有服务质量保证的数据密集型网格应用管理研究*)

石宣化 金 海
(华中科技大学计算机学院 集群与网格计算湖北省重点实验室 武汉 430074)

摘 要 在网格中数据密集型应用的管理由于需要综合考虑多种资源属性而变得非常复杂,本文针对该问题提出了一种有服务质量保证的数据密集型网格应用管理方法,该方法通过两个方面来保证服务质量,首先是通过基于数据可靠性的副本管理策略提高数据的可用性,其次采用基于任务截至期限的任务调度策略来动态调度任务。最后,给出模拟试验测试以及实际网格平台与实际应用的测试,测试结果表明本文提出的方法较其它方法有很大提高。

关键词 网格,服务质量,数据密集型,副本管理,截止期限

Data-Intensive Applications Management Framework with QoS Guarantee in Grid Environments

SHI Xuan-Hua JIN Hai
(Cluster and Grid Computing Lab, Huazhong University of Science and Technology, Wuhan 430074)

Abstract Effectively management of data-intensive applications in grid environments is challenging, due to a need to consider for both computational resources and data storage resources. This paper proposes a management framework for such applications with QoS Guarantee. This framework includes two parts, one is replica management based on availability of data, and the other is deadline-based job scheduling model. In the end, a simulation evaluation and two real applications evaluation over HUSTGrid are presented, the results of the evaluations show that this framework is suitable for data-intensive applications, and the performance of this scheduling method is better than the related methods.

Keywords Grid, QoS, Data-intensive, Replica management, Deadline

1 引言

网格数据密集型应用处理应用产生的中间以及最终结果数据达到TB乃至PB级,在这类应用中,用户需要处理分布的多种类型资源同时又要满足不同用户的服务质量标准,因而管理这类应用极具挑战性。针对数据密集型应用的网格平台通常称为数据网格(DataGrid)[1],目前多数网格平台都属此类。

过去对分布式系统的任务调度研究较多,但多数都是针对计算任务[2]。在Condor中系统考虑怎样调度任务在闲散的资源上以构建高吞吐率计算处理平台[3],EU-DataGrid上的资源代理是由Condor发展而来[4]。Thain等人给出了一种基于I/O社区的方式来调度作业与数据,但这种方法没有解决数据策略上的问题[5]。Nimrod-G给出了一种基于经济模型的任务调度方法,但该模型没有考虑到应用对平台性能的影响。类似的研究还有AppLes, GrADS等[6]。Ranganathan等人一种在数据网格环境下的任务调度[1],Park等人给出了一种综合考虑数据以及计算能力的网格任务调度方法[7],但以上两个方面的研究都是采用贪婪算法来调度任务。对于数据密集型应用[8],贪婪算法最大的缺点是导致网格上负载极度不平衡,最终导致大量的资源空闲。

针对数据密集型应用调度研究的需要,本文给出一种有服务质量保证的数据密集型应用的管理框架,该框架综合考虑数据密集型应用的特点,充分利用各种资源保证任务调度的服务质量,同时兼顾任务调度对系统总体性能的影响。

2 数据密集型应用管理框架

在以下的小节中我们首先给出一些参数说明以及一些相关的计算方法,随后给出副本的创建算法,最后给出系统的调度算法。

2.1 相关参数说明

为了算法描述的需要,本小节首先给出一些相关的参数说明。

定义1 作业响应时间是指从用户提交作业到最终用户获得需要的服务的时间,记作RT_i。

下面给出相关的参数的说明,具体见表1。

表1 数值试验输入参数

参数	表示意义
N_i	站点i上可用的计算节点数
D_{input}	任务输入数据的大小
D_{output}	任务输出数据的大小
D_{code}	任务应用程序的大小
$BW_{uan(i-j)}$	站点i与站点j之间的带宽
Tr_{ij}	从站点i到站点j的传输时间
Re_i	读取输入数据的时间
WT_i	写入输出数据的时间
PT_i	任务在i站点处理需要的时间
P_i	i站点处理器的处理能力

*)本文研究受国家自然科学基金重大专项(No. 90412010)和中国教育科研网格计划ChinaGrid资助。**石宣化** 博士,研究方向为集群与网格计算、容错、SOA、网络安全与网格安全;**金 海** 教授,博士生导师,研究方向为计算机系统结构、并行与分布式处理、集群与网格计算、高性能与外存储系统、无形计算与无形存储、网络安全与网格安全。

