基于任务复制的网格任务调度算法*)

林剑柠　吴慧中　陈学勤
(南京理工大学计算机系　南京 210094)

摘　要　网格中资源之间存在着通信延迟,通过任务复制的冗余,可以减少任务之间的通信开销,缩短整个计算程序的计算时间。目前网格中的任务调度算法基本上是没有考虑任务复制的;而基于任务复制调度算法往往会产生过多的复制任务,增大系统开销,甚至有可能延迟计算时间。由于基于任务复制的任务调度是一个 NP 问题,因此本文提出了一种基于任务复制的网格资源调度算法,以减少调度长度为主要目标、减少任务复制量和资源占用量为次要目标。该算法在调度长度和任务复制数量以及占用资源数量方面都等于或优于其它算法。

关键词　任务复制,任务调度,冗余任务,DAG 图

A Task Duplication Based Scheduling Algorithm in Grid Computing Systems

LIN Jian-Ning　WU Hui-Zhong　CHEN Xue-Qin
(Department of Computer, Nanjing University of Technology, Nanjing 210094)

Abstract　Grid computing is a new computing-framework to meet the growing computational demands. Computational grids provide mechanisms for sharing and accessing large and heterogeneous collections of remote resources. However, task scheduling is one of the key elements in the grid computing environment, and an efficient algorithm can help reduce the communication time between tasks. So far, the task scheduling algorithms in the grid computing environment have not been based on task duplication. But, the scheduling algorithms based on task duplication will generate too many task replications, which will enlarge the system loads and even add the makespan. As optimal scheduling of tasks is a strong NP-hard problem, this paper presents a scheduling algorithm based on genetic algorithm and task duplication, whose primary aim is to get the shortest makespan, and secondary aim to utilize less number of resources and duplicate less number of tasks. The chromosome coding method and the operator of genetic algorithm are discussed in detail. The relationship between subtasks can be obtained through the DAG. And the subtasks are ranked according to their depth-value, which can avoid the emergence of deadlock. The algorithm was compared with other scheduling algorithm based on GAs in terms of makespan, resource number and task replication number. The experimental results show the effectiveness of the proposed algorithm to the scheduling problem.

Keywords　Task duplication, Task scheduling, Redundant task, DAG

网格计算是伴随着互联网而迅速发展起来的专门针对复杂科学计算的新型计算模式。为了能充分利用网格中的资源,一个大的应用任务分解成多个具有一定约束关系的子任务,这些子任务向资源管理系统提出资源请求,尽可能地使多个子任务并行执行,从而使得所给定任务能在最短的时间内完成。在通常的情况下,子任务的数量多于资源的数量。因此,为了提高资源的使用效率,同时使计算任务在尽可能短的时间内完成,资源调度是一个极其繁琐复杂的问题。

已经有人对资源调度算法做了大量研究。Vincenzo 介绍了一种基于遗传算法的资源调度算法[1,2],其目的是为了尽可能地提高资源的使用率和吞吐量。另外,Ajith Abraham 等人介绍了模拟退火等进化算法在网格资源调度中的应用[3],XU 等人介绍了蚁群算法[4]的应用。但是他们没有考虑到任务的可复制性。无任务复制的调度策略更多的是考虑如何合理地安排关键路径上的任务节点,尽可能地缩短关键路径上的执行时间,但是它们无法缩短任务节点通信带来的延迟。并行的计算任务之间的通信延迟是影响整个计算程序完成时间长短的重要因素之一。基于复制任务的调度算法,通过把任务冗余分配到一个或多个计算资源上,能够减少任务彼此间通信负荷。利用该方法,不仅要考虑任务之间的制约关系,还要考虑复制哪些任务、在哪个资源上复制任务,因而该问题也是一个 NP 完全问题。通过任务在不同资源上的复制可以将两个通信时间比较长的子任务安排在同一个资源上执行,这样就可以将它们之间的通信延迟大大降低。以 TSA[5]和 OSA[6]为典型代表。由于任务分配属于 NP 问题,因此可以采用遗传算法解决[7]。但是这些算法均假设所有子任务在不同处理器上的执行开销相等,所用的 DAG 图中的权值固定。而在异构的网格中,相同的计算任务在不同资源上执行开销一般是不同的。而且,它们在该任务复制过程中都没有考虑到任务复制过程中带来的任务的过分冗余,导致任务过多地复制,不仅提高了复制任务带来的开销,而且有可能增加程序的执行时间;另外,网格中的资源一般都需要费用,因此减少占用的网格资源数量也是一个关键问题,而该算法没有考虑节约使用资源数量。因此本文针对以上缺陷,提出了一个符合网格资源特点的基于任务复制的遗传调度算法,以尽可能地减少程序执行时间为主要目标、尽可能减少资源占用数量

