高技术通讯 2012 年 第22 卷 第5 期:477 ~482

doi:10.3772/j.issn.1002-0470.2012.05.005

可靠性代价和 Makespan 驱动的分布式容错调度算法①

景维鹏②*** 吴智博③* 刘宏伟* 董 剑*

(*哈尔滨工业大学计算机科学与技术学院 哈尔滨 150080)

(**东北林业大学信息与计算机工程学院 哈尔滨 150040)

摘 要 为解决异构分布式环境下采用主副版本策略的可靠性调度问题,提出一种基于优先级约束的可靠性代价和 Makespan(调度时长)驱动的分布式容错调度算法 DRCAMD。该算法可在满足系统可调度性的前提下,以异构分布式环境的节点、通信链路的可靠性与 Makespan 做为可调节局部目标函数,实现具有较高可靠性及较短执行时间的容错调度策略,避免将任务分配到失效率较高的节点上执行。另外,算法的副版本采用被动和主副重叠方式执行,使得容错调度算法具有较大的灵活性。仿真实验表明,该算法性能优于现有容错算法。

关键词 主副版本,优先级约束,高失效率,主副重叠方式

0 引 言

异构分布式实时系统是指由一组异构计算机通过网络连接而成的紧密配合完成一项共同任务的系统,在航空航天控制、地震数据处理、防御系统、天气预报等领域得到了广泛应用。这种系统执行的任务都有严格的时间约束,如果任务不能在其时限内完成,往往会导致灾难性后果。为了保证系统在出现故障后仍然能够在时限内完成任务,需要为分布式系统提供一定容错能力,以提高系统可靠性。提高分布式系统容错能力的一个有效方法是采用主副版本(primary/backup)调度技术。主副版本调度方法的目标是在满足一定性能指标和优先级约束的条件下,将可以并行执行的实时任务按照一定分配策略确定优先级,并映射到处理机上有序执行,从而实现较少执行时间和较高系统可靠性。主副版本技术在满足系统实时性的约束下通过在备份处理机上执行副本任务实现系统的容错性。副版本有三种方式:主动复制、被动(passive)复制、主副版本重叠(overlapping)。主动方式要求副版本与主版本同时运行,而被动方式和重叠方式则是主版本出现故障后,副版本才执行。因而采用主动方式的系统执行大量冗余任务,从提高处理机利用率角度看,副版本应尽量以被动方式或重叠方式执行。

近年来,针对异构分布式系统的周期任务,采用主副版本策略的容错调度算法相继被提出[1-8]。文献[1,2]提出了基于非抢占周期性实时任务的异构分布式容错调度算法,该算法不适合抢占式任务,因而在实际应用中有很大局限性。文献[3,4]提出了适用于异构、周期任务的容错调度算法。与文献[1,2]相同,算法依据实时任务状态,副版本执行方式为主动或是被动方式,因而其处理机利用率不高。主副版本重叠方式的副版本可以有效提高处理机利用率和减少处理机使用数量,如文献[5]提出了一种适应性的容错调度算法,通过调节主/副版本间的重叠长度在系统可靠性和性能间取得均衡。文献[6]提出了基于抢占式、周期任务的容错算法,该算法使用被动和主副重叠方式执行副版本,同时依据任务负载不同将任务分成不相交子集进行调度,然而这些算法忽略了系统可靠性调度问题,任务可能分配到失效率较高的节点上,从而降低了系统调度性,增加了使用处理机的数量。文献[7]提出了优先级约束、可靠性代价驱动的容错调度方法,该算法

① 863 计划(2006AA01A103,2008AA01A201, 2009AA01A404)资助项目。

② 男,1979 年生,博士生;研究方向:容错计算,分布式计算;E-mail: nefujwp@gmail.com

③ 通讯作者,E-mail: wzb@ftcl.hit.edu.cn

(收稿日期:2010-11-19)