2.2 任务运行时间的计算方法

用户提交一个任务请求时，根据当前的应用部署情况，系统可以对该应用作如下的几种处理情况：(1)输入输出数据都在本地，本地的计算能力能够满足应用需要，最后该应用在本地执行；(2)本地的计算能力不能满足应用需要，虽然本地有输入数据的副本，需要应用使用本地数据但是在远程执行；(3)应用在本地执行，但本地没有输入数据集的副本；(4)应用在远程执行，同时该站点也有输入数据集的副本；(5)应用在远程执行，但需要从另外的站点传输输入数据集。

假设在每一个站点可用的资源都有其他作业在等待运行，这个队列长度为 Q_i，如果没有等待的任务，则 Q_i 为 0。Q_c 为该计算任务本身需要的计算量。那么任务在站点 i 运行需要处理的时间可以按照公式(1)来计算：

$$PT_i=\frac{Q_i}{P_i\times N_i}+\frac{Q_c}{P_i\times N_i} \tag{1}$$

数据量大小为 D_{code} 数据网络上的传输时间可以按照公式(2)来计算：

$$Tr_{ij}=\frac{D_{code}}{BW_{uan(i-j)}} \tag{2}$$

按照上面提到的五种情况，输入数据的传输时间可以按照公式(3)来计算：

$$ReT_i=\begin{cases}0, i=j\\ D_{input}/BW_{uan(i-j)}, i\neq j\end{cases} \tag{3}$$

也就是，如果输入数据在 i 节点运行，同时在 i 节点上输入数据的副本，则我们认为在本地读取输入数据的时间为 0，在其他情况下，都需要从别的站点读取输入数据，也就需要一个广域网上的数据传输过程。

按照输入数据类似的处理方法，我们可以得到输出数据的计算方法，如公式(4)：

$$WT_i=\begin{cases}0, i=j\\ D_{output}/BW_{uan(i-j)}, i\neq j\end{cases} \tag{4}$$

一个数据密集型应用的响应时间分为三个部分，任务的计算时间与数据的传输时间以及应用程序的传输时间，如公式(5)所示：

$$RT_i=ReT_i+PT_i+WT_i \tag{5}$$

综合公式(1)～(5)，任务在最终的相应时间如公式(6)所示，公式(6)表示网格用户通过站点 i 提交任务，任务最终在 j 站点运行，而应用的输入数据集在 k 站点：

$$RT_i=\frac{D_{code}}{BW_{uan(i-j)}}+\frac{D_{input}}{BW_{uan(j-k)}}+\frac{D_{output}}{BW_{uan(k-i)}}+\frac{Q_j}{P_j\times N_j}+\frac{Q_c}{P_j\times N_j} \tag{6}$$

2.3 副本管理模型

副本管理包括两方面：确定副本数与副本调度。

1)确定副本数目

为了讨论方便，这里首先对副本的服务能力作一个等级划分 L_i，副本的可靠性 A_i，按照 $A_i=L_i/10$ 计算。副本服务能力等级 L_i 定义成一个介于 0～10 之间的整数，定义如关系式(7)所示：

$$L_i=\left\lfloor 10\times\frac{S_i+D_i\times E_d}{R_i}\right\rfloor \tag{7}$$

公式(7)中 R_i 表示该副本被引用的总次数，S_i 表示该副本成功服务的次数，D_i 表示该副本降级服务的次数，不难得出服务失败的次数 $F_i=R_i-S_i-D_i$。E_d 表示降级服务指数，是一个位于 0-1 之间的数。该数字越大，表示降级服务对整体的影响越小，当 E_d 为 1 时，说明视降级服务和成功服务有一样的效果。当 E_d 为 0 时，则视降级服务和失败服务具有一样的效果。这样可得出 L_i 是一个 0～10 之间的数，这也是目前本系统采用的 10 级副本服务等级评估指数。

按照上面给出的副本可靠性，可以按照公式(8)计算 N_r 个副本出错的概率：

$$p_{un}=(1-A_i)_r^N \tag{8}$$

这样 N_r 个副本由一个副本可以提供服务的概率就是 1 $-p_{un}$，在副本定位服务搜索成功概率 p_d 下，一个副本能够成功为网格应用提高服务的概率如公式(9)所示：

$$p_s=p_d*(1-(1-A_i)^{N_r}) \tag{9}$$

在副本创建之前，首先给出一个副本可用的阈值 A_T，根据公式(9)，在满足 $p_s\geqslant A_T$ 的约束条件下就可以计算得出系统需要的副本数目 N_r。

