文章编号:1001 - 9081(2010)08 - 2066 - 04

可靠的网格作业调度机制

陶永才,石 磊
(郑州大学 信息工程学院,郑州 450001)
(ieyctao@ zzu. edu. cn)

摘 要:针对网格环境的动态性特征,提出了一种可靠的网格作业调度机制(DGJS)。按照作业完成时间期限,DGJS 将作业分为:高 QoS 级、低 QoS 级和无 QoS 级,不同 QoS 级作业有不同的调度优先权;基于资源可用性预测,DGJS 采用基于可靠性代价的作业调度策略,将作业尽可能调度到可靠性高的资源节点;另外,DGJS 对不同 QoS 级作业采用不同的容错策略,在保证故障容错的同时,节省网格资源。实验表明:在动态的网格环境下,较之传统的网格作业调度算法,DGJS 提高了作业成功率,减少了作业完成时间。

关键词:作业调度;网格;资源故障;容错;马尔可夫链
中图分类号: TP393;TP302.1;TP301.6 **文献标志码**:A

Dependable grid job scheduling mechanism

TAO Yong-cai, SHI Lei
(*College of Information Engineering, Zhengzhou University, Zhengzhou Henan* 450001, *China*)

Abstract: With regard to the dynamic feature of grid, a Dependable Grid Job Scheduling (DGJS) mechanism was proposed in this paper. According to the deadline of job finish time, DGJS classified the submitted jobs into three levels with different priority: high QoS level, low QoS level and no QoS level. Based on the resource availability prediction, DGJS exploited reliability cost-based job scheduling strategy, striving to schedule jobs to the resource nodes with high reliability. In addition, DGJS exploited different fault-tolerant strategies for jobs with different QoS levels. The experimental results show that in the dynamic grid environments, DGJS increases the job success ratio and reduces the job finish time.

Key words: job scheduling; grid; resource failure; fault tolerance; Markov chain

0 引言

网格整合了 Internet 上分布的、异构的各种资源,具有高度的动态性[1]。一方面,网格资源大都属于非专用资源,资源随意加入和退出,以及其软硬件故障、网络中断都可能导致资源不可访问;另一方面,网格用户竞争使用网格资源,资源性能处于动态变化,包括 CPU、内存、网络带宽等。网格环境的动态性特征给网格作业调度带来许多新的挑战,资源故障导致作业失败频繁发生,用户服务质量无法保证。

为弥补网格动态性对作业运行的影响,现有的网格系统多采用单故障容错机制,例如:Legion 和 DataGrid 采用检查点容错,Condor-G、Netsolve 采用重试容错,Mentat 采用主副本容错[2-3]。检查点容错需要额外的存储空间保存作业中间运行结果和数据;重试容错在作业运行失败的情况下对作业进行再次调度,包括再调度到本资源节点或其他资源节点;主副本容错在不同的资源节点运行作业的主版本和副版本,副版本又可以采用被动运行和主动运行方式。故障容错机制在一定程度上保证了作业的可靠运行,但该机制需要消耗大量网格资源,并且不能预防作业失败的发生。因此,为了避免作业失败,在作业调度阶段准确地预测网格资源节点的可靠性异常重要。

网格作业调度吸引了大量的研究工作。调度策略可以分为性能驱动、经济驱动和可靠驱动[2,4]。性能驱动策略旨在优化系统性能,包括最小化作业完成时间和最大化系统吞吐量。经济驱动策略在满足用户服务质量前提下,选择收费最低的资源节点进行作业调度。上述调度策略都未考虑资源节点的可靠性,因此不能保证网格作业的可靠运行。近年来,一些研究工作开始将资源节点的可靠性作为作业调度的重要因素[5],该机制在满足用户作业 QoS 请求的前提下,把作业分配给可靠性高的资源节点执行。文献[5]提出"可靠性代价(Reliability Cost, RC)"作为调度目标来提高作业运行的可靠性。可靠性代价(RC)定义如式(1),f_j 为资源节点 j 的故障率,CT_{ij} 为作业 i 在资源节点 j 上的完成时间。

$$RC_{ij} = f_j \cdot CT_{ij} \tag{1}$$

文献[5]提出的 Refine 算法将作业尽量调度到可靠性代价(RC)最低的资源节点。另外,Refine 算法采用主副本容错机制。研究表明资源节点在不同时间段的故障率是不同的[6],即不同作业在资源节点的运行期间,资源的故障率是不同的。上述研究没有涉及如何确定资源的可靠性,对于资源节点的故障率采用经验值。