强调“强主版本复制”,要求任务必须收到它所有前驱节点结果,因而该算法只考虑任务的前驱节点,而没有考虑所有节点任务的完成问题。文献[8]提出了可靠性驱动的实时容错调度算法,该算法忽略了系统被动副版本故障后的可靠性问题。可以看出,现有研究以主副版本策略为基础,在实现异构系统可靠性、可调度性的前提下,以减少处理机数量为目标实现异构分布式系统容错性,较少有对系统 Makespan(调度时长)驱动下的容错调度问题的探讨。本文提出了一个基于优先级约束的容错调度方法——DRCAMD,它在满足系统可调度性前提下,将可靠性与 Makespan 作为可调节的局部目标函数,实现具有较高可靠性及较短执行时间的容错调度方法,避免了将任务分配到失效率较高的节点上执行。此算法的副版本采用被动和主副重叠方式执行,使得容错调度算法具有较大的灵活性。仿真实验表明,该算法性能优于现有算法。

1 容错调度模型

本文考虑典型的异构多处理机和实时任务集构成的异构分布式系统。以下进行形式化定义:

定义 1 异构分布式系统一组处理机集合描述为 $P = \{P_1, P_2, \cdots, P_M\}$,其中 M 代表处理机数。

定义 2 一组实时周期任务集合描述为 $V = \{V_1, V_2, \cdots, V_N\}$,$V_i = \{S_i, C_i, D_i, T_i\}$,$i = 1,2,\cdots,N$,其中 V_i 代表第 i 个任务,S_i 表示任务 i 的开始时间,C_i 表示任务 i 的执行时间,C_{ij} 表示任务 i 在处理机 P_j 上执行时间。D_i 表示任务 i 的截止时间,T_i 为任务 i 的周期,V_i^P 表示任务 i 的主版本,V_i^B 表示任务 i 的副版本。

定义 3 在本模型中任务 i 的副版本 V_i^B 有两种执行方式:被动方式和主副版本重叠方式,其状态依据任务执行时间与周期比值确定,如公式

$$Status(V_i^B) = \begin{cases} passive, & C_i/T_i \leqslant 0.5 \\ overlapping, & C_i/T_i > 0.5 \end{cases} \quad (1)$$

所示。

定义 4 $m \times n$ 矩阵 $\boldsymbol{X}$ 为主版本任务与处理机映射关系,矩阵 $\boldsymbol{X}^B$ 为副版本任务与处理机映射关系。它表示将第 n 个主/副版本任务指定分派到 m 个处理机上,即如果 $X_{ij} = 1$,表示任务 V_i 被分派到处理机 P_j 上。如公式(2)所示:实时任务 V_1, V_2, V_3, V_4, V_5, V_6 分别被映射到处理机 P_2, P_1, P_3, P_1, P_2, P_3 上;实时副版本任务 V_1^B, V_2^B, V_3^B, V_4^B, V_5^B, V_6^B 分别被映射到处理机 P_3, P_3, P_1, P_2, P_3, P_1 上。

$$\boldsymbol{X} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \boldsymbol{X}^B = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \quad (2)$$

定义 5 设任务 V_i 的前驱节点为 $pred(i)$,V_i 必须收到其所有前驱节点消息后方可执行调度,同时忽略通信争用带来的延迟。

为了更好地讨论所关心的问题,本文做如下假设:(1)任务切换和进程调度所使用的 CPU 时间忽略不计;(2)任何时刻只存在一个处理机故障,不可能同时存在两个处理机出现故障情况;(3)处理机的故障是 fail-stop 模式,即处理机的工作状态为正常或者是故障停止,同时忽略故障检测的时间;(4)在每个处理机上的任务集按照 RM[9] 算法进行调度;(5)为了减少调度所需要的时间,将处理机集合划分成两个不相交的集合进行调度,即主版本任务在一个集合中、副版本任务在另一个集合中。