2)副本调度

副本调度主要包括：副本创建、放置与销毁。本文采用一种收益-代价差值最大化的方法来确定副本的放置位置。首先系统信息服务搜索可以放置副本的候选节点，这些节点满足三条：没有放置该副本数据；有足够的存储空间；以及一定的网络带宽。在候选节点选定后，以副本收益与副本代价差值最大的目标，按照如下算法选择放置副本的放置位置。首先给出副本收益函数：

$$B_{Ri}=Trans(F, N_i, App)-Trans(F, N_h, App) \tag{10}$$

其中，$Trans(F, N_i, App)$ 表示应用 App 从新副本节点 N_i 获取数据 F 的传输代价，$Trans(F, N_h, App)$ 表示应用从已有数据集的节点 N_h 获取数据 F 的传输代价。

副本的创建代价按照公式(11)计算：

$$C_{Ri}=S(F, N_i)+Trans(F, N_i, N_h) \tag{11}$$

其中 $S(F, N_i)$ 表示在 N_i 节点上存储副本的代价。

最后系统的目标函数为：

$$F_{Target}=\mathrm{Max}\sum_i^r(R_{Ri}-C_{Ri})) \tag{12}$$

副本创建：

a)置创建副本序号 $i\leftarrow 1$；

b)$i=r+1$ 结束，否则转 c)；

c)选择可以提供服务的副本序号，$h=i-1$；

d)$h=i$ 转 g)；

e)计算 $B_{Ri}-C_{Ri}$，选择具有最大值的节点部署副本；

f)$h\leftarrow h+1$，转 d)；

g)$i\leftarrow i+1$，转 b)。

在系统按照副本可靠性创建完应用需要的副本后，系统可以按照网络带宽、节点服务能力等选择需要的副本作为应用的输入或者输出数据。在副本使用的过程中系统定期监控副本的状态，从而动态地调整副本的创建与消耗。副本的销毁主要按照 LRU 算法来处理。

首先对于这些创建的副本计算副本访问率：

$$AR_i=N_{access}/(T_{current}-T_{stored}) \tag{13}$$

销毁具有最小 AR_i 值的副本，同时满足以下约束条件：

$$Disk_{Total}\times f_e>Disk_{Avail} \tag{14}$$

$Disk_{Total}$ 表示的网格系统总的存储容量，$Disk_{Avail}$ 表示目前系统中可用的存储容量，f_e 表示系统中副本销毁频率以及多个副本中需要销毁的数目。

2.4 作业调度算法

在用户提交作业后，系统首先检查用户任务的截止时间

D,如果 D 为 0,说明用户对于系统服务质量没有要求,系统就按照可用资源列表随机选择资源提交任务。如果 D 不为 0,使用如下的方法处理任务。

首先设置目标函数 $Diff$:

$$Diff=T_{target}-RT_i \tag{15}$$

其中 T_{target} 是完成任务的目标时间,可以按照下面的方法来计算:

$$T_{target}=T_{untildeadline}*Opt \tag{16}$$

Opt 是资源预测方面的准确性参数,数值在 0～1 之间。$T_{untildeadline}$ 可以按照公式(17)的方式来计算。

$$T_{untildeadline}=D-now \tag{17}$$

now 表示是当前的时间。

以上几个公式中目标函数 $Diff$ 表示任务能否在截止期限前完成任务,如果 $Diff$ 大于 0,表示任务能够在截止期限之前完成,同时调度器选择拥有最小 $Diff$ 值的资源完成任务,调度的过程包括应用代码、数据以及最终任务的计算等过程。

如果 $Diff$ 小于 0,说明在目前的系统运行状况下,调度器不能找到在截止期限之前完成该任务的资源。这种情况本文中通过采用资源预留的方法来处理这种情况,从而提高最终系统对于网格任务的服务能力。