网格环境下,用户作业 QoS 具有不同的需求。例如:实时作业对完成时间期限要求严格(如天气预告分析、地震预告分析),而普通作业只关心服务收费。不同 QoS 要求的作业竞争使用具有不同 QoS 的网格资源。传统的网格作业调度常常导致:无 QoS 要求的作业占用高 QoS 的网格资源,高 QoS 要求的作业处于等待状态,低 QoS 的网格资源得不到很好的

收稿日期:2010 - 02 - 23;**修回日期**:2010 - 03 - 17。 **基金项目**:国家 863 计划项目(2006AA01A115)。

作者简介:陶永才(1975 -),男,河南武陟人,博士,主要研究方向:网格计算、高性能计算; 石磊(1967 -),男,河南郑州人,教授,博士,主要研究方向:高性能计算、Web 数据挖掘。

利用。同时,高 QoS 要求的作业可能被调度到低 QoS 的资源,而导致作业失败率高。

针对网格环境的动态性特征,本文提出一种可靠的网格作业调度机制(Dependable Grid Job Scheduling, DGJS)。按照作业完成时间期限的要求,DGJS 将作业分为:高 QoS 级、低 QoS 级和无 QoS 级,不同 QoS 级作业有不同的调度优先权;基于资源可用性预测,DGJS 采用基于可靠性代价的作业调度策略,将作业尽可能调度到可靠性高的资源节点;另外,DGJS 对不同 QoS 级作业采用不同的容错策略:对高 QoS 级作业,采用复制容错策略;对低 QoS 级作业,采用主副本容错策略;对无 QoS 级作业,采用重试容错策略。DGJS 在保证故障容错的同时,节省网格资源。实验表明:在动态的网格环境下,较之传统的网格作业调度,DGJS 提高了作业成功率,减少了作业完成时间。

1　基于 Markov 链的资源可用性预测模型

本章描述如何建立基于 Markov 链的资源可用性预测模型,为网格作业的可靠调度提供信息基础。

目前,Markov 模型已被广泛应用于多个研究和应用领域,通过扩展 Markov 模型对随机过程进行建模。在离散时间 Markov 链中,$\boldsymbol{M}(S_1,S_2,\cdots,S_M)$ 表示一组有限的状态空间,$\boldsymbol{Q}=\boldsymbol{M}$,其中 $\boldsymbol{M}$ 表示状态转移矩阵。在状态转移矩阵 $\boldsymbol{Q}$ 中,$\boldsymbol{Q}_{ij}$ 表示从状态 S_i 到状态 S_j 下的变换概率。Markov 特性的过程状态无记忆性使得 Markov 模型可以用于预测过程未来的状态变化。假设在 t_0 时刻,系统所处的状态为 $S_i(1\leqslant i\leqslant M)$,并且在 t_0 时刻状态 S_i 的分布 $\boldsymbol{Q}_0(S_i)=\boldsymbol{e}_i$,其中 $\boldsymbol{e}_i$ 为 $1\times M$ 的行向量,在位置 i 其值为 1,在其他位置其值为 0。因此可以作出预测,状态 S_i 在 t_1 时刻的分布为:

$$\boldsymbol{Q}_1(S_i)=\boldsymbol{Q}_0(S_i)\boldsymbol{Q}=\boldsymbol{e}_i\boldsymbol{Q} \tag{2}$$

在 t_2 时刻的分布为:

$$\boldsymbol{Q}_2(S_i)=\boldsymbol{Q}_1(S_i)\boldsymbol{Q}=\boldsymbol{e}_i\boldsymbol{Q}^2 \tag{3}$$