2 算法描述

2.1 可靠性代价驱动

假设第 i 个处理机出错故障符合参数 λ_i 的泊松过程,并设 f_j 为处理机 P_j 故障概率,则在时间段 $[0, t]$ 分布式系统节点可靠性 $\mathrm{Pr}_0 = \prod_{j=1}^{M}\prod_{i=1}^{N}\exp(-f_j X_{ij} C_{ij})$。若第 f 个处理机出现故障时,节点可靠性为

$$\mathrm{Pr}_f = \prod_{j\neq f, j=1}^{M}\prod_{i=1}^{N}\exp[-f_j C_{ij}(X_{ij} + X_{ij}X_{ij}^B)] \quad (3)$$

设 g_{ab} 表示处理机 P_a 与处理机 P_b 之间链路的故障概率,d_{ab} 表示处理机 P_a 与处理机 P_b 之间传输单位数据长度所需的时间,W_{ij} 表示任务 V_i 到任务 V_j 之间传输的数据量。那么在时间段 $[0,t]$ 系统节点间链路可靠性如下式所示:

$$\mathrm{Pr}_{0L} = \prod_{a=1}^{M}\prod_{b=1}^{M}\prod_{i=1}^{N}\prod_{j=1}^{N}\exp(-g_{ab}X_{ia}X_{jb}W_{ij}d_{ab}) \quad (4)$$

若第 f 个处理机出现故障时,系统节点间链路可靠性如下式所示:

$$\mathrm{Pr}_{fL} = \prod_{a=1}^{M}\prod_{b=1}^{M}\prod_{i=1}^{N}\prod_{j=1}^{N}\exp(-g_{ab}X_{ia}X_{jb}W_{ij}d_{ab}) \times\{\prod_{a=1}^{M}\prod_{b=1}^{M}\prod_{i=1}^{N}\prod_{j=1}^{N}\exp[-g_{ab}X_{ia}(X_{jk}X_{jb}^{B})W_{ij}d_{ab}]\} \times\{\prod_{a=1}^{M}\prod_{b=1}^{M}\prod_{i=1}^{N}\prod_{j=1}^{N}\exp[-g_{ab}(X_{ik}X_{ia}^{B})(X_{jk}X_{jb}^{B})W_{ij}d_{ab}]\} \tag{5}$$

由此可以定义分布式系统可靠性 $R = \mathrm{Pr}_0 \times \mathrm{Pr}_{0L} + \mathrm{Pr}_f \times \mathrm{Pr}_{fL}$。系统的可靠性代价用公式表示为

$$\begin{aligned} RC = {} & \sum_{j=1}^{M}\sum_{i=1}^{N}f_jC_{ij}X_{ij} + \sum_{j=1}^{M}\sum_{i=1}^{N}f_jC_{ij}(X_{ij}+X_{ij}X_{ij}^{B}) \\ & + \sum_{a=1}^{M}\sum_{b=1}^{M}\sum_{i=1}^{N}\sum_{j=1}^{N}(g_{ab}W_{ij}d_{ab}X_{ia}X_{jb}) \\ & + \sum_{a=1}^{M}\sum_{b=1}^{M}\sum_{i=1}^{N}\sum_{j=1}^{N}\{g_{ab}W_{ij}d_{ab}[X_{ia}X_{jb} \\ & + X_{ia}(X_{jf}X_{jb}^{B}) + (X_{if}X_{ia}^{B})(X_{jf}X_{jb}^{B})]\} \end{aligned} \tag{6}$$

设 R_{ij} 表示任务 V_i 在处理机 P_j 上的可靠性代价,如公式

$$R_{ij} = (f_iC_{ij} + \sum_{p\in pred(i)}\sum_{q=1}^{M}g_{qj}X_{pq}W_{pi}d_{qj}) + [f_iC_{ij}X_{ij}^{B} + \sum_{p\in pred(i)}\sum_{q=1}^{M}g_{qj}W_{pi}d_{qj}(X_{pq}+X_{pq}^{B})] \tag{7}$$

