性.虚拟组织环境中,多个作业可以同时运行在一个节点上,这些作业按照诸如输入参数或计算目标对象不同等划分标准,可以划分为不同的作业类型,不同作业类型的作业对文件的访问需求不尽相同,表现在其经常访问的文件集合是不同的.即使是同一应用的作业,也可能属于不同的作业类型,而同属于一个作业类型的作业只可能访问与该作业类型相关的文件集合中的一个或多个文件.例如,东南大学

校园网格 SEUGrid 中就部署有用于空间粒子探测的 GEANT4 程序,它可以模拟不同粒子经过探测器时与探测器的相互作用情况,生成模拟数据以进行事件重建[17,18].用户提交不同粒子的重建作业需要访问不同的模拟数据,所有氦核重建作业和所有氢核重建作业显然是属于两个不同的作业类型.然而目前,单个虚拟组织节点上同时运行的所有作业的文件访问记录都混合记录在 SE 上的文件访问历史数据库中,这种混合的历史访问记录降低了预取规则建立的准确率,如图 3 所示.图 3(a)显示了两个作业 *Job*1 和 *Job*2 各自的文件访问记录以及 SE 的文件访问记录,设 *Job*1 的作业类型是 *Jtype*1,*Job*2 的作业类型是 *Jtype*2,*Job*1 和 *Job*2 同时运行在同一个节点上,导致节点 SE 上记录的文件访问记录是两者的混合.若以 SE 上记录的文件访问记录提取预取规则,取 m_1=3,m_2=6,提取出预取规则(24→567),(13→4567)等,如图 3(c)所示.当节点收到预取请求(1&3)时,则返回 4 和 5.而事实上,4 和 5 在 *Jtype*2 中从未访问过.

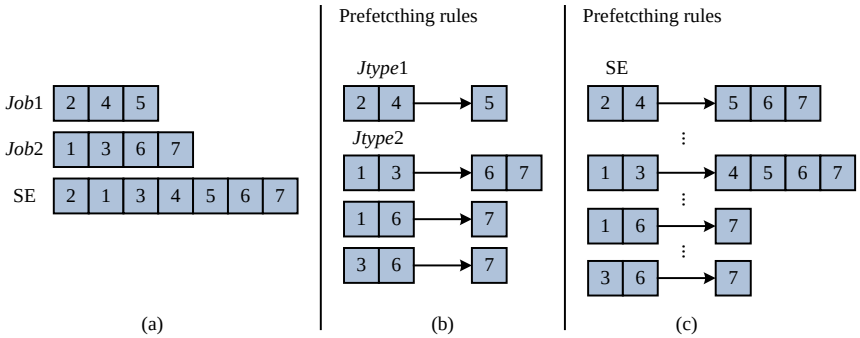


Fig.3 Prefetching rules under different situations

图 3 不同提取方式下的预取规则

本节提出一种以作业类型为中心的本地文件预取算法 $JCFP_{local}$(job-type centric file prefetching),其核心思想是,对存储在 SE 上文件访问历史数据库中的文件访问记录按作业类型分类,将同一个作业类型访问的文件历史数据聚合到一起,形成该特定作业类型下的文件访问模式,进而采用第 4.1 节中的方法提取出适合该特定作业类型的预取规则.如图 3(b)所示,在 $JCFP_{local}$ 中,当节点收到预取请求(1&3)时,会返回 6 和 7,而不是返回 4 和 5.本地文件预取算法 $JCFP_{local}$ 如算法 1 所示.

算法 1. 本地文件预取算法 $JCFP_{local}$(*prefix*1,*prefix*2,*Jtype*).

输入:预取规则库 *L*,文件预取请求 *FPR*=[(*m*,*n*),$Jtype_{id}$],$Jtype_{id}$ 为作业类型;

输出:预取文件对象数组 *PF*[·].