*)国防预研课题基金项目(51404020303BQ0220)和南京市重点攻关基金。林剑柠　博士研究生,研究方向:仿真网络。

和复制任务的数量为次要目标，消除过分冗余复制任务，减少占用的网格资源。

1 调度模型

这里我们把一组有序任务表示为一个有向无环图(DAG)，从而将对子任务的调度问题转化为对调度 DAG 图的问题。我们这里将特定的子任务表示为如下 DAG 图的形式：$G=\{V,E,\tau,C\}$。这里 $V=\{n_i \mid n_i$ 是有序任务集合，$i=1,2,3,\cdots,v, v=|V|$ 任务总数$\}$，$E=\{e_{ij} \mid e_{ij}$ 表示任务 n_i 到 n_j 的边$\}$，如果 $e_{ij}=1$，则说明 $n_i \rightarrow n_j$，即 n_i 是的 n_j 父任务。由于 G 是有向无环图，因此 e_{ij} 和 e_{ji} 只能有一个为 1。在网格环境下，不同子任务在异构资源上的执行开销定义为：$\tau=\{\tau_{ij} \mid$ 子任务 n_i 在资源 r_j 的执行开销，$1\leqslant i\leqslant|V|, 1\leqslant j\leqslant|R|\}$。不同子任务之间的通信开销定义为：$C=\{C_{ij} \mid$ 子任务 n_i 和子任务 n_j 之间的通信延迟，其中 $e_{ij}\in E\}$。任务 n_i 的前驱任务集合 pred $=\{n_j \mid e_{ji}\in E\}$；任务 n_i 的后继任务集合 succ $=\{n_i \mid e_{ij}\in E\}$，并规定 $|\mathrm{pre}(n_i)|$ 是任务 n_i 的前驱结点数量，$|\mathrm{succ}(n_i)|$ 是结点的后继任务数量。如果 $|\mathrm{pre}(n_i)|=0$，则定义该结点 n_i 是入口结点任务；如果 $|\mathrm{succ}(n_i)|=0$，则定义该结点为出口结点任务。$Rs=\{R_i \mid i\in N\}$ 是全部资源集合。我们约定 n_i/R_j 表示子任务 n_i 被分配到 R_j 上执行，$RT(R_i)=\{n_i \mid$ 所有在 R_i 上运行的子任务$\}$。

不失一般性，我们假定 DAG 图只有一个入口结点任务和一个出口结点任务。如果实际的 DAG 图中的入口结点数和出口结点数多于 1 个，则可以增加一个虚拟入口和一个虚拟出口。

2 算法描述

遗传算法(GA)是 Holland 于 1975 年受生物进化论的启发而提出的。并行性和全局解空间搜索是 GA 的两个最显著的特点。网格计算中的资源调度一般形式下是一个 NP 问题。由于遗传算法的并行性和全局解空间搜索的特点，因此非常适合求解 NP 问题。