依次类推,在 t_k 时刻状态 S_i 的分布为:

$$\boldsymbol{Q}_k(S_i)=\boldsymbol{Q}_{k-1}(S_i)\boldsymbol{Q}=\boldsymbol{e}_i\boldsymbol{Q}^k \tag{4}$$

基于 Markov 模型,在每一时刻都会得到状态 S_i 在此时刻以及下一时刻的分布,根据此状态分布,可以预测在下一时刻每个状态发生的概率。

网格资源节点可用性具有很大的随机性,Markov 模型可用于对其进行模拟预测。无故障运行时间(Time To Failure, TTF)通常用来表示资源节点的可用性,本文采用 TTF 值构成 Markov 预测模型的系统状态 $\boldsymbol{M}$。

在上述的 Markov 预测模型中,状态空间 $\boldsymbol{M}$ 和状态转换矩阵 $\boldsymbol{Q}$ 是常量,网格环境的动态性使得 $\boldsymbol{M}$ 和 $\boldsymbol{Q}$ 变得异常大,导致预测模型复杂不适用。针对此问题,本文提出基于 Markov 链的资源可用性预测模型,此模型针对网格特征动态地修改状态空间 $\boldsymbol{M}$ 和状态转换矩阵 $\boldsymbol{Q}$,具体原理如下:

当网格资源节点发生故障时,新的 TTF 值产生(TTF_{new}),系统遍历状态空间 $\boldsymbol{M}$,如果存在状态 S_i,它与 TTF_{new} 的绝对差值小于系统设定值,合并 S_i 与 TTF_{new} 取其平均值为新状态 S_i,状态转移矩阵中对应的变换概率加 1。相反,如果遍历状态空间 $\boldsymbol{M}$ 不存在满足上述条件的状态,将创建新的系统状态 S_{m+1},同时状态转移矩阵 $\boldsymbol{Q}$ 根据式(5)进行修正。

$$\boldsymbol{Q}_{ij}=n_{ij}/N_{\text{total}},N_{\text{total}}=\sum_K n_{ik} \tag{5}$$

其中:n_{ij} 表示系统从状态 i 到状态 j 的转移次数,$\sum_K n_{ik}$ 表示在 K 次故障中所有的状态转移次数。

2　可靠性代价

资源节点的可靠性为节点在作业运行期间不发生故障的概率。考虑一个具有 $\boldsymbol{M}$ 个资源节点的网格系统 $\boldsymbol{P}=\{P_1,P_2,\cdots,P_M\}$。设定 r_{ij} 表示作业 J_i 在资源节点 P_j 运行期间 P_j 无故障的概率。研究表明系统可靠性服从负指数分布[7],作业 J_i 在资源节点 P_j 运行成功的可靠性(Pr)可由式(6)计算获得。

$$Pr=\xi\cdot e^{-(1-r_{ij})\cdot CT_{ij}} \tag{6}$$

$$CT_{ij}=RT_{ij}+D_j(i)+SL_j \tag{7}$$

其中:ξ 是修复参数,CT_{ij} 为作业 J_i 在资源节点 P_j 上的完成时间,RT_{ij} 为作业 J_i 在资源节点 P_j 上的运行时间,$D_j(i)$ 为 J_i 在运行前需要取得所需数据花费的传输时间,SL_j 为资源节点 P_j 的作业调度长度。从式(6)分析可得,为了最大化 Pr,需要最小化 $(1-r_{ij})CT_{ij}$。根据式(1)的可靠性代价定义,网格作业的可靠性代价为:

$$RC_{ij}=(1-r_{ij})\cdot CT_{ij} \tag{8}$$

这样,为了最大化 Pr,需要最小化 RC_{ij}。

根据式(6)、(8)可知,要提高作业运行的可靠性,需要降低作业运行的可靠性代价。可靠性代价越低,作业运行的可靠性越高。

3　DGJS 调度机制

DGJS 调度机制可分为预调度和可靠调度两个阶段。

3.1　预调度

在预调度阶段,系统接收到用户提交的作业,按照作业完成时间期限,将作业分为 3 级:高 QoS 级、低 QoS 级和无 QoS 级,分别置于高、中、低 3 个调度优先级队列。系统空闲时,按照 QoS 级依次从高、中、低 3 个调度队列取出作业,进行调度。DGJS 预调度通过设置调度优先级满足完成时间期限要求严格的作业,同时也避免无 QoS 队列中的作业发生饥饿而一直得不到调度。DGJS 预调度作业 QoS 分级策略如下:

