高技术通讯 2010 年 第 20 卷 第 1 期:26 ~ 32

doi:10.3772/j.issn.1002-0470.2010.01.005

一种数据网格存储模型与并行下载调度算法①

曲明成② 吴翔虎③ 廖明宏* 杨孝宗 左德承

(哈尔滨工业大学计算机科学与技术学院 哈尔滨 150001)

(*厦门大学软件学院 厦门 361005)

摘 要 针对在数据网格中创建多副本虽可有效提升下载速度、降低网络流量,但多副本创建会带来大量存储开销和网络流量开销,以及基于 GridFTP 协议的各种并行下载算法虽可进一步提升下载速度,但仍不能解决多副本对存储空间和网络流量的影响的问题,提出了一个能保证数据的完整性、存储的可靠性和降低存储空间的数据网格存储模型,并基于该存储模型和 GridFTP 协议,提出了一个并行下载调度算法。实验表明,该算法只需要较少的冗余便可达到现有的针对全副本的并行下载算法可达到的理想下载速度,取得较好的效果,实现并行快速传输、节约存储空间和降低网络流量的目标。

关键词 数据网格,存储模型,并行下载调度算法,GridFTP

0 引 言

数据网格的概念来自网格(grid),它是网格技术在数据管理方面的应用和实现,目的是解决数据密集型计算应用中方便高效使用分布式数据资源的问题。针对数据网格中的热点数据,为了能有效地提高用户的访问速度,同时降低网络负载,近来很多研究集中于如何能够合理地创建副本并进行高效的副本定位[1],这在一定程度上提升了数据的下载速度,减轻了网络流量。但是这些研究都忽略了网格中数据的海量特性,以及创建多个副本所带来的大量存储开销和网络开销,如果数据量很大,创建多个副本甚至是得不偿失的[2,3]。

GridFTP 是 Globus 项目组开发的一个新的数据传输协议,它基于规范的文件传输协议(FTP)协议,并对其进行了全面扩展,旨在为网格上分离的存储系统间的互操作提供一个通用的、可扩展的底层数据传输协议,并为应用程序提供统一的访问接口,使数据传输的性能得到了明显的提高。针对 GridFTP 协议规范,很多研究分别基于 LINUX, UNIX, WINDOWS 进行实现方法研究,取得了较好的应用效果[4-6]。GridFTP 提供了条状数据下载方式,即 GridFTP 客户端可以并行地从多个 GridFTP 服务器端下载同一个数据的不同数据块。基于 GridFTP 协议与多副本理论,已研究出了很多的并行下载调度算法,这又大幅度提高了数据的下载速度[7-9]。但是这些研究仍旧没有回避部署多个副本所带来的存储资源浪费、网络流量问题。因此如何用较少的冗余存储,充分利用 GridFTP 并行传输协议来提升数据传输速度是当前的数据网格面临的挑战性问题。

基于上述理论和存在的问题,本文提出了一个数据网格存储模型,该模型使用较少的冗余保证了数据的可靠性。然后针对该存储模型和 GridFTP 并行下载协议提出了一个数据并行下载调度算法。实验证明该算法可以达到较理想的调度效果,与已有的基于全副本的算法进行了比较,取得了很好的结果,达到了节约存储空间、提升下载速度和降低创建副本带来的网络流量的多重优化目标。

1 基本原理

文献[7-9]基于 GridFTP 的多目标并行下载算法,通过为每个下载任务合理地分配多个副本中的不同数据块,使下载速度较快的任务下载较多的数据,其理论内涵为:能够满足在下载的过程中让每个下载任务始终不间断地执行下载操作,保证整体下载速度的最大化和时间最小化。令一次下载任务选

① 国家自然科学基金(60533110)资助项目。
② 男,1980 年生,博士生;研究方向:网格计算;E-mail:qumingcheng@126.com
③ 通讯作者,E-mail: wuxianghu@hit.edu.cn
(收稿日期:2008-11-24)

定的副本节点个数为 k,节点分别为 $N_i(0 \leqslant i \leqslant k-1)$,各节点对应一次数据下载请求的平均下载速度分别为 $V_i(0 \leqslant i \leqslant k-1)$。基于平均传输速度,文献[7-9]在理论上可以达到的理想下载速度为 $\sum_{i=0}^{k-1} V_i$,即从各副本节点下载速度的加和,那么在各节点下载到的数据量占总数据量的比例极限为 $V_j/(\sum_{i=0}^{k-1} V_i)(0 \leqslant j \leqslant k-1)$。从这些算法可以看出,在数据下载中每个副本只有一部分是有效下载数据。基于此,本文思路为:用较少的冗余量和相应调度算法来达到与已有的基于全副本的算法接近的下载速度,从而节省大量的存储空间和解决全副本创建带来的网络流量问题。