描述:

1. *prefix*1←*m*, *prefix*2←*n*, *Jtype*←$Jtype_{id}$;
2. *L*′=*Find*(*L*,*Jtype*);
3. **if** (*prefix*2!=*null*)
4. **then** *prefix*←*prefix*1+*prefix*2;
5. *PF*[*N*]=*FilePrefetch*(*prefix*,*L*′,*N*);
6. **return** *PF*[*N*];
7. **else** /*只有一个前缀*/
8. **then** *prefix*←*prefix*1;
9. *prefix*2[*k*]=*PrefixFind*(*prefix*);
10. **for** *i*←1 **to** *k*
11. **return** *prefix*2[*i*]+$JCFP_{local}$(*prefix*1,*prefix*2[*i*],*Jtype*);

算法 1 第 2 行 *Find* 函数用于找出作业类型为 *Jtype* 的所有预取规则.第 3 行~第 6 行针对的是文件预取请求有两个前缀的情况,即 *FPR*=[(*m*,*n*),$Jtype_{id}$].其中,第 5 行返回 *L*′中以(*m*,*n*)为前缀的逻辑支持度前 *N* 大的文件

对象,每个文件对象包含标识符以及逻辑支持度等信息.第 7 行~第 11 行针对的是文件预取请求只有一个前缀的情况,即 $FPR=[(m,*),Jtype_{id}]$.此时,对于预取规则 $PR=\{f_i\&f_j\rightarrow f_k\},f_i=m$,首先需要确定可能的 f_j,然后针对每个 $f_i\&f_j$,再确定 f_k;算法第 9 行 *PrefixFind* 函数用于查找可能的 f_j,第 11 行依次递归调用 $JCFP_{local}$ 函数,最后将得到的预取文件对象 f_k 和之前的 f_j 一起返回.

4.3 副本协作预取流程

节点一旦接收到来自其他节点的文件预取请求,立即启动本地文件预取算法,并将算法返回的预取文件信息以(f_{id},*Support*)的形式发送给发出预取请求的节点.其中,f_{id} 是网格唯一文件标识符,*Support* 是该文件对应预取规则的逻辑支持度.而发出预取请求的节点在接收到来自不同节点的预取文件信息后,会执行预取文件选择算法,筛选出相对较优的文件以启动副本复制过程.具体过程简述如下:本地节点接收到所有远程节点返回的预取文件信息后,对相同 f_{id} 的文件进行逻辑支持度求和运算;然后对所有 f_{id} 按照逻辑支持度大小降序排列,选取前 K 个文件作为预取对象,获取副本并存储于本地预取池中.预取池是 SE 上一个独立的存储空间,用于存放预取来的文件副本,其大小可以根据不同节点 SE 的空间大小进行配置.由于预取池存储空间有限,采用 LRU 算法进行管理.预取文件选择算法 PFileSelect 如算法 2 所示.

算法 2. 预取文件选择算法 *PFileSelect*(*K*).

输入:各个节点传回的预取文件信息集合 $PF^n[\cdot]$,n 表示有 n 个节点返回了预取文件信息;

输出:需要预取的文件集合 $PFile[\cdot]$,$|PFile[\cdot]|\geqslant 0$.

描述:

1. **for** every f_{id}, *Support*←0; /*对相同 f_{id} 逻辑支持度求和*/
2. **for** i←1 **to** n
3. **if** $PF^i[\cdot]$ has this f_{id}
4. **then** $Support=Support+PF^i.f_{id}.Support$;
5. $PFile[\cdot]=TopSelect(K)$; /*返回逻辑支持度前 K 个大的预取文件对象*/

在给出本地文件预取算法和预取文件选择算法的基础上,副本协作预取流程描述如下:

1. 本地节点收到某个作业对文件 a 的访问请求,其作业类型用 $Jtype_{id}$ 表示;若 a 不在本地,本地节点通过副本定位服务获得所有拥有文件 a 副本的远程节点地址;
2. 本地节点向远程节点发出文件预取请求:
   (1) 若文件 a 之前有连续文件访问请求 b,则向所有查询到的拥有文件 a 副本的远程节点发出文件预取请求 $FPR=[(b,a),Jtype_{id}]$;
   (2) 若文件 a 之前无连续文件访问请求,则向所有查询到的拥有文件 a 副本的远程节点发出文件预取请求 $FPR=[(a,*),Jtype_{id}]$;
3. 本地节点等待所有远程节点的回复;
4. 远程节点预取处理引擎接收文件预取请求 FPR:
   (1) 若 $FPR=[(b,a),Jtype_{id}]$,调用 $JCFP_{local}(b,a,Jtype_{id})$;
   (2) 若 $FPR=[(a,*),Jtype_{id}]$,调用 $JCFP_{local}(a,null,Jtype_{id})$;
5. 远程节点将预取文件信息返回给本地节点;
6. 本地节点等待所有远程节点的回复,M 时间后调用预取文件选择算法 PFileSelect,然后启动副本复制获取副本并存储于本地预取池中.

由流程第 1 步、第 2 步可以看出,副本协作预取发生在某个文件请求不能为本地节点所满足的时候,对于能够满足的文件请求则不触发协作预取,这是因为:(1) 若针对每个文件请求都触发协作预取,则需要消耗大量的存储资源和网络资源,甚至会影响节点间原有的文件传输速度,反而加大了本地数据访问延迟;(2) 若请求的文件已在本地,则无需进行副本复制,此时若要启动协作预取就需要额外调用副本定位服务,加重了计算资源的消耗,副本定位服务的频繁调用也容易造成该服务负担过重,从而导致响应速度降低.对于整个副本协作预取而

言,其协作性主要体现在隐性高价值文件的获取上,即由于本地节点无法单独确定隐性高价值文件,因此必须与虚拟组织中其他节点进行协作,通过获取其他节点的预取文件信息来确定隐性高价值文件并获取副本.

本文提出的虚拟组织副本协作预取机制还具有以下两个优点:(1) 不依赖于副本目录的组织形式,本地节点只需通过调用虚拟组织内的副本定位服务来确定需要发送文件预取请求的所有远程节点地址,进而启动协作预取,提高了该机制在具有不同副本目录组织结构的虚拟组织环境下的适应性;(2) 副本协作预取架构采用的是分布式架构,各个节点都拥有独立的 PE 模块,只有在接受文件预取请求时才与其他节点交换预取文件信息,网络通信开销较少,且易于实现和部署.

5 实验分析

为了分析副本协作预取机制的性能,我们采集了来自实际网格系统的作业运行日志,模拟网格作业从提交作业开始到发出文件请求以及文件请求被响应后的执行过程.本节通过对预取命中率和作业平均响应时间等指标的测量,表明采用本文提出的副本协作预取机制可以更加有效地降低网格虚拟组织中的数据访问延迟.

5.1 实验设置

SEUGrid 是东南大学专为 AMS 实验搭建的一个网格平台[19].AMS 实验是一个大型的国际合作项目,其目的是为了寻求宇宙空间中的暗物质和反物质.SEUGrid 目前由 3 个独立的虚拟组织组成,其中一个用于蒙特卡洛(Monte Carlo,简称 MC)仿真实验,用 VO_{MC} 表示,它是 AMS-02 探测器发射升空前最主要的用途;其余两个虚拟组织分别用于数据处理和数据分析.VO_{MC} 目前拥有 8 个网格节点,其中计算能力较强的两个节点是 IBM 集群和 DELL 集群,两个集群各自有 8 个刀片服务器用于 MC 仿真,其余的网格节点由 PC 组成.