所示。$pred(i)$ 为任务 V_i 的所有前驱节点。因此,提高系统可靠性,需要最小化处理机可靠性代价,在满足执行时间前提下,将任务调度到可靠性代价最小的处理机上执行。

2.2 Makespan 驱动

传统的 Makespan 驱动的启发式调度算法的调度目标是期待任务获得最短的任务结束时间,即获得较好的调度时长。这类算法一般使用任务 V_i 的预测执行结束时间作为目标函数。设 EAT 表示任务 V_i 最早可用时间,EST 表示任务最早开始执行时间。M_{ij} 表示任务 V_i 在处理机上 P_j 最短完成时间,如公式

$$M_{ij} = MAX\{O_p+\delta, MAX_{v_p^{P/B}\in pred(i)}EST(V_i^{B}, V_p^{P})\} + X_{ij}C_{ij} \tag{8}$$

所示。O_p 表示任务 V_i 直接前驱节点完成时间,即 $O_p = S_p + C_p, p \in pred(i)$。$MAX_{v_p^{P/B}\in pred(i)}EST(V_i^{B}, V_p^{B})$ 表示 V_i 收到直接前驱 V_p 消息后的最早开始时间,δ 表示链路传输延迟。

其副版本的最短完成时间如公式

$$M_{ij}^{B} = MAX\{O_p+\delta, MAX_{v_p^{P/B}\in pred(i)}(EST(V_i^{B}, V_p^{P}), MAX_{v_p^{P/B}\in pred(i)}EST(V_i^{B}, V_p^{B})\} + X_{ij}C_{ij} \tag{9}$$

所示。$MAX_{v_p^{P/B}\in pred(i)}EST(V_i^{B}, V_p^{P/B})$ 表示 V_i^{B} 收到直接前驱 V_p 消息后的最早开始时间。保证了 P_j 执行的任务能够收到其副版本发来的消息。

为了使调度算法在调度任务过程中能够同时考虑到任务的可靠性和 Makespan,需要将 R_{ij} 和 M_{ij} 合理地结合起来,构造同时包含这两项的代价函数,依据多目标优化函数的方法,给出下面的局部目标函数:

$$Cost_{ij} = \sqrt{(1-k)\cdot(\frac{R_{ij}}{\max\limits_{k\in Pred(i)}R_{ik}})^2 + k\cdot(\frac{M_{ij}}{\max\limits_{k\in Pred(i)}M_{ik}})^2} \tag{10}$$

该目标函数的 R_{ij} 是可靠性,M_{ij} 是关于最短调度时长。因为在很多情况下处理能力强的计算资源的可靠性不一定强,在这种情况下,R_{ij} 和 M_{ij} 是冲突的,无法使两个目标都获得最好的调度结果,因此在目标函数中增加了一个权重函数 $k(0 \leqslant k \leqslant 1)$,用户可以依据对性能的需求不同对调度目标进行调节。

2.3 可调度性分析

文献[10]提出并证明了完成时间测试定理(CTT),该定理表明,对于固定优先级调度的实时周期任务集,如果任务集的每一个任务的最坏响应时间都小于它们各自对应的截止期,那么该实时周期任务集是可调度的。文献[3]提出了适用于抢占式、主副版本调度的容错测试定理(FTCTT),然而该定理没有考虑任务优先级约束,仅通过判断最低优先级任务是否满足截止期判断任务可调度性,不适合本文提出的容错调度模型。因此本文提出并证明符合本文模型的容错测试定理。

由于将处理机进行分组,因此在判断任务可调度分析时不需要考虑主版本与副版本(被动/重叠)各种状态。只需考虑一种状态进行测试,以减少调度时长。

引理 1 主版本任务 V_i^{P} 在处理机 P_j 上可调度的唯一条件是只需判断 V_i^{P} 与所有被指定到 P_j 上的其他任务是否可调度:

$$\max_{V_i^{P}\in primary(p_j)}\min_{0<t\leqslant T_i-C_i}\{\sum_{1\leqslant r\leqslant i}C_r\cdot\lceil t/T_r\rceil/t\}\leqslant 1 \tag{11}$$