2 算法数据存储模型

定义 1 (*metasum* 等分)令总数据量为 M,对整个数据进行等分,令分割的份数 $metablks = k(k-1) \times metasum$,$k$ 为副本节点的个数。分割方式为:先将数据等分成 $k(k-1)$ 份,再将每一份等分成 *metasum* 份,这里 *metasum* 是一个可变参数,则每一份的数据量由式

$$metadata = \frac{M}{k(k-1) \cdot metasum} \tag{1}$$

所示。

定义 2 (本地数据)将定义 1 中分割的 $k(k-1) \times metasum$ 份数据平均分配到 k 个节点上,则每个节点存储 $(k-1) \times metasum$ 份数据。称这些数据为节点 N_i 的本地数据 LND_i。*LND* 表示本地数据,LND_i 表示节点 i 的本地数据。

定义 3 (本地数据虚拟组)将节点 N_i 存储的本地数据块进行虚拟组划分,即将 *metasum* 个数据划为一组,将划分后的组进行节点内编号。由定义 2 知,一个节点本地数据可以划分的虚拟组数为 $(k-1)$ 个,令虚拟组为 $G_i^j(0 \leqslant i \leqslant k-1,\ 0 \leqslant j \leqslant k-2)$。令 G_i 表示节点 N_i 的所有虚拟组,G 表示当前虚拟组。

定义 4 (剩余节点集合)令刨除节点 N_i 后的所有参与存储的节点集合为剩余节点集合,即 $\bigcup_{y=0}^{k-2} \overline{N_i^y} = \bigcup_{j=0, j\neq i}^{k-1} N_j$,$\overline{N_i^y} = N_j$,$\begin{cases} y = j & (j < i) \\ y = j-1 & (j > i) \end{cases}$。

定义 5 (交叉存储)将节点 N_i 的数据 G_i 存储到刨除 N_i 的其他 $k-1$ 个节点 $\overline{N_i^e}$ 上,即 $G_i \rightarrow \overline{N_i^e}$,并满足规则 $\left(\bigcup_{r=e}^{w}(G_i^r \rightarrow \overline{N_i^e})\right)\Big|_{e=0}^{k-2}\{w = (e + (p-1))\bmod k, p \leqslant k-1\}$,$p$ 为指定的常数,"→"表示存储到,称这种存储方式为交叉存储。

数据存储模型:节点 N_i 存储的所有数据 AND_i 包括本地数据 LND_i 和其他节点的交叉存储数据 OND_i,可以推出 $AND_i = (LND_i) \cup (OND_i) = (\bigcup_{j=0}^{k-2} G_i^j) \cup (\bigcup_{(a=0, a\neq i)}^{k-1}(\bigcup_{r=e}^{w} G_a^r))\begin{cases} e = i-1 & a < i \\ e = i & a > i \end{cases}$ 和 $\begin{cases} w = (e + (p-1))\bmod k \\ p \leqslant k-1 \end{cases}$,($p$ 为指定的常数),称其为数据存储模型。*OND* 表示本地数据,OND_i 表示节点 i 的本地数据。

定理 1 (P 完整性)如数据满足存储模型,则当有任意 P 个节点不可用时,其余 $k-p$ 个节点中数据的并集仍然等于整体数据量 M,即完整的数据,称这种性质为 P 完整性。

证明:对任意的一个 G_i^x 进行讨论,其中 $i \in (0, k-1)$,$x \in (0, k-2)$,G_i^x 表示任意一个节点 N_i 的任意一个本地虚拟组。