2.1 染色体编码

染色体的编码形式有很多，本文直接对子任务占用的资源编号编码，采用任务-资源的编码方式，如图 1 所示，设染色体为 chromosome[][]。

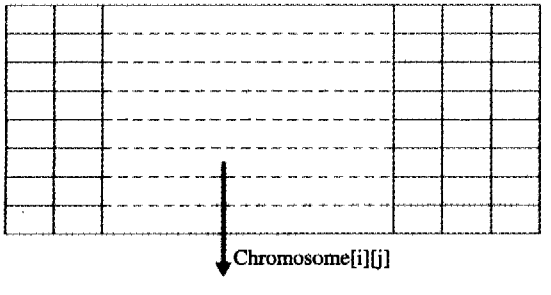

图 1 染色体编码

chromosome[i][j]二维数组中第一个坐标代表子任务编号，第二个坐标代表资源编号。如果 chromosome[i][j]=1，代表子任务 n_i 在资源 R_j 上执行；为 0，则代表该子任务没有在资源上执行。由于允许相同的子任务可以复制到不同的资源上执行，因此在同一行上面可能存在多个 1。

当产生了一个染色体后，还必须对其解码。我们按照资源分类就可以确定在不同资源上运行的子任务序列，生成子任务的执行次序序列并计算适应值。

2.2 染色体合法性

一个计算程序被分解为若干个可以并行的子任务，为了确保计算程序的完成，每个子任务都必须执行；由于子任务之间存在逻辑约束关系，在同一个资源上执行的子任务必须按照一定的执行顺序执行，否则将出现死锁现象[7]。因此我们约定：

①对于任意一个 $n_i\in V$，必 $\exists j$，使得 chromosome[i][j]=1，即每个子任务至少被分配到一个资源上运行；

②对于任意一个 $n_i, n_j\in V$，如果满足 $e_{i,j}\in E$，且 $n_i, n_j\in RT(R_k)$，则 n_i 必须先于 n_j 执行。

在产生初始种群的时候，为了使产生的初始种群满足约束①，我们得首先为每个子任务随机分配一个资源；然后产生每个任务的副本任务，分配到不同的资源上。这样，可以确保每个子任务至少在一个资源上执行。

对于生成的每一个染色体因子，将其解码后得到一组 $RT(R_i)$。对每个 $RT(R_i)$，采用根据每个子任务的高度值排序的方法来避免出现死锁现象。每个子任务的高度值计算方法是：

$$\mathrm{level}(\mathrm{sb}_i)=\begin{cases}0, & \mathrm{sb}_i \text{ 无父节点}\\ 1+\max(\mathrm{level}(\mathrm{parent}(\mathrm{sb}_i))), & \text{其它 } \mathrm{sb}_i\end{cases} \quad (1)$$

定理 1 对于同一资源上运行的子任务采用按高度值由低向高排序的方法，不会出现死锁现象。

证明：$\forall$ 资源 R_1, R_2，$\forall$ 子任务 n_1, n_2, n_3, n_4，不妨设 level $(n_1)<level(n_3)$，level$(n_2)<$level(n_4)。采用反证法证明。

假设子任务运行时出现死锁，且出现死锁的子任务在资源上的执行次序满足按照高度值从低到高排序，则它们在资源 R_1 和资源 R_2 上的分布情况必然是 n_1 和 n_4 在同一个资源上，不妨设为 R_1，且 n_4 先于 n_1 执行；同样，n_2 和 n_3 在同一个资源上，不妨设为 R_2，且 n_3 先于 n_2 执行。因为它们的执行次序满足高度值排序的要求，因此得到：level$(n_4)<$level(n_1)，level$(n_3)<$level(n_2)。又因为假设条件 level$(n_1)<$level(n_3)，所以得到 level$(n_4)<$level(n_2)。这样与已知条件矛盾，所以假设不成立。即对在同一资源上运行的子任务按照其高度值从低到高排序，可以避免死锁现象的出现，防止非法种子的产生。证毕。

