基于 GFS-Net 的动态复制*)

夏前军　李庆虎　叶晓俊
(清华大学软件学院　北京 100084)

摘　要　GFS-Net 是一种新的 DHT 覆盖网络。本文介绍了 GFS-Net 的拓扑结构，并完善了其搜索算法，提出了与 GFS-Net 相适应的动态复制算法，它包括两个部分：(1)根据文件的请求频率增减副本数目；(2)根据下载性能较低的站点的分布确定新增副本的位置。该算法可以根据副本的请求频率和分布，动态调整副本的数目和分布，平衡各个站点的负载，降低用户下载文件的响应时间。

关键词　P2P，覆盖网络，GFS-Net，动态复制

Dynamic Replication Based on GFS-Net

XIA Qian-Jun　LI Qing-Hu　YE Xiao-Jun
(School of Software, Tsinghua University, Beijing 100084)

Abstract　GFS-Net is a new DHT overlay network. This paper introduces its topology, a search algorithm used in searching useful information over the network, and a dynamic replicating algorithm which mainly consists of two steps: determine the number of replicas per file based on request frequencies and change the locations of replicas according to the distribution of those sites of poor downloading performance. The major advantages of the algorithm are dynamic adjustment of the number and the distribution of replicas, automatic load balancing for each site, and reducing the response time when downloading files from the network.

Keywords　P2P, Overlay network, GFS-Net, Dynamic replication

1　介绍

P2P 文件共享系统需要解决两个主要问题：一个是资源如何定位，另一个是如何保证系统的可靠性。分布式哈希表(DHT，Distributed Hash Table)是解决资源定位问题的有效方法，而复制技术则是解决系统的可靠性问题的重要手段。

1.1　资源定位

分布式哈希表将每一个文件名通过哈希函数映射到一个键值，按一定的策略将“键-值”对分布在确定的节点上，其中“值”是指对应文件的基本信息，包括它所在节点的地址等。检索请求可以从任意节点上发起，通过哈希函数获取资源所在的位置，经过路由迅速找到目标站点。目前的 DHT 系统主要有 CAN、Chord、Viceroy 等。

已有的 DHT 系统忽略了其覆盖网络与底层物理网络的差异性，而且这些覆盖网络几乎没有与底层物理网络重合的可能性，这就使得那些基于覆盖网络所作的各种性能评价都是不可靠的，甚至是没有意义的。GFS-Net[1,3] 是一种新的 DHT 算法，它不但使覆盖网络与物理网络的重合成为可能，而且能够在节点的邻接数维护在常数的情况下使每次搜索都能经过小于 $O(\log n)$ 次跳转到达目的节点[3]，从而拥有优异的系统性能。

文[3]中的 GFS-Net 搜索算法有一个漏洞，所以本文在介绍 qdGFS-Net 搜索算法的同时提出了完善方法(第 2 节)。

1.2　复制

P2P 系统中的每一个节点都可能随时失效或主动退出，为了保证系统的可靠性，可以采用复制技术，通过把资源或服务复制到不同的节点，形成同一种资源或服务的多个备用站点。由于多个站点同时失效的概率要比单个站点失效的概率小得多，从而使得系统的可靠性获得提高。复制也使得用户在访问资源或服务时可以选择合适的服务点以获得更高的性能；另外，复制也为并行地访问资源提供了可能，所以复制也是提高系统性能的重要手段之一。

文[4]对数据网格中的最佳客户(Best Client)[5]、级联复制(Cascading Replication)、简单缓存(Plain Caching)和快速传播(Fast Spread)[6]等复制策略作了比较，通过模拟实验，考查响应时间和占用带宽两个方面，得出了关于这些复制策略的如下结论：如果用户在访问系统中的文件时带有完全的随机性，则“快速传播”是最好的策略，但很明显，它的缺点是占用了过多的空间；如果用户的访问模式带有一定的地域集中性，则“级联复制”的效果比其他都要好，不过，这种复制策略主要问题是子节点必须从其父节点复制文件，这大大限制了动态复制的灵活性。

文[2]提出了适用于网格文件系统(GridFS，Grid File System)的动态复制算法，包括副本选择、副本请求和副本重置三个部分，副本的树形组织结构与 GridFS 的广播式搜索算法相适应，资源不需要从源端经过层次结构的每一层来中转，可以直接从源端下载。

GFS-Net 没有相应的动态复制算法，文[2]提出的动态复制算法也无法直接应用到 GFS-Net 系统中。首先，GFS-Net 中每个文件都通过一个哈希函数映射到一个站点，其搜索算