由定义 5 可知,$e \in (0, k-2)$,$r \in (e, w)$,又 $\{w = (e + (p-1))\bmod k, p \leqslant k-1\}$,则 e 到 w 共有 p 个数。取出一个子集 $e \in (x-(p-1), x) \subset (0, k-2)$,因为对于 e 的每次取值,r 需要从 e 开始取 p 个值。当 e 由 $x-(p-1)$ 增加到 x 过程中,r 的取值范围分别为 $\{x-(p-1), x\}, \cdots, \{x, x+(p-1)\}$,$x$ 重复了 p 次,又由于 e 经历了 P 个值的变化,即 G_i^x 有 p 次存储到节点 $\overline{N_i^e}$ 上,这里 $e \in (x-(p-1), x)$。加上 G_i^x 虚拟组所在的节点,共有 $p+1$ 个节点存储 G_i^x。因此如果有任意的 p 个节点不可用,仍然有一个存储 G_i^x 的节点。由于 G_i^x 具有任意性,则所有节点的所有虚拟组均满足上述推导。证毕。

引理 1 满足数据存储模型的数据,其整体存储数据量可以表示为 $k(1+p)(k-1) \times metasum$。

证明:由定义存储模型可以推出:

$$\begin{aligned} \sum_{i=0}^{k-1} AND_i &= \sum_{i=0}^{k-1}((k-1) \cdot metasum \mid_{LND_i} \\ &\quad + (1+p) \cdot metasum \cdot (k-1) \mid_{OND_i}) \\ &= k(k-1)(1+p) \cdot metasum \end{aligned}$$

证毕。

定义 6 (存储空间使用量率,SAV)采用存储模型总体数据存储量与采用 k 个完整副本存储的总数据量的比值定义为存储空间使用量率 *SAV*。由定义 1 和引理 1 可得 *SAV* 如式

$$SAV = \frac{k(k-1)(1+p) \cdot metasum}{k \cdot k(k-1) \cdot metasum} = \frac{1+p}{k} \tag{2}$$

所示。

定义7 （极限速度比，V_{ratio}）对于任意的节点 N_i、N_j，其极限下载数据量比值 V_{ratio} 为 V_i/V_j，且有 $\frac{1}{(k-1)(1+p)\cdot metasum} \leqslant V_{ratio} \leqslant (k-1)(1+p)\cdot metasum$。

定义8 （最大速度上限，V_{top}）在理想的下载时间内，任意节点 N_i 下载的数据量不能大于 AND_i，则有 $\left(V_{max}/\sum_{i=0}^{k-1}V_i\right)k(k-1)\cdot metasum \leqslant (k-1)(1+p)\cdot metasum \Rightarrow V_{top} \leqslant \frac{(1+p)}{k}(\sum_{i=0}^{k-1}V_i)$，称 V_{top} 为最大速度上限。

算例1 给定 $k=4, P=2, metasum=4$，则满足存储模型的数据分配推演过程包适下述6个步骤：

(1)由定义1得，数据分割份数 $=4\times(4-1)\times4=48$。令这些数据块分别为(1)，(2)…(48)。

(2)由定义2得：$LND_0=\{(1),(2)\cdots(12)\}$；$LND_1=\{(13),(14)\cdots(24)\}$；$LND_2=\{(25),(26)\cdots(36)\}$；$LND_3=\{(37),(38)\cdots(48)\}$。

(3)由定义3得：(a) $G_0^0=\{(1),(2),(3),(4)\}$，$G_0^1=\{(5),(6),(7),(8)\}$，$G_0^2=\{(9),(10),(11),(12)\}$；(b) $G_1^0=\{(13),(14),(15),(16)\}$，$G_1^1=\{(17),(18),(19),(20)\}$，$G_1^2=\{(21),(22),(23),(24)\}$；(c) (d) 略。

(4)由定义4得：(a) $\overline{N_0^0}=N_1, \overline{N_0^1}=N_2, \overline{N_0^2}=N_3$；(b) $\overline{N_1^0}=N_0, \overline{N_1^1}=N_2, \overline{N_1^2}=N_3$；(c) (d) 略。

(5)由定义5得：(a) $G_0\rightarrow\overline{N_0^0}=G_0\rightarrow N_1=\{G_0^0, G_0^1\}\rightarrow N_1, G_0\rightarrow\overline{N_0^1}=G_0\rightarrow N_2=\{G_0^1, G_0^2\}\rightarrow N_2, G_0\rightarrow\overline{N_0^2}=G_0\rightarrow N_3=\{G_0^2, G_0^0\}\rightarrow N_3$；(b) $G_1\rightarrow\overline{N_1^0}=G_1\rightarrow N_0=\{G_1^0, G_1^1\}\rightarrow N_0, G_1\rightarrow\overline{N_1^1}=G_1\rightarrow N_2=\{G_1^1, G_1^2\}\rightarrow N_2, G_1\rightarrow\overline{N_1^2}=G_0\rightarrow N_3=\{G_1^2, G_1^0\}\rightarrow N_3$；(c) (d) 略。