1)无 QoS 级。没有指定完成时间期限的作业,属于无 QoS 级作业。

2)低 QoS 级。指定有完成时间期限,并且满足式(9)的作业,属于低 QoS 级作业。

$$J_i.t_a+(1+\lambda)J_i.CT_{\min}\geqslant J_i.t_d;\qquad \lambda<1 \tag{9}$$

其中:J_i 表示第 i 个作业,t_a 为作业到达时间,t_d 为作业完成时间期限,CT_{min} 为作业最小完成时间,λ 为 QoS 级调整参数,通常 λ 值大于 1。

3)高 QoS 级,指定有完成时间期限,并且满足式(10)的作业,属于高 QoS 级作业。

$$J_i.t_a+(1+\lambda)J_i.CT_{\min}<J_i.t_d;\qquad \lambda<1 \tag{10}$$

3.2　可靠调度

在可靠调度阶段,系统空闲时,按照 QoS 级依次从高、中、低三个调度队列中取出作业,基于可靠性代价进行调度。同时,针对不同 QoS 级作业,DGJS 采用不同的容错策略,在最小

的资源消耗代价下保证作业的可靠运行。

3.2.1 基于可靠性代价的作业调度

假定资源 $P_j(1 \leqslant j \leqslant m)$ 已经分配了 n 个作业 $J_1, J_2, \cdots, J_n$，资源 P_j 的空闲时间为 $[0, st_j(J_1)], [ft_j(J_1), st_j(J_2)], \cdots, [ft_j(J_n), \infty]$。$st_j(J_i)$ 和 $ft_j(J_i)$ 表示作业 J_i 在资源 P_j 上的启动时间和完成时间。$est_j(J_i)$ 和 $lst_j(J_i)$ 为作业 J_i 在资源 P_j 上的最早启动时间和最迟启动时间。DGJS 对网格资源节点未来可用性进行预测，并采用基于可靠性代价的作业调度，见算法1所示。

算法1 基于可靠性代价的作业调度。

输入：$J[]$：作业集合；$P[]$：可用资源集合；

输出：$S[]$：调度结果；

while (∃ not scheduled job i in $J[]$) do

for $j = 1$ to m do

寻找 $est_j(J_i)$，P_j 空闲时间满足：

$(\max(ft_j(J_x), J_i.t_a) + CT_{ij}) < st_j(J_{x+1})$；

if $(J_i.t_a + CT_{ij}) < J_i.t_d$ then

$r_{ij} = \sum_{k=x}^{M} S_k Q_{pk} / \sum_{q=1}^{M} S_q Q_{pq}$；

// $S_x > CT_{ij} + (T_{\text{now}} - ST_j)$

$RC_{ij} = (1 - r_{ij}) CT_{ij}$；

$J_i.MinRC[i] = \text{Min}(J_i.MinRC[i], RC_{ij})$；

endif

endfor

if $J_i.MinRC[i] != \varnothing$ then

$S[] \Leftarrow \{P_j \leftarrow J_i\}$；

end if

endwhile

基于资源可用性预测模型，DGJS 对资源节点 P_j 在作业 J_i 运行期间的可靠性进行预测，即资源 P_j 在时间 CT_{ij} 内正常运行的概率，也是作业 J_i 运行成功的概率。假定资源节点在 t_q 时刻所处状态为 S_q，在下一时刻 t_{q+1} 的状态分布为 $(Q_{q1}\ Q_{q2} \cdots Q_{qM})$，$S_k(1 \leqslant k \leqslant M)$ 代表资源节点的无故障运行时间(TTF)，要保证作业 J_i 在资源 P_j 的可靠性运行，必须满足条件：