证明:在 $primary(p_j)$ 中只有主版本任务,因而当处理机出现故障时,所有的主版本任务只要在 $T_i - C_i$ 时间内被执行完毕,则是可调度的。

引理 2 被动副版本任务 V_i^{Backup} 被指定到处理机 P_f 上执行,其对应主版本任务 V_i^{P} 被指定到 P_j 执

行即 $recover(p_f, p_j)$，则其可调度的唯一条件是只需判断 V_i^{Backup} 与所有指定到 P_f 上的副版本任务且其主版本在 P_j 是否可调度：

$$\max_{V_i^{Backup} \in recover(p_f, p_j)} \min_{0<t\leqslant T_i - W_i} \left\{ \sum_{1\leqslant r\leqslant i} C_r \cdot \lceil t/T_r \rceil / t \right\} \leqslant 1 \tag{12}$$

证明：与主副本不同，对被动复制不但要求 V_i^{Backup} 所有更高的优先的任务能被执行完毕，而且要求必须在 $T_i - W_i$ 时间内执行完毕，则是可调度的。

引理 3 重叠副版本任务 $V_i^{overlap}$ 被指定到处理机 P_f 上可调度的唯一条件是只需判断 $V_i^{overlap}$ 与所有被指定到 P_f 上的其他任务是否可调度：

$$\max_{V_i^{overlap} \in overlap(p_f)} \min_{0<t\leqslant C_i} \left\{ \sum_{1\leqslant r\leqslant i} (2C_r - T_r) \cdot \lceil t/T_r \rceil / t \right\} \leqslant 1 \tag{13}$$

证明：由引理 1 可知只需判断 V_i^p 与所有被指定到 P_j 上的其他任务是否可调度。当主版本发生故障，V_i^p 在 P_j 上重叠部分 $2C_r - T_r$ 必须在 $C_r - T_r$ 时间内执行未完成的部分任务。因此 $V_i^{overlap}$ 能在 C_r 前完成。

3 仿真实验与结果分析

仿真实验从所需处理机数量、Makespan 及系统可靠性三个方面对算法的性能进行评价。将本文提出的算法 DRCAMD 与文献[11]提出的 FTRMFF 算法及文献[5]提出的 TPFTRM 算法进行了比较，并对得到的仿真结果进行了分析。

模拟实验的具体参数设置如下：(1)每组测试中实时周期任务集合 $100 \leqslant N \leqslant 1000$。(2)主副版本任务周期在[1,100]内均匀分布。(3)所有任务的最大负载 $\alpha = \max\{U_1, U_2, \cdots, U_N\}$，其中 $U_i = C_i/T_i$（U_i 为第 i 个任务的负载，C_i 为任务 i 的执行时间，T_i 为任务 i 的周期）；性能度量标准为给定任务集进行调度所需处理机数与任务集负载和的比值。定义负载和 $U = U_1 + U_2 + \cdots + U_N$，由于任务的最大负载与负载和已知，因而 M（处理机数）$/U$ 越小，反映算法性能越优良。(4)任务的执行时间在 $[1, \alpha T_i]$ 之间均匀分布，α 值分别设为 0.5,0.8。实验重复进行10次，以10次的平均值为最终结果。

算法 1 可靠性代价和 Makespan 驱动容错调度算法

算法 DRCAMD

1: $P = \{P_1, P_2, \cdots, P_M\}$; (* set of processors *)
2: $S = \varphi$; $U = V$
3: compute priority in U
4: $H(1) = \max(U)$
5: while $U \neq \varphi$
6: $V_i = H(1)$
7: compute $f(V_i^p, P)$
8: compute $f(V_i^B, P)$
9: compute R_{ij}
10: compute $Cost_{ij}$
11: If $\eta = 0$, $Cost_{ij}$ is reduced to find a high reliability
12: if $\eta = 1$, $Cost_{ij}$ is reduced to find makespan.
13: If $\eta = 0.5$, the two objectives have the same importance
14: compute min $\{Cost_{ij}\}$
15: schedule V_i to the corresponding processor
16: put V_i in S
17: update the priority in U
18 :end while