2.3 适应值的计算

这里适应值主要考虑 3 个因素：①计算程序的完成时间；②计算任务占用的资源数量；③计算任务复制的总数。因为采用复制任务的目的是为了降低计算程序的执行时间，因此本文将第一个因素放在最先考虑。但是由于网格环境的特殊性，网格中资源的使用是需要代价的，不可能占用过多的资源；而且任务的复制是需要网络传输开销的，过多的复制任务产生会导致过多的网络开销，因此必须考虑第二和第三个因素。定义适应值函数为：$F=(f_1, f_2, f_3)$。其中 f_1 表示计算程序时间的适应值；f_2 表示计算任务占用资源的适应值；f_3 表示计算任务复制总数的适应值。我们规定：如果 $F_i=F_j$，则说明满足 $f_{in}=f_{jn}, 1\leqslant n\leqslant 3$。反之，如果 $F_i>F_j$，则说明 $\exists f_{in}>f_{jn}$ 且 $f_{im}>f_{jm}$，其中 $m<n$。本文主要考虑尽可能缩短计算时间，因此我们在计算适应值时，优先考虑 f_1。

2.3.1 收敛时间计算

根据子任务之间的逻辑关系和不同资源上子任务的运行序列，可以计算出每个资源完成该资源上所有子任务花费的

时间。取所有资源的最大花费时间的倒数为适应值的大小，因此花费时间越长，适应值越小。

计算适应值必须计算每个子任务的完成时间。假设子任务 n_i 在资源 R_j 上的完成时间为 $f(n_i/R_j)$，则

$$f(n_i/R_j)=s(n_i/R_j)+\tau_{ij} \tag{2}$$

其中 $s(n_i/R_j)$ 为子任务 n_i 在资源 R_j 上的开始执行时间，由3个因素决定：①资源的空闲时刻，即该资源上子任务 n_i 前一个任务的完成时间，定义为 spare(R_j/n_i)。②子任务 n_i 的所有父节点任务的完成时间。③父结点任务所在资源与子任务 n_i 所在资源之间的通讯延迟。其计算公式为：

$$s(n_i/R_j)=\max(\text{spare}(R_j/n_i),\ \max_{n_k\in pred(n_i)}(\min_{l\in L}(f(n_k/R_l),\ \min_{l=j}f(n_k/R_l)+C_{i,k}))) \tag{3}$$

$$\min_{l\neq j}$$

其中 $L=\{l|\text{chromosome}[n_k][l]=1\}$，spare$(R_j/n_i)$是资源 R_j 上最近的一次空闲时刻。

根据公式(2)可以计算出子任务在不同资源上开始执行的最早时间，然后根据公式(2)计算出每个子任务的完成时间。记资源 R_i 上最后一个任务的完成时间为 $RF(R_i)$，则所有子任务完成的计算时间为 $f=\max_{R_i\in R}(RF(R_i))$。定义适应值为 $f_1=1/f$，则适应值越大，完成时间越短。

2.3.2 计算资源的确定

当计算求得的 $f_{i1}=f_{j1}$ 时，我们需要比较 f_{i2} 和 f_{j2}。遍历每个资源，如果有任务在该资源上执行，则使用的资源数加 rnum1。取 $f_2=1/\text{rnum}$，即适应值为资源数量的倒数。

2.3.3 计算任务数量的确定

当前两个适应值相等时，需要计算 f_{i3} 和 f_{j3}。遍历每个任务，如果它被复制到某个资源上执行，则复制任务数量加 tnum1。取 $f_3=1/\text{tnum}$，即适应值等于复制任务数量的倒数。

下面我们将讨论交叉和变异因子。

2.4 交叉算子

在交叉操作前，要选择适当的个体进行交叉操作。选择个体的方法很多，例如轮盘赌选择方法[7]、随机遍历抽样选择[7]等等。这里交叉算子采用 Syswerda(1989)提出来的 OX 交叉算子[8]，交叉点的位置随机确定。这里的交叉操作采用的是选择两个资源，交换它们上面执行的计算子任务，如图2所示。经过交叉生成的两个子个体，我们分别对它们计算适应值。如果生成的子个体的适应值大于父个体，则按照一定的交叉概率替换父染色体。反之，则选择原种群中适应值最小的染色体比较。如果新子个体的适应值大于最低适应值，则替换原染色体；否则舍弃生成的子染色体，重新选择父染色体，进行交叉操作。