我们采用 GARA 对计算资源以及网络资源进行预留[9]。通过 GARA 预留资源后,各站点的处理能力以及网络的传输能力可以用下面的公式来计算。调度器根据新的资源处理能力以及网络传输能力重新计算调度目标函数重新调度任务。

$$P_i=P_i+P_{reserve} \tag{18}$$

$$BW_{wan(i-j)}=BW_{wan(i-j)}+BW_{wan(i-j)reserve} \tag{19}$$

为了在资源服务能力的预测中提高准确性,调度器在目标函数上考虑其他因素。首先调度器采用修正的负载值的方法来重新评估系统系统的处理能力。在调度完几个作业后,系统的负载值按照如下的公式来修正,然后公式(1)中按照修正过的值来就算目标函数。

$$Q_{corrected}=Q_i+NUM_{jobs}*pload \tag{20}$$

其中 $pload$ 是调整参数,NUM_{jobs} 是已经提交并调度了的作业。在本文后面的具体应用处理的性能评测中 $pload$ 取值为 1。

在应用的调度中,由于资源服务能力预测的不准确性,导致最后任务不能再规定的时间内完成,本文中通过对于任务的监控,在任务中应用代码数据传输完成而任务计算开始之前,重新评估系统能否在此时间期限之前完成任务,并按照如下两个约束条件重新提交任务到新的节点:

$$\begin{cases} T_{untildeadline}<RT \\ N_{re\text{-}submission}\leqslant N_{\max} \end{cases} \tag{21}$$

上面的约束条件中,$N_{re\text{-}submission}$ 是重新提交任务的次数,$N_{\max}$ 是最多允许的提交次数,由于数据密集型应用里,数据传输的代价太大,我们选择 $N_{\max}$ 的值为 1,对于其他类型的应用,$N_{\max}$ 的值可以取得较大。

3 性能分析

本文采用的两种评估方法来分析系统性能:仿真试验评估方法以及实际系统数据分析。

3.1 仿真试验评估与分析

系统模拟测试是在集群与网格计算湖北省重点实验室的 PC 集群上完成的,集群配置如下:16 个节点,各节点通过 100M 以太网连接,每一个节点配置是单 PIII 1G 处理器,512M 内存,操作系统是 RedHat Linux 9.0。仿真评测两个方面:副本管理算法性能评估以及任务调度性能评估。

3.1.1 副本管理算法性能评估与分析

按照表 2 中的参数来评测系统的副本管理性能。

表 2 副本管理仿真试验参数

参数	数值 1	数值 2
站点数目	100	200
总存储容量	100TB	200TB
系统可用存储容量	60TB	120TB
单数据可靠性	0.6	0.3
输入文件大小	100GB	-
带宽	1000Mb/s	100Mb/s
副本定位成功概率	0.95	0.9

首先测试副本的可靠性与副本数目的关系,按照表 2 中给出的参数,测试不同应用可靠性需求下的副本创建数目。

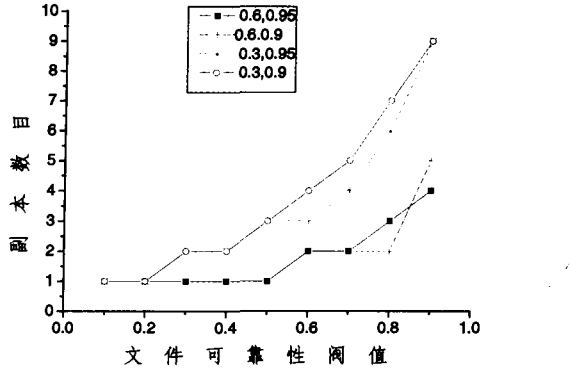

图 1 不同节点可靠性以及不同副本定位能力下的副本数目

图 1 给出了不同单数据的可靠性与不同副本定位能力下在应用规定的副本可靠性范围内副本的数目。图 2 中 0.6,0.95 曲线表示单数据可靠性为 0.6,副本定位成功概率为 0.95 下副本数,其他表示意义相应类推。系统需要根据目前的网络状况、用户行为等确定单数据的可靠性,从图 1 可以看出单数据可靠性是影响副本数目的最重要因素。而副本定位能力相对而言对副本数目的影响较小,这也是因为目前一般的系统中副本的定位搜索能力都研究得较为成熟。