本文模拟实验的数据都采集自 VO_{MC}.我们采集了 3 周 15 个工作日 VO_{MC} 中所有节点上 MC 仿真作业的运行记录,经过处理取得 4 982 条有效记录,每条记录包含作业的运行节点、作业请求的文件、请求文件大小等参数.模拟器模拟每条作业记录的运行过程,副本定位服务则通过调用 SEUGrid 提供的副本定位服务接口来实现.为简单起见,模拟器设定任意两个节点间是连通的,且带宽为 100Mbps,每个节点的存储资源按照节点实际提供数值设定,其中两个集群构成的节点分别提供 1TB 的存储资源,而每个 PC 机构成的节点则只提供 20GB 的存储资源.实验开始时,所有文件分布存储在各个节点之上,两个集群节点相对于其他 PC 节点会多存储一些文件,但并不存储所有文件或者是大部分文件的副本.若文件请求大部分都可以在本地被满足,则会降低该节点与其他节点相互协作建立副本的可能.提取预取规则时设定 m_1=2,m_2=4,t_s=0.5 小时,t_o=24 小时,t_u=8 小时.MC 仿真程序有多种作业类型,分为氦核计算、氢核计算等,我们通过在请求文件时向文件系统传递一个参数用以表明作业类型.我们将采集到的第 1 周的文件访问记录按照作业类型进行分类,在此分类的基础上提取预取规则,生成预取规则库;而后两周的记录则用作测试集.需要说明的是,由于节点计算能力的不同,计算能力越强的节点被分配的作业相对就越多,请求的文件也较多.因此,由集群构成的两个网格节点上所采集到的文件访问历史记录比其他由 PC 构成的网格节点要多.

5.2 结果及分析

首先,我们定义两个性能指标:

(1) 预取命中率(hit ratio,简称 HR):这是指预取得到的文件副本被请求的总次数占文件请求总数的百分比.

$$HR = \frac{PF_{req}}{TF_{req}} \tag{2}$$

其中,PF_{req} 表示预取得到的文件副本被请求的总次数,TF_{req} 表示文件请求总次数.HR 越大,表示预取效果越好.

(2) 作业平均响应时间(average response time,简称 ART):作业响应时间指的是作业从提出文件请求开始直到获得所有所需文件的时间.如果作业请求的文件存储在本地,则可认为响应时间小到可以忽略不计,模拟器中用 0 表示;反之,则需要从其他节点获取所需文件的副本.这时,文件传输时间占据了响应时间的绝大部分,模拟器中设定这种情况下的响应时间即为文件传输时间,文件传输时间可用文件大小比上节点间带宽得到.所有作

业的平均响应时间称为作业平均响应时间.

实验 1 考察文件动态流行度对预取性能的影响.由于 DELL 集群上收集的文件访问历史数据较多,更有利于实验分析,因此选择 DELL 集群上收集的文件访问历史数据进行模拟.我们用 $PFile_1$ 表示该节点上测试集开始第 1 天通过预取得到的所有文件副本集合,以此类推,$PFile_2$ 和 $PFile_3$ 分别表示第 2 天和第 3 天预取得到的所有文件副本集合.图 4 显示的是 $PFile_1$,$PFile_2$,$PFile_3$ 在随后 7 天内的预取命中率.以 $PFile_1$ 曲线为例,分别记录了随后 24 小时内、48 小时内直至 168 小时内所有请求 $PFile_1$ 中文件的请求数占所有文件请求数的百分比.实验中,公式(1)取 α=0.5,PFileSelect 算法取 K=3.由图 4 可以看出:$PFile_1$ 的预取命中率逐渐升高,并在第 3 天达到峰值,之后便逐渐下降;$PFile_3$ 的预取命中率也是先上升后下降;$PFile_2$ 则稍有区别,预取命中率在达到峰值下降后又稍有回升,但最终还是趋于下降.上述曲线反映出动态变化的文件流行度确实对预取性能有所影响,预取命中率增大是因为在一段时间内针对该文件集合中的文件请求次数增加较多,即文件流行度增大;反之亦然.