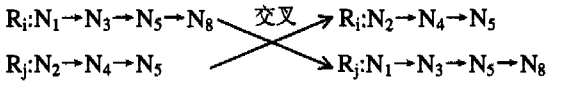

图2 交叉算子

2.5 变异操作

变异算子在一定程度上克服了算法的早熟收敛，有利于增加种群的多样性。将变异算子分成3个操作：删除算子、迁移算子和复制算子。删除算子操作就是将选中的在某个资源上执行的子任务删除；迁移算子就是将选中的子任务从原来的资源上迁移到其它资源上执行；复制算子就是将选中的子任务增加一个副本，在不同的资源上运行。我们规定，在执行复制和迁移算子的时候，优先考虑将子任务复制(或移动)到它的父任务所在的资源上。如果所有父任务的资源上都已经有该子任务，则再随机产生一个新的资源，将该子任务复制到新的资源上执行。

3 过分冗余任务的删除

由于我们采用的是资源-任务的间接编码方式，交叉和变异算子生成新染色体的过程实际上是对子任务占用的资源的重新分配。在交叉和变异操作后，对所有子任务重新按其占用的资源分类，并对在同一资源上运行的子任务，按照深度值的大小排序，这样就不会存在死锁现象。因此在交叉和变异操作后生成的新个体依然符合子任务之间的逻辑关系。

3.1 过分冗余结点定义

经过交叉、变异产生的染色体对应着一组任务复制到不同资源的分布关系，根据染色体编码，我们可以得到任务在资源上的分布图。但是，其中有些任务对于整个计算程序的执行时间并没有任何影响，只是在交叉、变异过程中产生附加冗余任务。我们可以删除这些无用的复制任务，减少任务的复制数量。下面首先给出过分冗余任务的定义，然后给出删减算法。

定义1 过分冗余任务：对于在资源 R_j 上运行的结点任务 n_i，如果取消该任务在资源 R_j 上运行后得到的收敛时间不大于取消前的收敛时间，这时我们定义该结点为过分冗余任务。

3.2 删除算法

过分冗余任务的删减算法如下：

输入：删减前的染色体 chromosome[][]，该染色体的适应值 oldfitness；

输出：删减后的染色体 chromosome[][]。

算法Ⅰ步骤：

```
1) for(i=0;i<nodenumber;i++);{
2)  for(j=0;j<resourcenumber;j++){
3)   if(chromosome[i][j]=1){
4)    flag=JudgeCopy(); //判断该子任务是否还有其它副本
                          运行在别的资源上。True 表示
                          有,false 表示无。
5)    if(flag){
6)     chromosome[i][j]=0
7)     fitness=computefitness(); //计算删减后的适应值;
8)     if(oldfitness>=fitness)
9)      oldfitness=fitness;
10)     else
11)      chromosome[i][j]=1//恢复原来的值 }}}}
```

定理2 根据删减算法得到的优化后的染色体，根据其编码得到的任务-资源分布关系计算出来的收敛时间不会大于优化前计算出的收敛时间，而且根据该删减算法不会打破约束1和约束2。

证明：由算法Ⅰ可以得出删减某个任务的复制版本后，要比较新得到的染色体和旧染色体的适应值大小，选择适应值大的染色体保存。如果原有染色体的适应值大，则重新恢复该复制任务(第11行)；反之，保存新生成染色体的适应值(第9行)。因此，染色体的适应值不会小于原有适应值，即收敛时间不会大于优化前的适应值。而且由于在删减前首先判断是否该任务还有其它副本存在，当且仅当存在其它复制任务在其它资源上执行时，才执行删除算法，避免了所有子任务副本都被删除的可能，因此满足约束1和约束2。