(6)由存储模型得出如下的一个数据分配：$AND_0=\{(1),(2)\cdots(12)$；(13)，(14)，(15)，(16)；(17)，(18)，(19)，(20)；(25)，(26)，(27)，(28)；(29)，(30)，(31)，(32)；(37)，(38)，(39)，(40)；(41)，(42)，(43)，(44)$\}$；$AND_1=\{(13),(14)\cdots(24)$；(1)，(2)，(3)，(4)；(5)，(6)，(7)，(8)；(29)，(30)，(31)，(32)；(33)，(34)，(35)，(36)；(41)，(42)，(43)，(44)；(45)，(46)，(47)，(48)$\}$；AND_2, AND_3 略。

3 并行下载调度算法

3.1 算法基本思想

定义9 （本地数据剩余量，S_i）节点 N_i 理想的下载数据量与本地数据 LND_i 的差值定义为本地数据剩余量，即 $S_i=V_i/(\sum_{j=0}^{k-1}V_j)-LND_i$。

该指标反映出节点下载任务的负载情况，S_i 趋于或均等于0时说明各节点可以大致同时完成下载。如果 S_i 为正说明该节点下载速度较快，可以分担 S_i 为负的节点的下载任务。从定义9可以看出，在理想情况下(数据块无限小)有 $\sum S_i=0$。

实例1 $k=4$，$p=1$，$metasum=3$，$V_1=12$，$V_2=10$，$V_3=4$，$V_4=28$。算法目标为：尽量保证各节点同时完成下载任务，且下载的数据不重复。那么我们先给出算法最终的一个调度结果如表1所示。从节点 N_i 下载加灰的数据块，根据速度的大小可以算出各节点理想的下载块数分别为8、6.7、2.7、18.7，实际的调度结果为8、7、3、18。调度前节点 N_i 负载 S_i 为(-1，-2，-6，10)，调度后节点 N_i 的负载情况 A-B 为(0，-0.3，-0.3，0.7)，负载已基本均衡。

因为块数为整数，所以必须进行四舍五入。$SAV=50\%$，在接近理想下载的情况下较完整副本部署节约了50%的存储空间。

表1 实例1

节点	速度	S_i	理想 A	实际 B	A-B	节点 *LND* 数据	节点 *OND* 数据
N_1	12	-1	8	8	0	(1)(2)(3)(4)(5)(6)(7)(8)(9)	(10)(11)(12)(19)(20)(21)(28)(29)(30)
N_2	10	-2	6.7	7	-0.3	(10)(11)(12)(13)(14)(15)(16)(17)(18)	(1)(2)(3)(22)(23)(24)(31)(32)(33)
N_3	4	-6	2.7	3	-0.3	(19)(20)(21)(22)(23)(24)(25)(26)(27)	(4)(5)(6)(13)(14)(15)(34)(35)(36)
N_4	28	10	18.7	18	0.7	(28)(29)(30)(31)(32)(33)(34)(35)(36)	(7)(8)(9)(16)(17)(18)(25)(26)(27)

从表中可以看出，较快节点在完成 *LND* 下载任务的同时，从 *OND* 数据中分担较慢节点的下载任务。比如，数据块(25)(26)(27)位于节点 N_3 的 *LND* 中，但是最终却从节点 N_4 的 *OND* 数据中下载。

3.2 算法描述

目标:优化各节点的 S_i,使其接近或趋于0,达到各节点负载的均衡。

说明:对于任意两个节点 ZN、FN,如果 ZN 的 S_i 为正,并且 FN 的 LND 数据中仍有没有被 ZN 的 OND 数据分担的部分,那么 ZN 可以分担 FN 的下载任务,每分担一块则 ZN.S_i --,FN.S_i ++。如果数据被分担,则对该数据块(包括存储在其他节点中的)进行下载位置标记,其他节点中的该数据块不再进行处理。

输入:k,p,V_i,满足存储模型的数据。

输出:从各节点下载数据块的调度方案。

(1) 构建 S_i 为正的节点链表 *ZSi _ list*,对 *ZSi _ list* 进行降序排列。

(2) 构建 S_i 为负的节点链表 *FSi _ list*,对 *FSi _ list* 进行升序排列。

(3) **For** every node **FN** in *FSi _ list* deal from first//依次处理 FSi _ list 中的每个节点