*)国家 863 项目(2003AA4132301)。**夏前军**　硕士，研究方向：数据库，网格计算；**李庆虎**　博士，研究方向：数据库，网络计算，Peer-to-Peer；**叶晓俊**　教授，研究方向：数据库理论。

法也是利用文件名映射后的键值，如果文件名映射的站点是副本树根节点，搜索副本树的下层节点的代价较高(用户请求一个文件时对响应时间的要求较高)，如果文件名映射的站点是副本树的根节点的一个子节点，则搜索副本树根节点的其他子节点的代价较高，客户端下载文件时不适宜使用响应时间全局最优的副本的算法；其次，没有利用 GFS-Net 键值区间的地域集中性，简单地在请求文件的响应时间较长的客户站点上放置副本并不能较好地预测将来副本请求和分布，没有将副本放在最合适的站点上。因此，本文提出了适用于 GFS-Net 的动态复制算法(第 3 节)。

2 GFS-Net 及其搜索算法

基于 GFS-Net 的 P2P 系统的拓扑结构是一个变形的 B-Tree 结构，如图 1 所示，所有的叶节点都在同一层上，每个节点管理一个区间的对象。树结构的一个缺点就是某个分支节点的失效会导致以此节点为枢纽的上层节点与下层节点之间通道中断，使得系统的可靠性较差。为了提高系统的可靠性，GFS-Net 在同一个父节点的兄弟节点之间添加了连接，在爷-孙、叔-侄节点之间也添加了连接。为了更高的可靠性，还可以在辈分相差更大的节点之间添加连接，这由各个节点根据站点的可靠性指标自主决定。连接越多并不意味着节点的邻接数会随着网络规模的扩大而增大，因为一旦某个节点所要新添连接的范围确定下来，它的邻居数就不会大于某个确定的常数了。

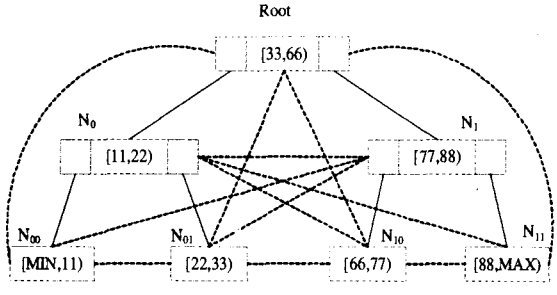

图 1 GFS-Net 示例

在介绍搜索(路由)算法之前，需要了解每个节点上的路由表的主要内容，然后定义树的层数的概念。邻居、覆盖区间、本地区间是路由表的主要条目，其中邻居是指与该节点有连接关系的节点；覆盖区间是指该节点及其所有子孙节点所存储区间的并集；而本地区间则仅仅是该节点自身所存储的所有存储区间的并集。覆盖区间是连续的，而本地区间很可能是间断的(子节点的个数大于 2 时)，而且本地区间是覆盖区间的子集。我们定义树的根节点为第 1 层，层数向下递增。

在 B-Tree 中，对键的搜索都必须起始于根节点，然后在路过的每个节点经过键值的比较逐步导向目标节点；而在 GFS-Net 中，对键的搜索可以从任意节点发起，经过一定的路由算法导向目标节点。任意节点 N 在收到一个对键 K 的查询请求 Q_K 后，会分三个步骤对它进行处理：

1) 检查 K 是否包含在节点 N 的本地区间 Rlocal 里，如果是，则在本地对 K 进行搜索，有则取出对应的元信息返回给查询者，没有则返回查询失败；如果不是，则继续下一步；

2) 在路由表里查找覆盖区间 R(overlay,i)包含 K 的邻居节点 Ni，如果这样的 Ni 存在，则把 Q_K 转发给 Ni，否则继续下一步；

3) 遍历路由表得到覆盖区间 R(overlay,j)离 K 最近的邻居节点 Nj，并把 Q_K 转发给它。如果最近的邻居节点有多个(K 到 N 的多个邻居节点的距离相等且最小)，则选择层数小于节点 N 的层数且层数最大的邻居节点 Nj，并把 Q_K 转发给它。其中，键 K 到区间[a; b)的距离 $d=\min(|K-a|,|K-b|)$。