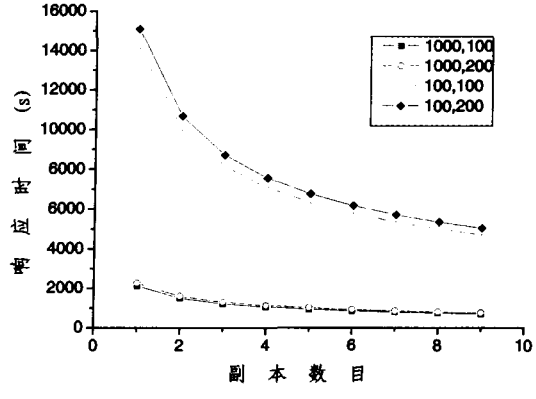

图 2 不同带宽以及不同节点数目下副本数目对响应时间的影响

按照表 2 中给出的输入数据的大小以及带宽大小,图 2

中给出了副本数目对于网格应用处理时间的影响，副本数目就按照图 1 中给出，输入文件的大小按照表 2 中给出的 100GB，按照带宽不同以及输入数据不同来测试平均应用等待时间。

图 2 中 1000，100 的曲线表示的是 1000Mb/s 的带宽、100 个节点情况下的数据传输响应时间，其他表示意义相应类推。从图 2 可以看出带宽对于数据传输的传输具有决定性影响；同时副本数目对于数据传输时间影响较之节点数目对数据传输时间要大，这可以看出本文中给出的副本创建策略的有效性。

3.1.2 任务调度算法性能评估与分析

本文通过两个方面来评估与分析调度算法性能：调度过程中用户给定执行时间范围的任务的失败率（也就是未能在规定时间范围内完成该任务）与输入数据大小的关系；任务失败率与重新提交次数的关系。

首先仿真测试重新提交次数 N_{max} 与任务失败率的关系，按照输入数据为 10GB、网络带宽按照 100Mb/s、应用程序大小 200MB、输出文件为 1GB、任务的计算时间为 2000s 的情况来模拟。评测过程中主要比较本文中给出的算法在不同的 *opt* 值下与贪婪（Greedy）算法的性能比较（贪婪算法是希望每次任务执行时间最短），如图 3 所示。

图 3 中 0、1、2、3、4 分别表示系统允许重新提交次数的最大值 N_{max} 下任务的失败率。其中横坐标上的 Greedy 表示 Greedy 算法在不同允许重新提交次数下的失败率，0.5、0.6 等表示不同 *opt* 值下的任务失败率。

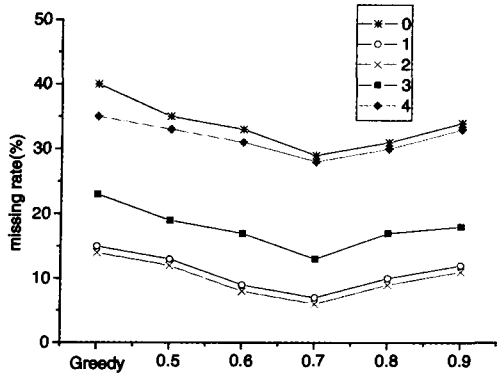

图 3 不同 *opt* 以及不同重提交任务次数下对任务失败率的影响

通过分析图 3 可以看出，系统能否允许任务重新提交对调度的性能影响很大，在不允许重新提交的情况下，也就是 N_{max} 为 0 的情况下，不管是 Greedy 算法还是本文中给出的算法，任务的失败率都在 25% 以上。同时，从图 3 还可以看出在数据密集型应用中允许重新提交的次数过大，系统的效率反而越低，图 3 中当重新提交次数可以是 3 次以上时，系统性能反而下降，这种现象是数据密集型应用重新调度任务需要进行大量的传输，最后导致系统长时间在等待传输数据而导致机器空闲。

按照如下参数评测任务失败率与输入数据之间的关系：系统允许的重新提交作业的次数 N_{max} 为 1，网络带宽按照 100Mb/s、应用程序大小为 200MB、输出文件为 1GB、任务的计算时间为 2000s。我们按照 *opt* 以及输入数据大小两个变量来模拟，模拟结果如图 4 所示。