实验 1 已经表明,预取算法性能受到文件动态流行度的影响.因此在实验 2 中,我们考察采用基于动态支持度的预取规则对预取算法性能的影响.同样选择 DELL 集群上收集的文件访问历史数据进行模拟.由公式(1)可知,α越大,预取规则的逻辑支持度受最近支持度的影响就越大.分别取 α=0.5,α>0.5,α>>0.5(远大于)这 3 种情况,α=0.5 表示不采用基于动态支持度的预取规则;α>0.5 表示逻辑支持度受最近支持度的影响较大,实验中,取 α=0.7; α>>0.5 时表明逻辑支持度受最近支持度的影响要比过去支持度大得多,实验中取 α=0.9.图 5 显示了在 PFileSelect 算法取相同 K 值的情况下,整个测试集模拟完毕后,协作预取在不同 α 取值下的预取命中率.α=0.7 和 α=0.9 时的预取命中率都比 α=0.5 时要高,这说明我们提出的基于动态支持度的预取规则能够较好地反映文件动态流行度对预取性能的影响,提高预取命中率;而 α=0.9 时的预取命中率比 α=0.7 时要高,说明在本实验中,增大最近支持度对逻辑支持度的影响权重可以提高预取命中率.但这并不是说 α 越大,预取命中率就越高.事实上,对于不同的数据集,α 的最佳取值范围是不同的,需要通过具体实验分析加以确定.需要说明的是,本实验并没有与其他预取算法相比较,是因为目前主流的预取算法都没有考虑文件动态流行度对预取性能的影响,而本文提出的基于动态支持度的预取规则的思想则不需要通过复杂转换即可应用于其他预取算法,以提高这些算法的预取命中率.

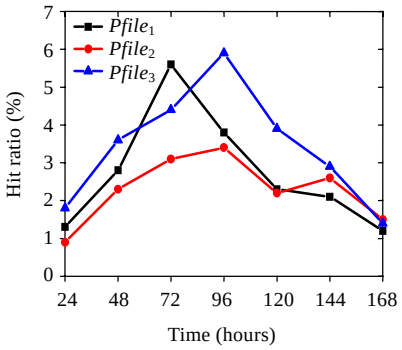


Fig.4 HR of $PFile_1$, $PFile_2$ and $PFile_3$

图 4 $PFile_1$,$PFile_2$,$PFile_3$ 的预取命中率变化

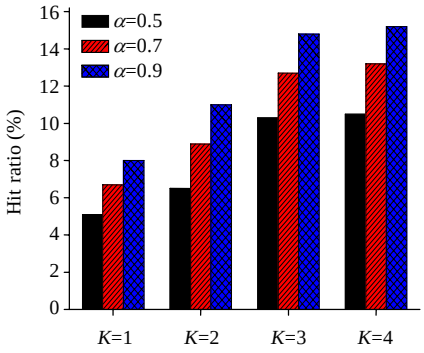


Fig.5 HR with respect of α

图 5 不同 α 取值下的预取命中率

实验 3 考察以作业类型为中心的本地文件预取算法 $JCFP_{local}$ 对预取性能的影响.实验采用 3 种不同的预取规则提取方式:第 1 种是直接从 SE 上的混合文件访问记录中提取预取规则,这也是传统的预取方式,用 TFP_{local} 表示;第 2 种是按应用类型对文件访问记录进行分类后提取属于不同应用类型的预取规则,用 $ACFP_{local}$ (application centric file prefetching)表示;第 3 种则是 $JCFP_{local}$.图 6 显示了 3 种不同预取算法的命中率情况,α 都取 0.9.由图 6 可以看出:TFP_{local} 的命中率最低,这证明了混合的文件访问记录确实降低了预取准确率;而 $JCFP_{local}$ 的命中率比 $ACFP_{local}$ 要高,这是因为,一个应用类型可能包含多个作业类型,因此,$JCFP_{local}$ 算法的分类粒度更细,相对来说提取的预取规则准确率也较高.事实上,在 Web 预取中,有研究者曾经提出以应用为中心的网页内容预

取[20],这启发了我们按应用类型对文件访问历史记录进行分类进而预取.然而,文献[20]中的应用其网页数量和内容都是确定的,所以预取对象也已事先确定.但是网格应用则不同,它所涉及的文件数量较多,即使属于同一应用的作业所访问的文件也不尽相同.因此,我们把分类的粒度进一步细化,从作业类型上对文件访问历史记录进行分类.但是,并不是粒度越细,预取的效果就越好,因为分类粒度越细,同一类下的文件访问模式就越少,提取的预取规则自然变少,预取效果反而可能降低.实验结果表明,针对虚拟组织中的网格应用,按照作业类型进行分类已能较好地反映不同类型作业对文件的需求.