算法 1 基于 GFS-Net 的 P2P 系统的路由算法

```
if (N. local-range includes Key)
  return VALUE(Key);
j = 0;
for (int i=0; i<N. neighbors--count; i++)
{
  if (N. overlay-range [i] includes Key)
    return OOKUP(Key, N. neighbors [i]);
  if ( distance(N. neighbors [i], Key)
    <distance(N. neighbors [j], Key)
    j = i;
  if (distance(N. neighbors [i], Key) ==
      distance(N. neighbors [j], Key) and
      depth(N. neighbors [i])>
            depth(N. neighbors [j]) and
      depth(N. neighbors [i])<depth(N))
      j = i;
}
return LOOKUP(Key, N. neighbors [j]);
```

由算法 1 可以看出，由于同一个父节点的兄弟节点之间添加了连接，在爷-孙、叔-侄节点之间也添加了连接，既增加系统的可靠性，也减少了层数较低的节点的负载。

需要说明的是文[3]的搜索算法的第三步没有考虑最近的邻居节点有多个的情况，然而这种情况是有可能出现的，比如说拓扑结构是一棵满五叉树 T，总共有 10 层，根节点下有 5 棵子树，分别记为 A、B、C、D 和 E。同一层上右侧节点的区间上的任何一个值大于等于左侧节点区间上界。我们从整棵树 T 的第 7 层且是子树 A 的最右侧节点 Nc 开始搜索，目标键值 K 在整棵树 T 的第 7 层最右侧节点上，A 子树的每一层的最右侧节点的覆盖区间的上界都是相等的，计为 b，而要找的键值 K 大于上界，所以 K 到所有这些节点的距离都是 $K-b$，这时 K 到 Nc 的最近的邻居节点就有多个。本文在这一点上对搜索算法进行了完善：由于随机选择可能出现搜索过程在树中转圈，向下寻找无法到达目标节点，因此要向上寻找。记多个最近的邻居节点中层数小于当前节点层数且层数最大的邻居节点为 Nj，K 到节点 Nj 的某些邻居节点距离可能比 K 到当前节点的所有邻居距离小，也就是说将查询 Q_K 从当前节点传递给节点 Nj 导致查询 Q_K 向目标节点靠近，所以查询向节点 Nj 转发(见算法 1 的斜体部分)。

3 GFS-Net 的动态复制算法

文[2]中提出了基于 GridFS 结构的动态复制算法，其实基于 GFS-Net 动态哈希表的键值分布特点，我们设计更好的动态复制算法。这里，我们称原始文件为主副本，原始文件和该文件的所有拷贝均称为副本。

3.1 副本的组织

GFS-Net 系统中副本的组织采用星形拓扑结构，如图 2 所示。主副本在文件的键值映射的节点上，其他副本则没有要求。搜索文件时均查找主副本站点，由主副本站点提供所有副本的信息。主副本站点定期检查各个副本的存在，并维护这些副本的地址列表(站点地址和文件在站点上的路径)，除主副本之外的每个副本定期也检查主副本的存在，并定期将副本所在站点的平均输出速度发送给主副本所在站点。

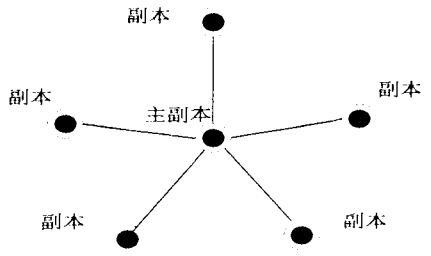

图 2　副本的组织

这种组织的可靠性是有保证的，假如主副本站点和一些其他副本站点退出系统，只要系统中还存在一个副本，就可以利用这一副本在该文件键值所在的站点 S 处做一个主副本（将副本内容拷贝到站点 S，将其看作是主副本）。并建立相应的连接（主副本和其他副本之间相互记录对方的地址）。如果还存在其他副本，则在做上面同样操作时会发现主副本已经存在，只要建立连接就可以了，星形拓扑结构可以很容易地恢复。

与文[2]中的副本树拓扑结构相比，这种拓扑结构的主要优点是：1）可以迅速获得全部副本的信息，而且这些信息是可靠的，文件获取者可以得到响应时间全局最优的站点而不是局部最优的站点；2）除主副本站点外单个节点故障影响的范围较小（1 个站点），主副本站点失败恢复也比较容易。不过它也有其缺点，访问某个文件均要通过同一个主副本站点，如果是一个热点文件，则主副本站点需要处理大量的查询请求，产生性能瓶颈，不过这种瓶颈站点仅处理一个文件，与整个系统仅有一个有特殊用途的服务器相比就微不足道了。为了减少主副本站点的负载，可以考虑让原来主副本所在的站点不存放副本，这同时也增加灵活性，加快主副本站点失败时的恢复速度。