4 MGaTDS 算法步骤

Ⅰ. 产生初始种群 chromosome[popid][][]，并计算初始状态下的种群中每个染色体的适应值；

Ⅱ. 采用蒙特卡罗算法选择两个染色体基因，进行交叉算子操作；

Ⅲ. 随机选择一个染色体，进行变异操作；

Ⅳ. 当迭代次数小于规定次数时，继续步骤Ⅰ；

Ⅴ. 优化种群中所有染色体，消除过分冗余任务；

Ⅵ. 找出种群中最大适应值对应的染色体，并输出其收敛时间和任务复制分布。

5 实验结果及分析

我们针对图 3 中所示的 DAG 图进行了实验，圆圈中的 $n_i(i=1,2,\cdots,11)$ 表示任务结点编号，箭头线段上旁的数字表示任务之间的通信延迟。表 1 是不同子任务在不同资源上的执行开销。我们对 TDS 算法、普通的基于遗传算法的任务复制算法(CGaTDS)和本文的算法(MGaTDS)进行比较，从最小计算完成时间、占用资源数量、复制任务数目 3 个方面做比较。这里种群大小为 200，交叉概率取 0.8，变异概率取 0.05，最大迭代数目 10000。3 种算法比较结果如表 2 所示，3 种算法计算产生的时间轴如图 4、图 5、图 6 所示。由图中可以看出，虽然 3 种结果产生的计算收敛时间相同，但是复制的计算任务不同。采用 MGaTDS 的复制任务数量最少。

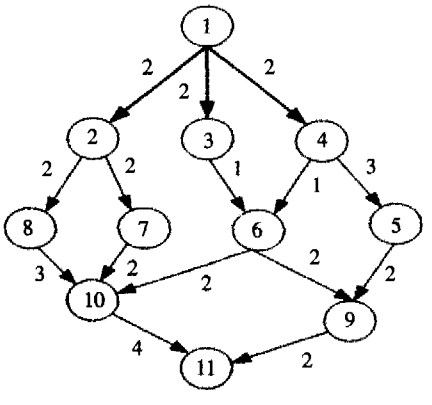

图 3 子任务 DAG 图

表 1 不同子任务在不同资源上的执行开销

处理资源→子任务编号↓	资源 1	资源 2	资源 3	资源 4
1	4	4	4	4
2	5	5	5	5
3	4	6	4	7
4	3	3	3	3
5	3	5	3	4
6	3	7	2	2
7	5	8	5	5
8	2	4	5	3
9	5	6	7	5
10	3	7	5	2
11	5	6	7	8

表 2 三种算法比较

	计算时间	资源占用数	复制任务数量
TDS 算法	25	4	15
CGaTDS 算法	25	4	16
MGaTDS 算法	25	4	12

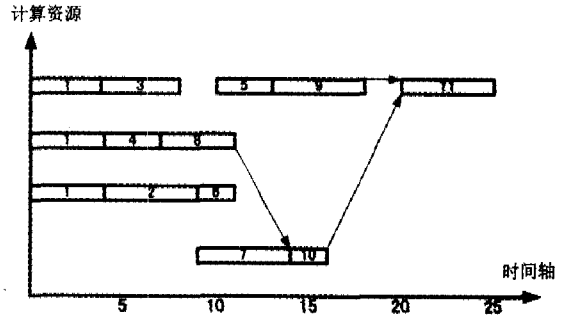

图 4 MGaTDS 算法时间轴

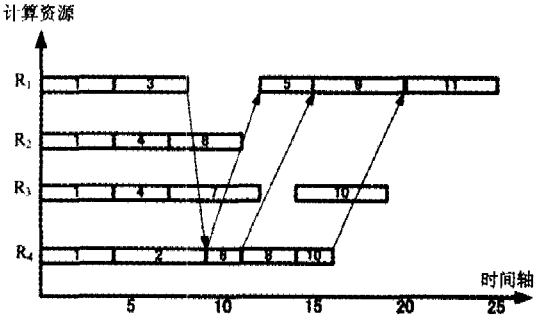

图 5 CGaTDS 算法时间轴

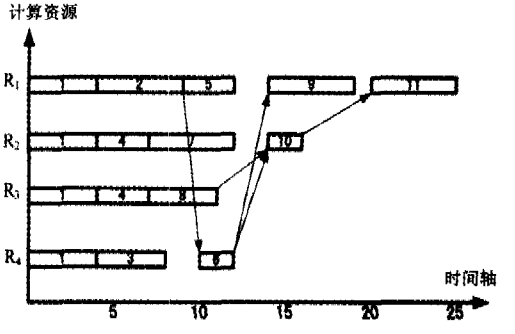

图 6 TDS 时间轴

我们还随机产生了结点数为 20、30、40 的树型结构的 DAG 图，其子任务的运行开销随机产生。我们分别采用 MGaTDS 算法和 CGaTDS 算法进行了调度，结果如表 3 所示。