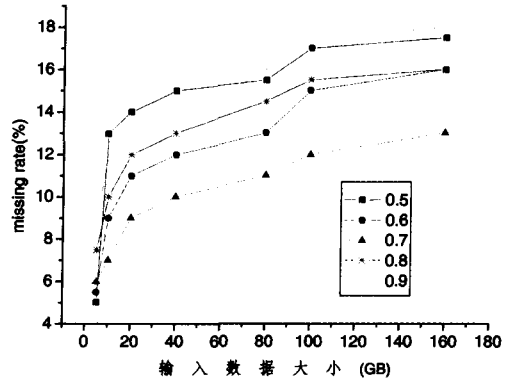

图 4 不同 *opt* 以及不同输入数据大小对任务失败率的影响

从图 4 中可以看出，随着输入数据的增大，任务的失败率增高，尤其是在 5GB 到 10GB 之间，在输入数据达到 20GB 以上时任务失败率升高幅度并不是很大；在输入数据较小的情况下，*opt* 值较小时任务的失败率较低，当数据较大时，*opt* 在 0.7 的时候任务的失败率较低，其主要原因是在输入数据较小的情况下，数据传输占用带宽的情况预测更难，最终预测的准确性也就越差，因此导致上面的情形。

3.2 实际系统评测

实际系统评测基于华中科技大学校园网格平台（HUSTGrid），华中科技大学校园网格包括 3 个网格节点，它们是集群与网格计算实验室（CGCL）、水电仿真中心（NHEESL）以及高性能计算中心（武汉）（HPCC），这些网格节点基本上都是由集群组成，校园网络带宽是 1000Mb/s。在集群与网格计算实验室拥有三个集群，水电仿真中心拥有一套集群，另外湖北、武汉高性能计算中心拥有两套集群。具体硬件具体配置如表 3 所示。

表 3 实验评测硬件环境具体参数

站点	集群名称	配置
集群与网格计算实验室	IA-64 集群	IA-64 1GHz，73GB HD，6GB Mem，2 节点
	PC 集群 1	PIII 1GHz，40GB HD，512MB Mem，16 节点
	PC 集群 2	PIII 1GHz，40GB HD，512MB Mem，16 节点
水电仿真中心	Alpha 集群	alpha 500MHz，20GB HD，376MB Mem，5 节点
国家高性能计算中心（武汉）	Dawning3000	4xPowerPC375MHz，9GB HD，2GB Mem，3 节点，500GB RAID
	Dawning2000	Power604E 200MHz，2GB HD，128MB Mem，8 节点

在校园网格上测试两个应用：基因匹配程序（FASTA），该应用需要分析大量的数据库中存储的数据；视频转换（Video Conversion）。在表 4 中给出了两个应用的详细参数说明。我们在校园网格上运行了 100 个 FASTA 作业以及 1000 个视频转换任务来测试我们的调度算法。

与仿真评测中类似，首先比较贪婪算法与本文中给出的算法在 N_{max} 值为 1、不同 *opt* 值下的性能比较，性能比较还是通过任务失败率来比较。比较结果如图 5 所示。

表4 FASTA应用以及视频转换应用参数

参数	FASTA	Video Conversion
输入数据大小	500MB	10GB
输出数据大小	20MB	650MB
应用程序大小	1MB	1MB
标准执行时间 (P4 1GHZ CPU, 512MB Memory)	44856 sec.	18007 sec.

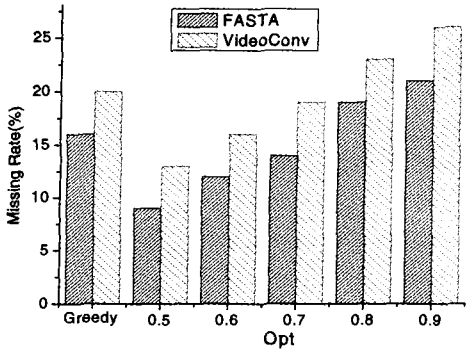

图5 Greedy算法与不同 *opt* 下本算法对于具体应用的任务失败率的比较

从图5可以分析得出以下结论:(1)输入数据的规模越大导致更高的任务失败率。这与前面仿真试验的结论一致,原因就是因为校园网格的带宽比较局限,尤其是校园网是属于共享信道,网络状况变化较大,导致最终系统对于执行时间以及资源状况的预测出现偏差,同时输入数据的增大也到更多的计算资源在等待作业从而导致更多的资源空闲。(2)贪婪算法与本文中的算法在 *opt* 为0.6、0.7左右时的性能相当,这个与仿真试验的结果有点不同,原因是仿真实验室中我们假设系统资源基本上趋于无穷大,而实际评测中系统资源是有限的。(3)当 *opt* 值较小时,任务的失败率较低一些,原因在仿真测试中已经说明。