3.2　副本的选择

客户端 C 对自己所需要的文件发起请求之后，经过在网络内的搜索，如果找到了所需文件就会收到这个文件在网络中分布情况的一个列表，列表里的每一项都包含有对应节点的联系地址以及副本在其上的路径信息，客户端 C 就要在这些列表里挑选一个副本进行后续的访问，这个挑选的过程称为“副本选择”。

副本选择依靠的标准就是从客户端 C 到这些源端节点的距离，这个距离主要由响应时间来反应。客户端 C 通过它本身的历史记录及查询结果中包含的信息可以对连往每个节点的响应时间作出较为合理的预测。

响应时间（Tr）由两个部分组成——服务时间（Ts）和等待时间（Tw），即 $Tr = Ts + Tw$。客户端 C 将选择 Tr 值最小的那个源端进行后续下载。等待时间是指跟源端建立连接所需要的时间，也就是下载时收到第一个数据包之前所等待的时间；服务时间是指从得到第一个数据包到收到最后一个数据包之间所花费的时间，它可以用将要下载的文件的大小（S）除以传输速率（V）来得到，这样一来响应时间（Tr）的计算公式就变为：

$$Tr = S/V + Tw \tag{1}$$

对于曾被客户端 C 所访问过的节点，C 维持有与该站点之间最近的 N 次关于 Tw 与 V 的连接记录的平均值，这些平均值记录在客户端 C 的最近连接信息表中。客户端 C 与站点每一次连接之后需要对客户端 C 与该站点的平均响应时间和平均速度进行更新。假设客户端 C 与某个站点 Site 1 的连接总次数为 M，客户端 C 的连接信息表中站点 Site 1 的平均响应时间为 Tw，平均传输速率为 V，客户端 C 与站点 Site 1 进行了一次新的连接，响应时间为 tw，传输速度是 v，更新后客户端 C 与站点 Site 1 的平均响应时间为 Tw'，平均速度为 V'。如果连接总次数 M 小于 N（N 为站点预设的一个整数），则：

$$Tw' = Tw - Tw/(M+1) + tw/(M+1) \tag{2}$$

$$V' = V - V/(M+1) + v/(M+1) \tag{3}$$

如果连接总次数 M 大于等于 N，则：

$$Tw' = Tw - Tw/N + tw/N \tag{4}$$

$$V' = V - V/N + v/N \tag{5}$$

公式(1)中的 Tw 和 V 分别取客户端 C 最近连接信息表中站点 Site 1 的 Tw 和 V 的值。如果 C 上没有关于访问站点 Site 1 的历史记录，Tw 取 C 所了解的那些节点的等待时间平均值的平均值，V 则直接使用站点 Site 1 最近的平均输出速度 VSite1，VSite1 是搜索结果的一部分，它由副本所在的站点 *Site* 1 实施监控并定期发送给主副本站点。

3.3　副本的增减

每个站点 S 都维护一张副本使用频率表，每一条记录对应该站点上一个副本，定期更新，记录该副本的结束时间为最后一次更新该记录的长度为 T 的时间段内使用频率 f（由下面的计算方法可知它实际上是一个近似值）。每个站点有另外一张最近副本使用次数表，每一条记录对应该站点上一个副本，记录最近一次更新副本使用频率表该副本相应的记录后该副本被使用的次数。每隔时间 t（$t < T$，T 一般取 t 的整数倍），主副本站点发出命令，更新使用频率表中相应副本的纪录。假设距离上次更新副本使用频率表中某个副本时间 t 时，副本使用频率表中记录的该副本的使用频率 f，最近副本请求次数表中记录的该副本的使用次数是 n，则更新后该副本的使用频率 f' 为：

$$f' = f - f * t/T + n \tag{6}$$

每次更新副本使用频率表中副本使用频率后需将最近副本使用次数表中该副本的使用次数置 0。

文件使用频率 F 是它的所有副本的使用频率之和（不考虑一次文件请求使用多个副本的情况）：

$$F = \sum_{1}^{l} f_i \quad (l \text{ 是副本数，} f_i \text{ 是第 } i \text{ 个副本的使用频率}) \tag{7}$$