表 3 两种算法比较

	收敛时间		占用资源数		复制任务数	
	CGaTDS	MGaTDS	CGaTDS	MGaTDS	CGaTDS	MGaTDS
结点 20、资源 8	66	66	8	6	38	26
结点 30、资源 10	151	155	10	9	67	32
结点 40、资源 12	229	225	12	10	94	51

从表 2 可以看出，两种算法收敛时间结果相差不大，MGaTDS 算法小于或等于 CGaTDS 算法。但是采用 MGaTDS 算法计算出来的复制任务更少、占用的资源更少，显然更符合网格中缩短计算程序时间和降低计算开销的目的。

结束语 本文在分析了基于任务复制的几个典型算法后，提出了以使调度长度最小作为主要目标、减少处理机数目作为次要目标的 MGaTDS 算法，缩短了整个任务的执行时间，减少了网格资源的占用数量和复制任务的数量。通过与

(下转第 105 页)

该中心用于在紧急事件中发布命令。

(7)意识培养和培训项目:建立对机构人员进行意识培养和技能培训的项目,以便灾难恢复计划能够得到制定、实施、维护和执行。

(8)维护和测试灾难恢复计划:对计划进行演练测试,并评估和记录演练的结果。制定维持连续性能力和 DRP 文档更新状态的方法,使其与企业的策略方向保持一致。通过与适当标准的比较来验证 DRP 的效率,并使用简明的语言报告验证的结果。

3 灾难恢复体系结构的三维模型

灾难恢复利用先进的软、硬件设备和环境,以灾难恢复等级(量化指标)为中心,综合运用各种技术手段(灾难备份与恢复技术)和管理措施(灾难恢复计划与措施),实现信息系统的灾难恢复要求。为了更好地表示信息系统的灾难恢复的层次结构、备份与恢复技术、灾难恢复计划与措施以及灾难恢复指标之间的关系,本文用图 3 所示的三维模型来进一步阐述体系结构中各元素之间的关系。

一个特定信息系统的灾难恢复体系结构(Disaster Recovery Architecture)可定义为一个三元组:$DRA=(O,T,P)$,其中:

$O=\{\langle O_1,O_2,O_3,O_4\rangle \mid O_1,O_2,O_3,O_4$ 分别表示不同类型指标 RPO,RTO,NRO,DOO 的值约束$\}$,即 O 表示该信息系统所要达到的各种灾难恢复指标;

$T=\{T_i\}$ 表示为保证达到该系统的各种灾难恢复指标,必须提供的多种备份与恢复技术 T_i;

$P=\{P_j\}$ 表示为保证达到该系统的各种灾难恢复指标,必须实施的各种灾难恢复计划和措施 P_j。

在灾难恢复体系结构的三维模型中,可根据三元组定义若干的规则点。由若干规则点构成的集合,则形成特定信息系统的灾难恢复解决方案,为整个体系结构的各个层次、各个组件协调工作起到核心控制作用。

结论 信息系统的灾难恢复体系结构以灾难恢复等级为中心,对备份与恢复技术及灾难恢复计划与措施统一进行管理,使体系结构中的各个层次、各个组件有机、协调地统一运作,为在发生灾难性事故的时候,以灾难恢复指标为标准对原系统进行恢复,以保证数据的完整性、可靠性和安全性以及业务的连续可用性。本文详细分析了灾难恢复系统的各个组成部分,根据容灾层次重新制定了灾难恢复等级,提出了灾难恢复系统的体系结构及相应的三维模型,从而为信息系统提供一个整体的灾难恢复解决方案。