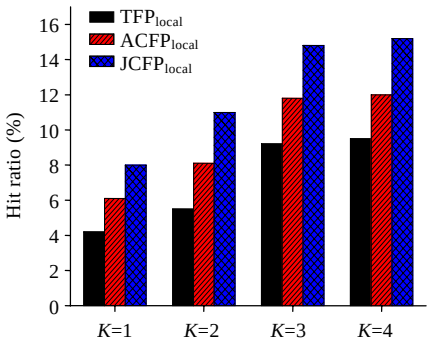


Fig.6 JCFP$_{local}$ vs. TFP$_{local}$ vs. ACFP$_{local}$ (HR)

图 6 不同预取算法的预取命中率

副本协作预取机制 CoRPM 应用的最终目标是为了降低网格作业的数据访问延迟,实验 4 通过对作业平均响应时间的测量来考察 CoRPM 对数据访问延迟的影响.本文在第 2 节已经指出,CoRPM 必须与基于文件访问历史的副本复制策略相结合才能进一步降低数据访问延迟,我们用“with CoRPM”表示这种副本方式,用“without CoRPM”表示仅采用基于文件访问历史的副本复制策略的方式.虽然基于文件访问历史的副本复制策略众多,但实验目标与采用何种基于文件访问历史的副本复制策略并无关系.因此,实验 4 采用了一种容易实现且普遍应用的策略:当某个网格节点对同一个文件的请求次数达到阈值时,就在该节点上创建该文件的副本.图 7 表示了模拟器模拟运行 100,200 直到 1 500 个作业后,两种不同的副本复制方式所对应的作业平均响应时间.模拟运行的作业分布在所有节点上,PFileSelect 算法取 K=3,公式(1)取 α=0.9.由图 7 可以看出,采用 CoRPM 后,作业平均响应时间得到了有效降低,最多的时候降幅接近 30%.这说明采用副本协作预取和基于文件访问历史的副本复制相结合的副本复制方式,确实可以进一步降低数据访问延迟;而随着运行作业的增多,作业平均响应时间仍能保持不错的降幅.这表明 CoRPM 具有稳定的性能表现.需要说明的是,模拟器并没有考虑服务调用时间、文件写入写出缓存等因素对作业平均响应时间的影响,节点带宽也默认一直是 100Mbps,因此,若在实际网格系统中应用 CoRPM,则作业平均响应时间的降幅可能稍低.

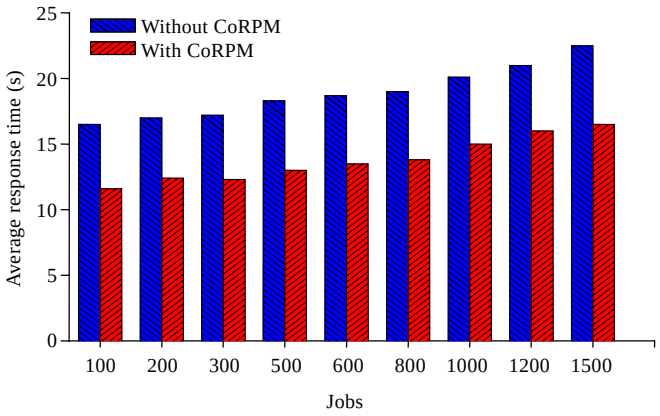


Fig.7 Without CoRPM vs. with CoRPM(ART)

图 7 采用 CoRPM 前后的作业平均响应时间

实验 4 表明,采用 CoRPM 确实可以有效地降低作业平均响应时间.但是,引入 CoRPM 不可避免地会带来额外的存储空间和网络带宽的消耗.前者主要是指虚拟组织各个节点上预取池占用的存储空间,其大小可以根据不同节点 SE 的空间大小进行配置.后者主要是由更多的数据传输所造成的,与原有的基于文件访问历史的副本

复制策略相比,CoRPM 需要从远程节点预取一些隐性高价值文件副本,这必然会带来更多的带宽消耗.但是,CoRPM 有效地降低了本地作业的平均响应时间,即以额外的带宽消耗换取了响应时间的降低.随着网络带宽的飞速增大,大多数网格平台,尤其是针对科学应用的网格平台,都具有十分充裕的带宽资源[21].因此,以额外带宽消耗换取响应时间降低的方法是可行的.