$$S_k - T_{\text{now}} > CT_{ij} + ST_j \tag{11}$$

其中：T_{now} 表示当前时间，ST_j 表示资源 P_j 的启动时间。可计算作业 J_i 在资源 P_j 的运行可靠性为：

$$r_{ij} = \sum_{k=x}^{M} (S_k Q_{qk}) / \sum_{h=1}^{M} (S_h Q_{qh}) \tag{12}$$

根据式(12)，可计算作业 J_i 在资源 P_j 上成功运行的可靠性代价为：

$$RC_{ij} = (1 - r_{ij}) \cdot CT_{ij} \tag{13}$$

最后，DGJS 调度作业到具有最小 RC_{ij} 值的资源节点。

3.2.2 基于作业 QoS 级的容错机制

DGJS 针对不同 QoS 级作业，采用灵活的容错策略。对高 QoS 级作业，采用复制容错策略；对低 QoS 级作业，采用主副本容错策略；对无 QoS 级作业，采用重试容错策略。

1)高 QoS 级作业。

高 QoS 级作业主要指网格中的实时作业，网格应用中的实时作业都有严格的完成时间期限，如果没有在截至时限内完成，可能带来灾难性的后果。因此，实时作业的执行应具有高可靠性。对高 QoS 级作业，DGJS 采用复制容错策略。复制容错也称主副本并行运行容错，其机制为：将主本作业和副本作业同时调度到可靠性代价最低的两个资源节点，并同时启动运行。最后，不论是主本作业还是副本作业运行成功，作业即被标记为成功。复制容错策略通过将作业调度到两个不同资源节点并行运行来保证作业运行的可靠性，作业成功率高，但消耗资源大。

2)低 QoS 级作业。

相比高 QoS 级作业，低 QoS 级作业完成时间期限比较宽松。为在最小的资源消耗代价下保证作业的可靠运行，提高资源利用率，DGJS 采用主副本容错策略。主副本容错不同于复制容错，其特点如下：

基于可靠性代价进行主副本作业调度，作业被调度在不同的资源节点。

主本作业应尽可能早地调度运行，副本作业则尽可能晚地调度运行。

副本作业采用被动模式和主动模式两种调度方式。如果副本作业的最迟启动时间与主本作业的预期完成时间重叠，副本作业以主动模式启动，即在主本运行期间，副本作业即启动；否则，副本作业以被动模式启动。如式(14)所示，PJ 为主本作业，BJ 为副本作业。

$$(PJ.t_a + CT_{ij}) \geqslant lst(BJ_i) \tag{14}$$

如果主本作业运行成功，则取消副本作业运行。副本作业的被动模式容错与重试容错的区别在于，副本作业的主动和被动模式容错采用资源预留的方式进行调度，而重试容错则是在作业发生故障后再进行调度。

3)无 QoS 级作业

无 QoS 级作业没有指定完成时间期限。为了节省系统资源，保证高 QoS 级作业和低 QoS 级作业的可靠执行，对无 QoS 级作业采用重试容错，即在作业发生故障后，将作业基于可靠性代价再次调度到其他资源节点。

4 性能评价

实验采用10节点 PC 集群，各节点通过 100 Mbps 以太网连接，每一个节点配置是单 PIII 1 GHz 处理器，1 GB 内存，操作系统是 Red Hat Linux 2.4.20-8。实验采用 GridSim[8] 进行模拟实验，GridSim 是在 SimJava 的基础上开发的，它提供丰富的函数库以支持模拟网格环境中的异构资源、用户、应用程序、用户代理和调度器，可以快速地搭建模拟实验平台。实验采用2个性能指标：1) 平均成功率。表示成功运行的作业数与提交的作业总数之比值；2) 平均完成时间。表示从作业提交到得到作业结果的时间。实验设置式9)和10)中 λ 值($\lambda = 1.5$)，作业包括3种类型：完成时间期限严格作业、完成时间期限宽松作业、无完成时间期限要求作业。作业提交服从泊松分布(0.4)，模拟作业执行时间为 5 min ~ 20 min，每次提交作业从3种类型中随机选择。实验比较 DGJS、Min-min[9] 和 Refine[5] 算法的性能表现。