参 考 文 献

1 Keeton K, Santos C, Beyer D, et al. Designing for disasters. In: Proceedings of the 3th USENIX Conference on File and Storage Technologies, San Francisco, CA, USA, 2004. 59~72

2 Lewis W, Richard Jr, Watson T, et al. An Empirical Assessment of IT Disaster Risk. Communications of the ACM, 2003, 46(9): 201~206

3 Zussman G, Segall A. Energy efficient routing in ad hoc disaster recovery networks. Proc of INFOCOM, 2003

4 IBM公司. IBM 容灾白皮书. http://www-900.ibm.com/cn/support/download/Disaster_recovery.pdf

5 Kim S-K, Dshalalow J H. Stochastic disaster recovery systems with external resources. Mathematical and Computer Modelling, 2002, 36(11): 1235~1257

6 Choy M H, Leong H V, Wong M H. Disaster Recovery Techniques for Database Systems. Communications of the ACM, 2002, 43(11)

7 *Toigo J W. Disaster Recovery Planning: Strategies for Protecting Critical Information Assets.* Prentice Hall PTR, 3rd edition, 2002

8 Veritas软件公司. 企业重生——信息系统的灾难恢复. 北京:机械工业出版社, 2004

9 牛云,等. 数据备份与灾难恢复. 北京:机械工业出版社, 2004

10 Azagury A, Factor M E, Satran J. Point-in-time copy: Yesterday, today and tomorrow. In: Proceedings of the Tenth Goddard Conference on Mass Storage Systems and Technologies. 2002. 259~270

11 Patterson H, Manley S, et al. SnapMirror: file-system based asynchronous mirroring for disaster recovery. In: Proc. File and Storage Technologies(FAST), 2002. 117~129

12 Ji M, Veitch A, Wilkes J. Seneca: remote mirroring done write. In: Proceedings of USENIX Technical Conference, 2003. 253~268

13 Wiesmann M, Pedone F, Schiper A, et al. Understanding replication in databases and distributed systems. In: Proceedings of 20 th International Conference on Distributed Computing Systems(ICDCS'2000), 2000. 264~274

14 Ahmad I. Cluster Computing: A Glance at Recent Events. IEEE Concurrency, 2000, 8(1): 67~69

(上接第 92 页)

相关工作的比较可以看出,本算法在调度长度与处理器使用数目上均优于 CGaTDS 算法或与其相当。

参 考 文 献

1 Di Martino V. Scheduling in a grid computing environment using genetic algorithms. In: Mililotti M, ed. 16th International Parallel and Distributed Processing Symposium (IPDPS2002). Florida, USA, April, 2002

2 Di Martino V, Mililotti M. Sub optimal scheduling in a grid using genetic algorithms. Parallel Computing, 2004, 30: 553~565

3 Abraham A, Buyya R. Nature's Heuristics for Scheduling Jobs on Computational Grids. In: The 8th International Conference on Advanced Computing and Communications(ADCOM 2000). Cochin, India, 2000

4 XU Zhihong, HOU Xiangdan, SUN Jizhou. An Algorithm-Based Task Scheduling in Grid Computing. CCECE 2003 - Canadian Conference on Electrical and Computer Engineering. Montreal, Canada, 2003

5 Park Chan-Ik, Choe Tae-Young. An optimal scheduling algorithm based on task duplication. In: The 8th International Conference on Parallel and Distributed Systems(ICPADS2001), Kyongju City, Korea, June 2001. 9~14

6 Ranaweera S, Agrawal D P. A task duplication based scheduling algorithm for heterogeneous systems. In: The 14th International Parallel and Distributed Processing Symposium (IPDPS2000), Cancun, Mexico, May 2000. 445~450

7 Tsuchiya T, Kikuno T, Osada T. A new heuristic algorithm based on GAs for multiprocessor scheduling with task duplication. In: 3rd International Conference on, Dec. 1997. 295~308

8 王小平,曹立明. 遗传算法. 西安:西安交通大学出版社, 2002