(4) **For** every node **ZN** in *ZSi _ list* deal from first//让 ZSi _ list 中的节点分担 FSi _ list 中节点的下载任务

(5) **IF FN.** $S_i \geqslant 0$ **THEN** *FSi _ list*. **Move _ to _ Next, go to**(3)

(6) **IF ZN.** $S_i > 0$ **THEN**

令当前节点 **ZN** 的 OND 数据分担节点 **FN** 中 LND 的数据,并根据分担的块数更新 **ZN.** S_i,一次分担一块,并时刻判断以下两个条件:

(a) **IF FN.** $S_i \geqslant 0$ **THEN** *FSi _ list*.**Move _ to _ Next, go to**(3)

(b) **IF ZN.** $S_i \leqslant 0$ **THEN** *ZSi _ list*.**Move _ to _ Next, go to**(4)

(7) 在上述(1)-(6)的处理中,让速度较快的节点分担较慢节点的下载任务,只要两个链表有一个走到尾则跳出。

(8) 如果此时 FSi _ list 中仍然有 S_i 为负的节点 FN,那么从 ZSi _ list 头开始遍历,如果 ZSi _ list 中节点 ZN 的 OND 数据中没有被处理的数据块与 FN 中 LND 的没有被处理的数据块有交集,则令 ZN 分担 FN 的下载任务,直到没有交集为止,而不管 ZN 的 S_i 是否已经为负。

(9) 此时 FSi _ list 中的节点有两种情况,S_i 分别为正或非正。那么让 S_i 为正的节点的 OND 数据分担 S_i 为负的节点的 LND 数据,直到不能分但为止。

(10) 此时 ZSi _ list 中的节点存在两种情况,S_i 分别为正或非正。针对 ZSi _ list 中的节点,重复(1)-(6)步处理。完成后根据各节点数据块的下载标记进行下载。

4 实 验

实验环境:采用教育网内6台普通 PC 进行实验,操作系统使用 linux,在其中的5台机器上安装 GridFTP 服务器,每台服务器只允许一个客户端连接,对每次连接,随机地限定下载速度,分别为几十 k 到上百 k(实验中控制)。在其中一台机器上安装 GridFTP 客户端,在客户端运行调度算法。根据 k 值每台服务器允许启动两个 GridFTP Server 服务,则共可以模拟 10 台 Server。根据存储模型对 winrar 数据进行切割和存储。实验中速度单位为 k/s,存储与数据块单位为 M,为了叙述方便后文将省略单位。

文献[7-9]研究的各种调度算法基于 gridFTP 协议,基于平均预测速度,理论上各节点所能达到的最理想下载数据值为 $M \cdot V_i/(\sum_{j=0}^{k-1} V_j)(0 \leqslant i \leqslant k-1)$,实验中将对本文调度算法达到的下载时间与这些算法所能达到的理想下载时间进行比较(主要体现在指标 TRER),分析各种参数对算法性能的影响(实验1、2、3),并对所用存储空间进行对比(实验结果分析)。

定义 10 (超出理想下载时间百分比,*TRER*),令本文算法实际下载完成时间为 T_{re}(最后一个完成下载的节点所用时间),基于平均速度文献[7-9]理想下载时间为 T_{id},则令 $TRER = (100(T_{re} - T_{id})/T_i)\%$。$T_{id} = M/\sum V_i$。该指标是对本文算法与已有算法进行的性能比较。

定义 11 (单位速度方差 $D(V)$),方差除以速度的总和,即为速度均值 $E(V) = \left(\sum_{i=0}^{k-1} V_i\right)/k$,单位速度方差 $D(V) = (\sum_{i=0}^{k-1}(V_i - E(V))^2)/(k(\sum_{i=0}^{k-1} V_i))$。该指标用来评估速度分布对算法性能的影响。

4.1 实验1

主要验证 $D(V)$,V_{top},V_{ratio} 对 *TRER* 的影响。

令 $p = 1, k = 4$, $metasum = 15$,速度分布区间为(1,100),此时理论 $V_{ratio} = 90$。连续记录算法20次运行结果,统计分析速度单位方差 $D(V)$,极限速度比 V_{ratio},最大速度上限 V_{top} 对 *TRER* 的影响。数据如表2所示。