1) Min-min 算法。该算法计算每个作业在各个机器上的期望完成时间，获得每个作业的最小完成时间及其机器，再将具有最小完成时间的作业调度到相应的机器。

2) Refine 算法。该算法基于可靠性代价进行作业调度，并采用主副本容错。

网格环境下，资源故障特征具有时间局部性和空间局部性[6]，而不是平均分布。先前研究工作[10]对基于 Markov 链的资源可用性预测模型的有效性进行了验证。实验使用资源日志模拟器模拟资源故障[10]。该模拟器采用 Weibull 分布模拟故障的时间分布、Zipf 法则模拟故障的空间分布、Pareto 分

布模拟故障的修复时间。式(15)为两参数 Weibull 分布的分布函数,其中 α 是形状参数,η 是尺度参数。式(16)为基本 Zipf 定律,K 为常数,β 为 Zipf 参数。式(17)为两参数 Pareto 分布的分布函数。

$$F_W(t) = 1 - e^{-(t/\eta)^{\alpha}} \tag{15}$$

$$P(i) = K/i^{\beta} \tag{16}$$

$$F_P(x) = 1 - (\theta/x)^{\omega};\quad \lambda,\theta > 0, x \geqslant \theta \tag{17}$$

实验在资源低故障率和高故障率下分别进行性能测试。Weibull 分布参数设置如表 1 所示。

表 1　资源故障时间分布

尺度参数 η	形状参数 α	故障数	故障到达率/(min^{-1})
18000	0.083	106	0.016
	0.871	3135	2.177

Zipf 分布参数 $K = 46$、$\beta = 0.76$,β 值越大表示偏态分布越严重,即大部分的故障集中在少数的资源节点。Pareto 分布参数 $\theta = 127$, $\omega = 0.87$,ω 值越大表示故障修复时间重尾度越严重,即资源故障大部分属于暂时性故障,故障恢复时间较短。基于资源故障模拟日志,构建资源可用性预测器的状态空间 $\boldsymbol{M}$ 和状态转移矩阵 $\boldsymbol{Q}$。

图 1 ~ 2 所示为不同资源故障率下 DGJS、Min-min 和 Refine 算法的性能比较。由图 1 可知(故障模拟器参数为:η =18000, $\alpha = 0.873$, $K = 46$, $\beta = 0.76$, $\theta = 127$, $\omega = 0.87$) 在资源低故障率的情况下,Min-min 和 DGJS 表现比较好,Refine 表现最差。由图 2 所知(故障模拟器参数为:η =18000,$\alpha = 0.871$, $K = 46$, $\beta = 0.76$, $\theta = 127$,$\omega = 0.87$),在资源高故障率的情况下,Min-min 表现最差,DGJS 最好。分析原因如下:

1)Min-min 算法尽可能将作业调度到不仅完成时间最早,而且执行它最快的节点,而不考虑资源节点的可靠性。因此,在稳定的资源环境,Min-min 性能最好。在不稳定的资源环境,频繁的资源故障导致 Min-min 故障率高,再调度故障作业将增加作业的完成时间。

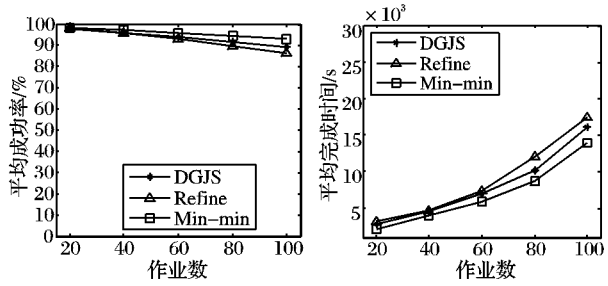

图 1　低故障率下性能比较

2)DGJS 和 Refine 在调度时,都考虑作业在资源节点上运行的可靠性代价。在稳定的资源环境,计算可靠性代价会带来额外的计算开销。另外,DGJS 和 Refine 采用复制容错和主副本容错会额外占用系统资源,影响作业运行,因此性能要比 Min-min 算法差。在不稳定的资源环境,DGJS 采用基于可靠性代价的调度机制,根据作业特点,将作业尽可能调度到可靠的资源节点,提高作业运行的可靠性。