6 结束语

副本复制是数据网格中降低数据访问延迟的有效方法.本文在深入分析虚拟组织文件访问特性的基础上,通过引入隐性高价值文件概念,提出虚拟组织中的副本协作预取机制,以进一步降低数据访问延迟.首先,给出了虚拟组织副本协作预取架构,各个虚拟组织节点上的 PE 模块以协作的方式为虚拟组织内节点提供文件预取服务;其次,设计了副本协作预取流程,并针对不同作业类型文件访问的特点提出以作业类型为中心的预取算法和预取文件选择算法.模拟实验结果表明,副本协作预取机制与已有的基于文件访问历史的副本复制策略相结合,可以更加有效地降低数据访问延迟.

在下一步的工作中,我们计划将副本协作预取机制实现到 SEUGrid 校园网格平台中,逐步解决在实际网格环境部署中遇到的各种问题.此外,我们还将继续研究预取规则的存储和查询机制,引入结构化 P2P 模型来存储预取规则,以提高信息查询速度.

References:

[1] Pacitti E, Valduriez P, Mattoso M. Grid data management: Open problems and new issues. Journal of Grid Computing, 2007,5(3): 273−281. [doi: 10.1007/s10723-007-9081-9]

[2] Venugopal S, Buyya R, Ramamohanarao K. A taxonomy of data grids for distributed data sharing, management, and processing. ACM Computing Surveys, 2006,38(1). [doi: 10.1145/1132952.1132955]

[3] Eu-Datagrid.Org. The data grid project. http://www.eu-datagrid.org/

[4] Foster I, Kesselman C, Tuecke S. The anatomy of the grid: Enabling scalable virtual organization. Int'l Journal of High Performance Computing Applications, 2001,15(3):200−222. [doi: 10.1177/109434200101500302]

[5] Carman M, Zini F, Serafini L, Stockinger K. Towards an economy-based optimisation of file access and replication on a data grid. In: Proc. of the 2nd IEEE/ACM Int'l Symp. on Cluster Computing and the Grid (CCGRID 2002). Washington: IEEE Computer Society, 2002. 340−345. [doi: 10.1109/CCGRID.2002.1017156]

[6] Ranganathan K, Foster IT. Identifying dynamic replication strategies for a high performance data grid. In: Lee CA, ed. Proc. of the Int'l Grid Computing Workshop. LNCS 2242, London: Springer-Verlag, 2001. 75−86.

[7] Iamnitchi A, Doraimani S, Garzoglio G. Filecules in high-energy physics: Characteristics and impact on resource managemen. In: Proc. of the 15th IEEE Int'l Symp. on High Performance Distributed Computing (HPDC 2006). Los Alamitos: IEEE Computer Society, 2006. 69−80. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1652137&tag=1 [doi: 10.1109/HPDC.2006.1652137]

[8] Handurukande SB, Kermarrec AM, Fessant FL, Massoulié L, Patarin S. Peer sharing behaviour in the eDonkey network, and implications for the design of server-less file sharing systems. In: Proc. of the 2006 EuroSys Conf. NewYork: ACM Press, 2006. 359−371. [doi: 10.1145/1218063.1217970]

[9] Doraimani S, Iamnitchi A. File grouping for scientific data management: Lessons from experimenting with real traces. In: Proc. of the 17th IEEE Int'l Symp. on High Performance Distributed Computing (HPDC 2008). NewYork: ACM Press, 2008. 153−164. http://dl.acm.org/citation.cfm?id=1383429 [doi: 10.1145/1383422.1383429]

[10] Kroeger TM, Long DDE. Design and implementation of a predictive file prefetching algorithm. In: Proc. of the 2001 USENIX Annual Technical Conf. (USENIX 2001). 2001. 105−118. http://www.usenix.org/event/usenix01/kroeger.html

[11] Ban ZJ, Gu ZM, Jin Y. A survey of Web prefetching. Journal of Computer Research and Development, 2009,46(2):202−210 (in Chinese with English abstract).