所以文件使用频率 F 也是定期更新的，每次更新后基于更新后的文件使用频率 F 判断需要增加副本还是减少副本。我们可以选择一个文件副本数 M 和文件使用频率 F 关系的函数来确定文件的文件使用频率 F 下副本数 M，这个函数必须满足当文件使用频率 F 增加或减少时副本数 M 会相应地增加或减少，因此我们可以取：

$$M = \lfloor K \sqrt[b]{F^a} \rfloor \quad (a, b \text{ 是正整数且 } b \geqslant a; K \text{ 是一个正常数}) \tag{8}$$

公式(8)中常数 a，b 和 K 可以由用户选择，每一个副本所在的站点有一张客户信息表，用于登记从副本所在的站点下载的副本的客户端键值区间的中点和下载速度。客户端 C 从一个副本处下载完毕后，将下载响应时间和客户端 C 设定的响应时间阈值比较，如果响应时间小于该阈值，则通知副本所在的站点将客户端键值区间的中点和下载速度记录下来。一定时间后，文件的各个副本所在的站点可能都有一些这个

（下转第 82 页）

段的数据就用该数据密钥进行加密，并且该密钥由用户密钥加密后也存入密钥数据库中。因此，要想对加密字段进行解密，需要有数据密钥，而数据密钥的获得，必需有户密钥才行，用户密钥的获得，只有输入正确的用户口令才可以。从中可以看出，密钥的管理是多层次的，没有密钥或仅知道部分密钥，是不可能对数据库加密数据进行破解的，这就大大提高了数据库数据的安全性能。

5. *数据库连接模块* 数据库连接由几个 JavaBean 构成，实现 Web 服务器与后台数据库的连接，并读取和存入数据。本系统通过 JDBC-ODBC 与数据库建立连接，并将用户数据请求转换为相应的 SQL 语句，对数据库进行查询、添加、删除和修改等操作。

6. *加密/解密引擎* 数据库加/脱密引擎是数据库加密系统的核心部件，负责在后台完成数据库信息的加/解密处理，对应用开发人员和操作人员是透明的。同数据库连接一样，该模块也是做成 JavaBean 的形式，当加解密控制确定用户数据不需要加密/解密处理的时，由连接模块和数据库直接连接；当需要对数据加密时，由 JCE 产生密钥，并对数据进行加密；对于需要解密的数据，则首先从密钥库中分别取出用户密钥和数据密钥，并利用用户密钥解密数据密钥，而后再利用数据密钥对加密数据进行解密。

3.4 系统安全性分析

数据库中敏感数据经加密系统处理后，变成形式上无规则的乱码，这样即使非法使用者窃取了数据库文件，他仍然难以得到所需的信息。另外，在本系统中由于密钥管理采用的是多级加密机制，无论是数据密钥还是用户密钥都是加密形式存储，因此即使非法用户打开了数据库，由于他不知道用户口令，无法解密用户密钥，更不可能得到数据密钥，因此他无法对加密数据进行解密，这就大大提高了关键数据的安全性。

结束语 数据库加密方面的探索工作已进行多年，随着网络技术的发展，基于数据库的安全研究出现了新课题，就是如何在网络的环境下提供一个安全的数据库环境。在实际应用中要实现一个有效、快捷和真正安全的网数据库是十分困难和复杂的。本系统对基于 B/S 结构的数据库加密技术进行了研究，设计了一个加密系统，并利用 JSP 技术进行了实现。理论分析和实验结果表明，该系统可以提高数据库系统的安全性，有效抵御非法窃取和访问数据库数据。但在网络环境下，数据在网络上传输时仍是明文数据，仍然存在被非法窃取的可能性。解决这方面的问题要运用 JAVA 的数字化身份认证体系和数据的安全传输等一系列的安全措施来构建起一个安全体系。这也是我们下一步要做的研究工作。

参 考 文 献

1 戴一奇，尚杰，陈卫，等. 一种新的数据库加密密钥管理方案[J]. 清华大学学报(自然科学版)，1995，35(4)：43～47
2 Stallings W. 密码编码学与网络安全—原理与实践(第 2 版)[M]. 北京：电子工业出版社，2001
3 Garms J. Java 安全性编程指南[M]. 北京：电子工业出版社
4 李广鑫，马志欣，等. 基于 B/ S 结构的远程实时监测系统[J]. 计算机应用研究，2003，10：147～150
5 张华桁，宋立群，柯科峰. B/S 构架信息系统的安全策略研究与开发[J]. 计算机工程与应用，2004，13：159～162
6 朱鲁华，陈荣良. 数据库加密系统的设计与实现[J]. 计算机工程，2002，28(8)：61～63

(上接第 69 页)