3.1 所需处理机数

首先将无故障条件下处理机的数目与任务负载比值作为性能度量的标准，图 1 和图 2 分别表示 $\alpha = 0.5$，$\alpha = 0.8$ 时的仿真结果。可以看到 DRCAMD 具有较好的处理机利用率。并且当 $\alpha \leqslant 0.5$ 时，DRCAMD 使用副版本进行调度，其所需的处理机较采用延迟副版本的 FTRMFF 明显减少，另外，由于采用相同的分组策略及副版本调度方法，DRCAMD 与 TPFTRM 所需的处理机数基本相同。

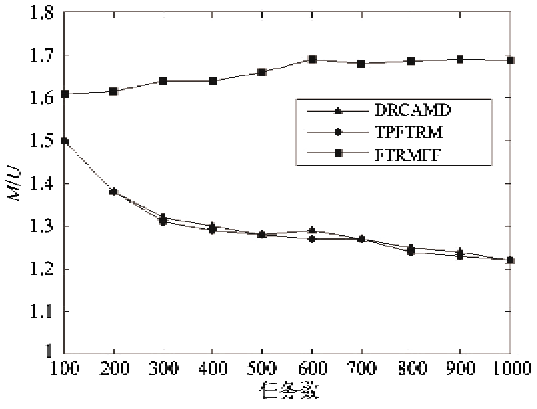

图 1 $\alpha = 0.5$ 时三种算法性能的比较

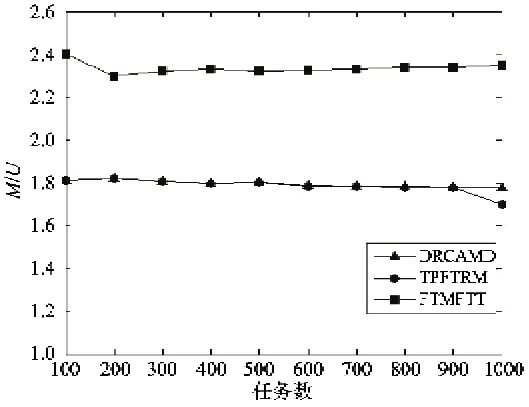

图 2　α =0.8 时三种算法性能的比较

3.2　不同失效概率系统可靠性

为了更好地衡量算法的可靠性,设节点最小失效率 $MIN\_F = 10^{-6}/h$,最大失效率 MAX_F 为 $3.5 \times 10^{-6}/h$ 到 $7.5 \times 10^{-6}/h$。每小时增加 $0.5 \times 10^{-6}/h$,链路失效概率 $LINK\_F$ 为 $0.65 \times 10^{-6}/h$ 到 $0.95 \times 10^{-6}/h$。由图 3 可以看到任务数为 100,处理机数量为 20 时的系统可靠性变化情况。与 FTRMFF 及 TPFTRM 算法相比,DRCAMD 具有较高的可靠性,其可靠性分别提高了 10.5% 和 22.3%。这是由于 TPFTRM 只考虑节点的可靠性问题而忽略链路之间的通信可靠性问题,而 FTRMFF 忽略可靠性问题,仅对性能进行调度分析。