接下来测试副本对于任务失败率的影响,由于校园网格系统资源本身的局限,我们通过如下的方式来部署副本:(1,CGCL IA-64集群),(2,CGCL IA-64集群,CGCL集群1),(3,CGCL IA-64集群,CGCL集群1,CGCL集群2),(4,CGCL IA-64集群,CGCL集群1,CGCL集群2,HPCC Dawning3000),(5,CGCL IA-64集群,CGCL集群1,CGCL集群2,HPCC Dawning3000,NHEESL)。按照 *opt*=0.7来进行试验评测。评测结果如图6。

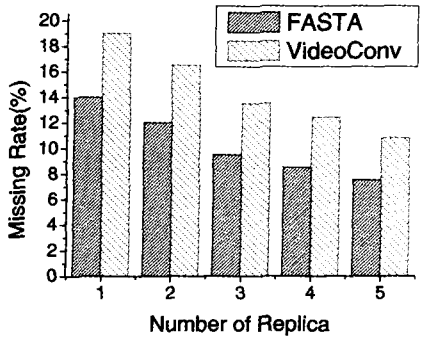

图6 *opt*=0.7的情况下副本数目对于具体应用的任务失败率的比较

从图6可以看出:(1)随着副本数目的增加,任务失败率明显下降。这是数据密集型应用的最明显特征,因为副本的增加减少了数据传输的时间,也从而降低了计算资源空闲的时间,另外传输时间的减少也提高了系统对于资源性能以及任务执行时间的预测准确性,进一步提高了调度的效率。(2)如果副本在处理能力比较强的站点上,任务失败率下降的更多。这个原因很直接,因为任务调度过程中处理能力强的站点处理作业较多,因而在这些站点上如果具有输入文件的副本将直接减少数据传输的时间,从而提高了调度效率。

综合仿真试验与实际系统的评测结果,本文提出的管理方法对于数据密集型应用比较有效,其中系统能够保证任务90%数据密集型的应用在用户规定的时间范围内完成,从仿真试验的结果也可以看出对于输入数据较小的应用,我们给出的算法任务的失败率更低,可以保证用户的任务95%左右能够在规定时间范围内完成。

小结 本文给出了具有服务质量保证的数据密集型应用的管理框架,该类应用调度的处理方法。该框架包括两个方面:优化的副本管理算法与基于截至期限的任务调度算法。其中基于截止期限的任务调度算法综合考虑了副本管理以及计算任务的调度。同时,任务的调度中集成了资源预测以及资源预留技术。对于上述算法,本文给出了仿真测试与实际系统测试两种测试方案。按照此测试方案,对测试结果进行详细分析,得出了影响该类应用的调度的关键因素以及改进方法。同时,从测试结果可以看出本文中给出的算法非常有效。

参考文献

1 Chervenak A, Foster I, Kesselman C, et al. The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets. Journal of Network and Computer Applications, 2001

2 Ranganathan K, Foster I. Decoupling Computation and Data Scheduling in Distributed Data-Intensive Applications. In: Proceedings of 11th IEEE International Symposium on High Performance Distributed Computing (HPDC-11), Edinburgh, Scotland, July 2002

3 Condor. http://www.cs.wisc.edu/condor/

4 Data Grid Project WP1. Definition of Architecture, Technical Plan and Evaluation Criteria for Scheduling, Resource Management, Security and Job Description. Datagrid document DataGrid-01-D1.2-0112-0-3, 14/09/2001

5 Thain D, Bent J, Arpaci-Dusseau A, et al. Gathering at the Well: Creating Communities for Grid I/O. In: Proceedings of Supercomputing 2001, Denver, Colorado, November 2001

6 Application Level Scheduling (AppLeS). http://apples.ucsd.edu/

7 Park S M, Kim J H. Chameleon: A Resource Scheduler in A Data Grid Environment. In: Proceedings of the 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid, Tokyo, 2003

8 Takefusa A, Casanova H, Matsuoka S, et al. A Study of Deadline Scheduling for Client-Server Systems on the Computational Grid. In: Proceedings of 10th IEEE International Symposium on High Performance Distributed Computing (HPDC-10), 2001

9 GARA. http://www-fp.mcs.anl.gov/qos/