[12] Shen HT, Shu Y, Yu B. Efficient semantic-based content search in P2P network. IEEE Trans. on Knowledge and Data Engineering, 2004,16(7):813−826. [doi: 10.1109/TKDE.2004.1318564]

[13] Capozza L, Stockinger K, Zini F. Preliminary evaluation of revenue prediction functions for economically effective file replication. Technical Report, DataGrid-02-TED-020724, 2002. http://www-vis.lbl.gov/~kurts/research/eco_model_evaluation.pdf

[14] Chu R, Nu XC, Xiao N. A data prefetching algorithm for RAM grid. Journal of Software, 2006,17(11):2234−2244 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/17/2234.htm [doi: 10.1360/jos172234]

[15] Sirivianos M, Park JH, Yang XW, Jarecki S. Dandelion: Cooperative content distribution with robust incentives. In: Proc. of the 2007 USENIX Annual Technical Conf. (USENIX 2007). 2007. 157−170. http://www.usenix.org/event/usenix07/tech/ sirivianos.html

[16] Yan XF, Han JW, Afshar R. CloSpan: Mining closed sequential patterns in large databases. In: Proc. of the 3rd SIAM Int'l Conf. on Data Mining (SDM 2003). 2003. 166−177. http://books.google.com.hk/books?id=AyJ9xrrwDnIC&hl=zh-CN&source=gbs_navlinks_s

[17] Geant4.org. The Geant4 toolkit. http://www.geant4.org/geant4

[18] Allison J, Amako K, Apostolakis J, Araujo H, Dubois PA. Geant4 developments and applications. IEEE Trans. on Nuclear Science, 2006,53(1):270−278. [doi: 10.1109/TNS.2006.869826]

[19] Luo JZ, Song AB, Zhu Y, Wang XP, Ma T, Wu ZA, Xu YB, Ge L. Grid supporting platform for AMS data processing. In: Chen G, Pan Y, Guo M, Lu J, eds. Proc. of the ISPA Workshops 2005. LNCS 3759, Berlin: Springer-Verlag, 2005. 276−285. [doi: 10.1007/11576259_31]

[20] Pons AP. Web-Application centric object prefetching. Journal of Systems and Software, 2003,67(3):193−200. [doi: 10.1016/S0164-1212(02)00129-2]

[21] Al-Kiswany S, Ripeanu M, Iamnitchi A, Vazhkudai S. Are P2P data-dissemination techniques viable in today's data-intensive scientific collaborations? In: Kermarrec AM, Bouge L, Priol T, eds. Proc. of the 13th European Int'l Conf. on Parallel Processing. LNCS 4641, Heidelberg: Springer- Verlag, 2007. 404−414. [doi: 10.1007/978-3-540-74466-5_44]

附中文参考文献:

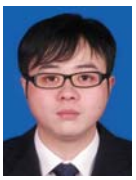
[11] 班志杰,古志民,金瑜.Web 预取技术综述.计算机研究与发展,2009,46(2):202−210.

[14] 褚瑞,卢锡城,肖侬.一种内存网格的数据预取算法.软件学报,2006,17(11):2234−2244. http://www.jos.org.cn/1000-9825/17/2234.htm [doi: 10.1360/jos172234]

田田(1983－),男,江苏淮安人,博士生,主要研究领域为网格与云计算,分布式数据处理.

罗军舟(1960－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为下一代网络体系结构,网格与云计算,网络安全,服务计算.

宋爱波(1970－),男,博士,副教授,CCF 会员,主要研究领域为网格与云计算,海量数据处理,Petri 网理论与应用.



伍之昂(1982－),男,博士,CCF 会员,主要研究领域为网格计算,服务计算.