3)Refine 算法对所有作业都采用主副本容错机制,在系统低负载时,Refine 表现很好。但是,当系统负载比较大时,由于主副本容错占用多一倍的资源,从而导致系统资源更加紧张,作业等待时间会变长,故障率相应会提高,导致作业成功率降低、完成时间增加。DGJS 针对不同 QoS 级作业灵活采用不同的容错机制,不仅保证作业的可靠运行,而且节省系统资源。

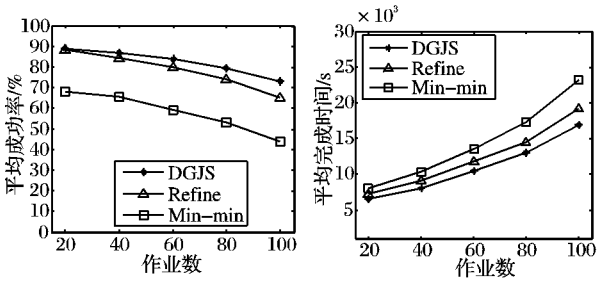

图 2　高故障率下性能比较

5　结语

本文提出一种可靠的网格作业调度机制 DGJS,根据作业的完成时间期限,将作业进行分级,不同 QoS 级作业有不同的调度优先权。DGJS 采用基于可靠性代价的作业调度策略,将作业尽可能调度到可靠性高的资源节点。DGJS 针对不同 QoS 级作业,采用灵活的容错策略:对高 QoS 级作业,采用复制容错策略;对低 QoS 级作业,采用主副本容错策略;对无 QoS 级作业,采用重试容错策略。实验比较了 DGJS、Min-min 和 Refine 算法的性能表现,实验结果表明 DGJS 在动态的网格环境下有较好的性能表现。

参考文献:

[1] FOSTER I, KESSELMAN C. The grid: Blueprint for a new computing infrastructure[M]. 2nded. San Francisco: Morgan Kaufmann Publishers, 2003.

[2] KRAUTER K, BUYYA R, MAHESWARAN M. A taxonomy and survey of grid resource management systems for distributed computing[J]. Software Practice and Experience, 2002, 32(2): 135 - 164.

[3] HWANG S, KESSELMAN C. Grid workflow: A flexible failure handling framework for the grid[C]//12th International Symposium on High Performance Distributed Computing. Washington, DC: IEEE, 2003: 126 - 137.

[4] 杜晓丽, 蒋昌俊, 徐国荣, 等. 一种基于模糊聚类的网格 DAG 任务图调度算法[J]. 软件学报, 2006, 17(11): 2277 - 2288.

[5] LUO W, QIN X, BELLAM K. Reliability-driven scheduling of periodic tasks in heterogeneous real-time systems[C]// 21st International Conference on Advanced Information Networking and Applications Workshops. Washington, DC: IEEE, 2007, 1: 778 - 783.

[6] FU S, XU C Z. Quantifying temporal and spatial correlation of failure events for proactive management[C]// 26th International Symposium on Reliable Distributed Systems. Washington, DC: IEEE, 2007: 175 - 184.

[7] CALABRIA R, GUIDA M, PULCINI G. A Bayes procedure for estimation of current system reliability[J]. IEEE Transactions on Reliability, 1992, 41(4): 616 - 620.

[8] CASANOVA H. Simgrid: A toolkit for the simulation of application scheduling[C]// 1st International Symposium on Cluster Computing and the Grid. Washington, DC: IEEE, 2001: 430 - 437.

[9] MANDAL A, KENNEDY K, KOELBEL C, *et al*. Scheduling strategies for mapping application workflows onto the grid[C]//14th International Symposium on High Performance Distributed Computing. Washington, DC: IEEE, 2005: 125 - 134.

[10] 陶永才. 网格环境下作业可靠调度机制的研究[D]. 武汉: 华中科技大学, 2009.