文件相关的客户端键值区间的中点和下载速度记录，利用 K-means 聚簇算法处理这些键值区间中点和下载速度数据，可以找出最需要副本的键值点，根据这些键值点，我们可以找到一些站点，在每个这样的站点上做一个文件副本。

如果文件的使用频率下降，需要的副本数可能需要减少。根据公式(8)，如果系统需要减少文件的副本数时，根据需要可以将使用频率最小的几个副本撤销。

3.4 副本的重置

对于一个副本节点 S_{bak}，它上边的文件副本除了会被上面的过程给强迫释放外，另一种情况就是主动转移或释放。当 S_{bak} 上需要创建新的文件副本而存储空间不足时，这种情况就发生了，此时站点 S_{bak} 就会试图把站点 S_{bak} 上所有副本中使用频率最低的副本 $F1$ 给转移到它在 *GFS-Net* 中的邻居节点上。邻居节点的选择的次序的依据是副本节点 S_{bak} 的键值区间中点到邻居的距离。如果 $F1$ 的使用频率都不比邻居节点 Sn 上的任何一个副本的使用频率高且 Sn 上没有足够的空间用来放置副本 $F1$，则找另外一个邻居节点。如果副本 $F1$ 的使用频率比全部的邻居站点任何一个副本的使用频率低且邻居都没有足够的空间用来放置副本 $F1$，则 S_{bak} 把 $F1$ 直接释放。相反，如果找到一个邻居节点 Sn：如果 Sn 上有足够的空间来盛放副本 $F1$，S_{bak} 就会把 $F1$ 转移给 Sn；如果副本 $F1$ 的使用频率比 Sn 上的某个文件副本 $F2$ 的使用频率高，则尝试用类似 $F1$ 的方法将 $F2$ 移出，然后再查看 Sn 的空间是否够用，如果够用则将 $F1$ 放到站点 Sn，否则继续分析是否有副本可以移出。重复上面的操作直至找到一个有足够的空间的邻居节点放置副本 $F1$。如果副本节点 S_{bak} 上的存储空间仍不满足要求，则继续上述过程直到 S_{bak} 上的存储空间满足要求为止。节点 S_{bak} 上负载太重也可以用上述方法处理其上的副本以降低站点的负载。

该动态复制算法，使用星形拓扑结构，利用近期用户请求的频率和分布预期将来的用户请求的频率和分布，动态调整副本的数目和分布。在用户的访问模式带有一定的地域集中性情形下，与级联复制相比除了具有更高的复制灵活性之外还具有更好的性能。

总结 本文的主要贡献：1)完善了 GFS-Net 的搜索算法；2)提出了适用于 GFS-Net 的动态复制算法。该动态复制算法包括两个部分：依据副本的请求频率增减副本数的方法；根据下载响应时间较长的客户端的分布(利用客户端上的覆盖区间的中值)布置新增副本位置的方法。这些方法平衡了各个站点的下载冲击，降低用户文件下载的响应时间。

对于大文件，由于客户端请求文件时返回的是所有副本的地址信息列表，客户端可以选择多个源端来进行条带传输(stripped transfer)以提高下载速度，而如何选择源端和如何协调各个下载线程是我们下一步的工作内容。另外，公式(8)中常数 a，b 和 K 的选择虽然通过在远程教育平台的应用实验过，但还要通过其他应用来总结出合理的取值。

参 考 文 献

1 李庆虎. 基于 P2P 架构的网格文件系统研究：[博士论文]. 清华大学，2004
2 Li Q H, Wang J M, Lam K Y, Sun J G. Gridfs: A web-based data grid for the distributed sharing of educational resource files. Advances in Web-based Learning Intl. Conf. ICWL 2003, Lecture Notes in Computer Science 2783, 2003. 81～92
3 Li Q H, Wang J M, Sun J G. GFS-Btree: A scalable peer-to-peer overlay network for lookup service. The Second International Workshop on Grid and Cooperative Computing, LNCS3032: 340～347
4 Ranganathan K, Foster I. Identifying dynamic replication strategies for a high performance data grid. In: Proc. of the Intl. Workshop on Grid Computing, Denver, Colorado, Nov. 2001
5 Gwertzman J S, Seltzer M. The case for geographical push-caching. In: Proc. Fifth Workshop on Hot Topics in Operating Systems, May 1995
6 Michel S, Nguyen K, Rosenstein A, Zhang L, Floyd L, Jacobson V. Adaptive web caching: Towards a new global caching architecture. In: Proc. of the 3rd Intl. WWW Caching Workshop, 1998