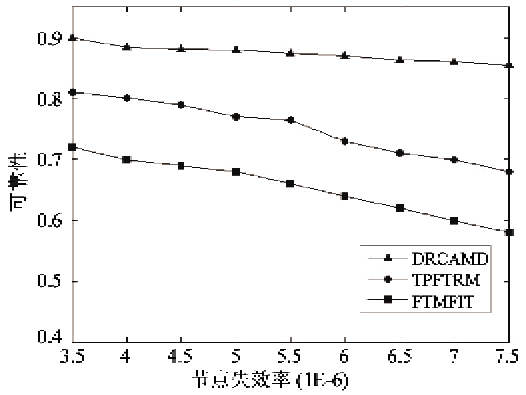

图 3　三种算法在不同失效概率下的可靠性比较

3.3　Makespan 与可靠性分析

在针对不同数目的任务集进行实验时,处理节点数为 10,任务数范围是 $100 \leqslant N \leqslant 1000$;在对不同数目的处理节点进行实验时,任务数为 300,处理节点为 $10 \leqslant M \leqslant 50$。依据式(9)分别选取不同的 k 的取值,分析 k 值变化与 Makespan 与可靠性变化情况。

从图 4、图 5 可以看到,随着任务数目的增加,系统的 Makespan 线性增加,可靠性线性降低,这是因为随着任务数目的增加,应用计算负载会随之增加,执行时间增长,执行过程中出现资源失效的概率也会增加,因此系统的可靠性随之下降。在任务计算量相同的情况下,k =1 时,DRCAMD 算法都获得了最优的 Makespan,但获得的应用可靠性却最低,这是因为这时算法都仅关注最小化应用的 Makespan;k =0 时 DRCAMD 算法获得了最高的可靠性,但是 Makespan 却最差,这是因为当 k =0 时,算法 DRCAMD 实际上仅考虑到最大化应用的可靠性;随着权重参数 k 的增加,算法获得的可靠性逐渐降低,获得的 Makespan 逐渐变优;该结果表明在局部代价函数中系统可靠性的数据项能够有效地引导调度算法提高可靠性,可靠性的提高是以增加应用的 Makespan 为代价的。同时表明可靠性和 Makespan 调节目标的不可调和性,因而在具体的实际应用中可以通过调节权重参数 k 来平衡这两个目标,以满足不同的需求。

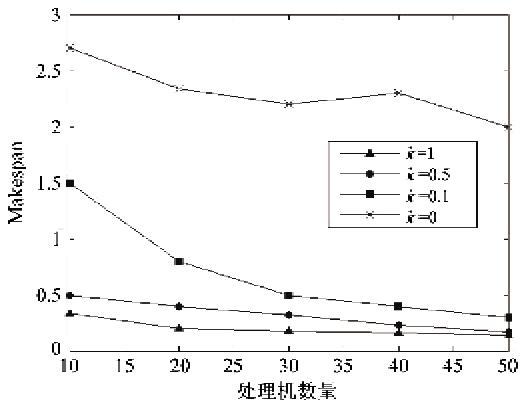

图 4　不同 k 值的处理机数量与 Makespan 值变化

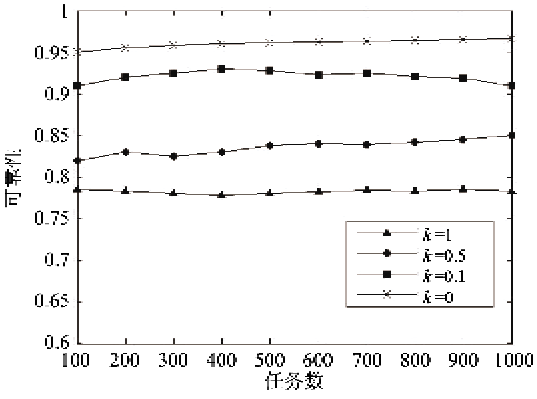

图 5　不同 k 值的任务数与 Makespan 值变化

4 结 论

针对分布式环境中主副版本调度策略可靠性调度问题,提出了一种考虑节点及链路可靠性和Makespan 驱动的分布式容错调度方法 DRCAMD。该方法通过设置 k 值来调整目标权重函数,平衡不同的系统需求。提出了一种适于本文中容错调度模型的容错测试定理。该方法仅判断副版本可调度性不需要考虑主版本与副版本(被动/重叠)各种状态。只需考虑一种状态进行测试,减少调度时长。仿真实验证明该方法的有效性和可行性。本文提出的调度策略是一种适于分布式环境的可靠性与Makespan 结合的调度策略,因而可将其应用在云计算环境中。下一步工作包括:探讨云计算环境下资源需求约束及计算费用、可靠性等目标的调度问题,采用更加高效的任务分配算法,寻求计算资源高效利用。