求出 $D(V)$的均值 $ED(V)$,将 $D(V) > ED(V)$ 的观测结果归为一组,小于的归为另一组。表 1 中序号 1-9 为 $D(V) > ED(V)$ 的实验。求出的 ***TRER*** 均值,为 4.40。

对 1-9 项和 10-20 项分别求 ***TRER*** 均值,分别为 7.40 和 1.87,分别大于和小于整体的均值 4.40。第 2、8 项里,实际 V_{top} 大于理论 V_{top}, 其相应的 ***TRER*** 也较大。

表 2 $D(V)$, V_{ratio}, V_{top}对 *TRER* 的影响

序号	$> D(V)$ 均值?	$D(V)$	V_i	实际 V_{ratio}	理论 V_{top}	实际 V_{top}	*TRER*	*TRER* 均值
1	Y	5.34	37,10,88,93	9.30	114.00	93.00	2.70	
2	**Y**	**4.47**	**12,23,55, 1**	**55.00**	**45.50**	**55.00**	**18.03**	
3	Y	3.07	85,18,89,76	4.94	134.00	89.00	0.00	
4	Y	4.48	1,77,67,75	77.00	110.00	77.00	0.00	
5	Y	4.48	95, 9,72,49	10.56	112.50	95.00	3.22	**7.40 (>4.40)**
6	Y	4.59	7,85,57,91	13.00	120.00	91.00	5.33	
7	Y	3.92	55,95,70,12	7.92	116.00	95.00	6.33	
8	**Y**	**5.89**	**12,24,76,10**	**7.60**	**61.00**	**76.00**	**29.44**	
9	Y	4.00	67,34, 1,34	67.00	68.00	67.00	2.37	
10	N	2.16	58,66,22,83	3.77	114.50	83.00	2.16	
11	N	0.46	36,28,41,21	1.95	63.00	41.00	2.50	
12	N	2.03	22,26,50,67	3.05	82.50	67.00	2.24	
13	N	1.99	33,94,89,73	2.85	144.50	94.00	2.48	
14	N	1.09	35,31,45,67	2.16	89.00	67.00	1.71	
15	N	0.75	19,10,32,24	3.20	42.50	32.00	0.00	**1.87 (<4.40)**
16	N	2.35	19,13,46,55	4.23	66.50	55.00	5.00	
17	N	2.47	64,22,66,90	4.09	121.00	90.00	0.00	
18	N	2.62	70,68,44,14	5.00	98.00	70.00	1.46	
19	N	1.94	85,29,46,64	2.93	112.00	85.00	2.99	
20	N	1.82	51,19,16,45	3.19	65.50	51.00	0.00	
平均		**3.00**					**4.40**	

4.2 实验 2

主要验证 p 和速度分布区间对 ***TRER*** 的影响。

令 $k = 10$,改变 p(1,2,3)和速度分布区间的范围(5-25,5-50,5-75),共有 9 种组合,对各组合分别观测 10 次,并记录数据。求出每种组合的 ***TRER*** 平均值,柱状图如图 1 所示。

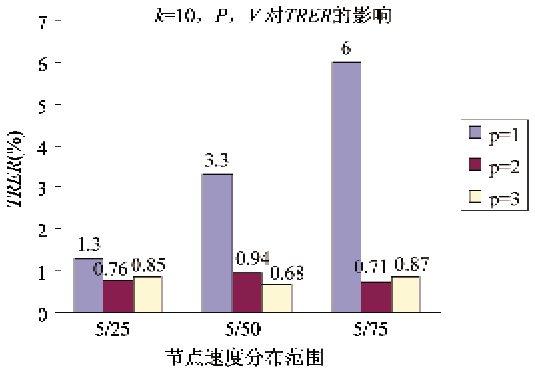

图 1 $k = 10$; p 和 V_i 分布区间对 *TRER* 的影响

令 $k = 2$, $p = 2$, 观测速度分布区间对 ***TRER*** 的影响。在每个速度分布区间内观测 10 次算法运行结果,求出其平均 *TRER*,曲线图如图 2 所示。

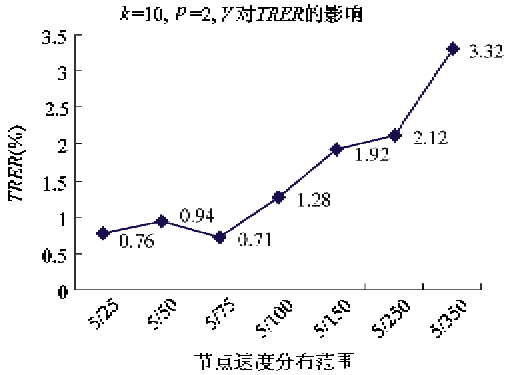

图 2 $k = 10$; $p = 2$;不同 V_i 分布区间对 *TRER* 的影响

由图 1 可以看出, $p = 1$ 时,速度的变化对 ***TRER*** 影响较大,但当 $p = 2,3$ 时随着随机速度区间的

增大,*TRER* 几乎不受影响,调度算法近似地等于理想下载速度。

由图 2 可以看出,当 $p=2$ 时,随着随机速度区间继续增大,*TRER* 逐渐增加。但是速度区间在 5-150 范围内时 *TRER* 都在 1 附近,效果较好。

4.3 实验 3

主要验证 k 和速度分布区间对 *TRER* 的影响。

令 $p=2$,$V_i \in (1,50)$,观测 k 值变化对 *TRER* 的影响。k 每取一个值,观测 10 次,求 *TRER* 均值。从图 3 可以看出,随着 k 值增大,*TRER* 逐渐增大。

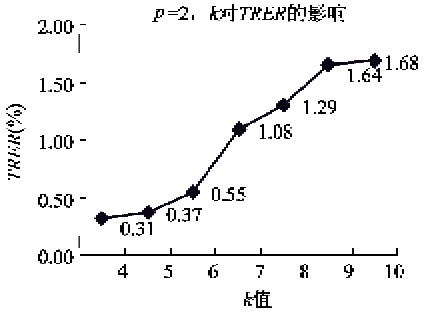

图 3 $p=2$;$V_i \in (1,50)$时;k 对 *TRER* 的影响

令 $p=2$,$V_i \in (5,50)$,观测 k 值变化对 *TRER* 的影响。K 每取一个值,观测 10 次,求 *TRER* 均值。从图 4 可以看出,随着 k 值增大,*TRER* 值变化微小,调度序列都近似等于理想速度。

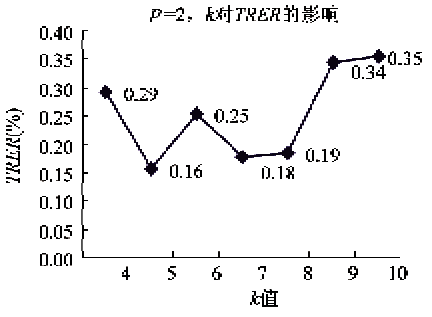

图 4 $p=2$;$V_i \in (5,50)$时,k 对 *TRER* 的影响

在 $V_i \in (1,50)$时,各节点速度之间的差值较大,最大理论 $V_{top}=50$;而 $V_i \in (5,50)$,各节点速度之间的相差不大,最大理论 $V_{top}=10$。

令每个节点的数据量为 300,为了保证 *metadata* 在 k 和整体数据量变化时不受影响,做如下处理。令 $k=4$ 时,$metasum=100$,根据式(1)反推出其他节点的 *metasum* 值。具体数据取值如表 3 所示。因为增加了节点,如果数据量不增加,则整体的下载时间必然缩短,导致 *TRER* 不准确,因此需要做特殊处理保证实验参数一致性。

表 3 k,M 与 *metasum* 取值

k	4	5	6	7	8	9	10
M, *metasum*	1200,100	1500,75	1800,60	2100,50	2400,43	2700,38	3000,33

4.4 实验结果分析

实验 1 显示出,在节点速度分布极不均匀并且最大、最小速度相差较大的情况下,都将可能使 *TRER* 增大。如果 V_{top}超出理论值,则必然引起 *TRER* 增大,而在 k,p 速度区间固定时,较大的 $D(V)$更容易使 *TRER* 增大;实验 2 显示出,在一个随机速度区间内,当 p 已经可以使 *TRER* 维持在较理想的情况下,继续增加 p 值不会降低 *TRER*;实验 3 显示出,速度区间较大时,k 值对 *TRER* 有影响,而一旦速度区间差异不大时 k 对 *TRER* 影响较小。

实际中数据网格各节点的速度差异不会非常大,实验中给出的速度范围是为了验证算法在速度极端分布时的性能。实际中只要根据速度的区间,合理配置 p 值,则一定可以达到较小的 *TRER*,使调度结果近似理想,而同时又可以大大节省存储空间,减少创建副本引起的网络流量。在实验中 $p=1$ 时,效果已经较好。由定义 7 计算 $p=1$,*SAV* 与 k 值取值如表 4。在 $k=3$ 时,$SAV=66\%$,而 $k=10$ 时,$SAV=20\%$。