参考文献

[1] Qin X, Han Z F, Pang L P. Real-time scheduling with fault-tolerance in heterogeneous distributed systems. *Chinese Journal of Computers*, 2002, 25(1):121-124

[2] Qin X, Hong J. Dynamic, reliability-driven scheduling of parallel real-time jobs in heterogeneous systems. In: Proceedings of 2001 International Conference on Parallel Processing, Valencia, Spain, 2001

[3] 岁威,阳富民,庞丽萍等. 异构分布式系统中实时周期任务的容错调度算法. 计算机学报,2007,30(10):1740-1749

[4] Luo W, Yang F M, Pang L P, et al. Fault-tolerant scheduling based on periodic tasks for heterogeneous systems. *Lecture Notes on Computer Science*, 2006, 4158: 571-580

[5] Al-Omari R, Somani A K, Manimaran G. An adaptive scheme for fault-tolerant scheduling of soft real-time tasks in multiprocessor systems. *Journal of Parallel and Distributed Computing*, 2005, 65 (5): 595-608

[6] Wang J, Sun J L, Wang X Y, et al. Efficient scheduling algorithm for hard real-time tasks in primary-backup based multiprocessor systems. *Journal of Software*, 2009, 10: 2628-2636

[7] Qin X, Jiang H. A novel fault-tolerant scheduling algorithm for precedence constrained tasks in real-time heterogeneous systems. *Parallel Compute*, 2006, 32(6):331-356

[8] Luo W, Yang F M, Pang L P, et al. Fault-tolerant scheduling based on periodic tasks for heterogeneous systems In: Proceedings of the 3rd International Conference on Autonomic and Trusted Computing, Wuhan, China, 2006. 571-580

[9] Liu C L, Layland J W. Scheduling algorithms for multiprogramming in a hard real-time environment. *Journal of ACM*, 1973, 20 (1): 46-61

[10] Klein M H, Lehoczky J P, Rajkumar R. Rate-monatomic analysis for real-time industrial computing. *Computer*, 1994, 27(1):24-33

[11] Bertossi A, Mancini L V, Rossinil F. Fault-tolerant rate-monotonic first-fit scheduling in hard-real-time systems. *IEEE Trans on Parallel and Distributed Systems*, 1999, 10(9):934-945

A reliability-cost and Makespan driven fault-tolerant scheduling algorithm for distributed systems

Jing Weipeng[*,**], Wu Zhibo[*], Liu Hongwei[*], Dong Jian[*]

([*]School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150080)

([**]The College of Information and Computer Engineering, Northeast Forestry University, Harbin 150040)

Abstract

To solve the reliability scheduling problem of primary-backup in heterogeneous distributed computing systems, the paper puts forward the DRCAMD, a fault-tolerant scheduling algorithm for distributed systems based on priority constraints of reliability-cost and Makespan (the schedule length) driven. Under the premise of meeting schedulability, the algorithm realizes a higher reliability and shorter execution time fault-tolerant scheduling strategy with the heterogeneous distributed environment nodes, at the same time it can avoid allocating the tasks to the nodes of higher failure rate for execution. In addition, the algorithm of minor version can execute in passive and overlap between main and side, making the fault-tolerant scheduling implemented with the greater flexibility. And the simulation result shows that the DRCAMD outperforms the exiting fault-tolerant scheduling algorithms.

Key words: primary-backup, precedence constrain, higher failure rate, overlapping backup copy