表 4 $p=2$,k 与 *SAV* 关系

k	3	4	5	6	7	8	9	10
SAV	0.66	0.5	0.4	0.33	0.29	0.25	0.22	0.2

在实验中发现 1 附近的 TRER 较多情况下是由于数据的分块引起,各节点理想的下载量是建立在对整体数据任意分割的基础上,因此只要增加 *metasum* 则可以降低由此产生的实验误差,可以认为 *TRER* 在 1 附近时是近似达到了理想的调度。

5 结 论

本文针对数据网格并行下载中多副本创建引起的存储空间浪费和网络负载较大等问题,提出了一个数据存储模型和基于该模型的并行下载调度算法。证明和算例推演表明该模型既保证了数据的完整性又能使其具有多副本存储互为备份的可靠性,同时节省了大量的存储空间。实验表明基于存储模型给出的并行下载调度算法在给定合理的 p 值和速度分布合理时可以产生近似理想的下载调度序列。这说明只需要较少的冗余便可以达到近似完全副本并行下载算法可以达到的效果,与基于全副本的并行下载算法相比,使用本文存储模型、调度算法达到了并行快速传输、节约存储空间和降低网络流量的多重优化目标。

参考文献

[1] 陈磊, 李三立. 数据网格中一种填空式副本分配算法. 电子学报, 2006,34(11):1-4

[2] Liu P F, Wu J J. Optimal replica placement strategy for hierarchical data grid systems. In: Proceedings of the 6th IEEE International Symposium on Cluster Computing and the Grid. Washington D.C., USA: IEEE Computer Society, 2006. 1-4

[3] Tim H, David A. A unified data grid replication framework. In: Proceedings of the 2nd IEEE International Conference on e-Science and Grid Computing. Washington D. C., USA: IEEE Computer Society, 2006. 1-8

[4] Feng J, Cui L L, Glenn W, et al. Toward seamless grid data access: design and implementation of GridFTP on .NET. In: Proceedings of the 6th IEEE/ACM International Workshop. Washington D.C., USA: IEEE Computer Society, 2005. 1-8

[5] Vazhkudai S. Enabling the co-allocation of grid data transfers. In: Proceedings of the 4th International Workshop on Grid Computing. Washington D. C., USA: IEEE Computer Society, 2003. 1-8

[6] William A, John B, Rajkumar K, et al. The globus striped GridFTP framework and server. In: Proceedings of the 2005 ACM/IEEE SC/05 Conference. Washington D. C., USA: IEEE Computer Society, 2005. 1-11

[7] Bhuvaneswaran R S, Yoshiaki K, Naohisa T. Dynamic co-allocation scheme for parallel data transfer in grid environment. In: Proceedings of the 1st International Conference on Semantics, Knowledge, and Grid. Washington D.C., USA: IEEE Computer Society, 2006. 1-6

[8] Vazhkudai S. Distributed downloads of bulk, replicated grid data. *Journal of Grid Computing*, 2004, 2:31-42

[9] Gaurav K, Umit C, Tahsin K, et al. A dynamic scheduling approach for coordinated wide-area data transfers using GridFTP. In: Proceedings of the 22nd IEEE International Parallel and Distributed Processing Symposium (IPDPS 2008), Miami, Florida, USA, 2008. 1-12

A storage model and its corresponding parallel download scheduling algorithm for data grid systems

Qu Mingcheng, Wu Xianghu, Liao Minghong*, Yang Xiaozong, Zuo Decheng

(School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001)

(*Software School, Xiamen University, Xiamen 361005)

Abstract

To solve the problems that creating a number of copies in data grid systems can bring great overheads in data storage and network traffic and a variety of parallel download algorithms based on the GridFTP protocol can not eliminate the influence of multi-copy on the storage space and network traffic, the paper puts forward a storage model for data grid systems to guarantee data integrity, storage reliability and reduce storage space, and presents a parallel download scheduling algorithm based on the model and the GridFTP protocol. The experimental results show that the proposed algorithm can achieve a better effect in fast parallel transmitting, storage space saving and network traffic reducing.

Key words: data grid, storage model, parallel download scheduling algorithm, GridFTP