异构分布式系统 DAG 可靠性模型与容错算法

谢国琪 李仁发 刘 琳 杨 帆

(湖南大学嵌入式与网络计算湖南省重点实验室 国家超级计算长沙中心 长沙 410082)

摘 要 异构分布式系统性能得到大幅度提升的同时,却造成故障率大增,以有向无环图(Directed Acyclic Graph, DAG)任务模型研究异构分布式系统的容错调度成为当前的研究热点.广泛采用的基于任务复制的容错算法存在以下问题:(1)DAG 任务可靠性需求与 DAG 可靠性需求的约束存在缺陷且缺乏严谨的理论证明;(2)每个任务仅有一个副版任务,不足以应对任务潜在的多次发生的故障;(3)盲目地使每个任务拥有 $\varepsilon+1$ 个副版来容忍可能的 ε 个故障,虽然提高了系统的可靠性但易造成系统冗余度过高,并付出昂贵的计算资源.文中首先分析 DAG 图中任务依赖关系,确定 DAG 任务的可靠性概率模型,并建立 DAG 可靠性模型;接着提出满足可靠性目标的任务复制下限值算法、经济的任务复制策略算法和贪婪的任务复制策略算法,精确量化各个任务需要复制的次数,最后在上述算法的基础上提出可选策略的 DAG 容错算法 OPDFT(Optional Policy on DAG Fault-Tolerant).实验表明,OPDFT 算法的经济复制策略和贪婪复制策略的可靠性代价分别是盲目策略算法可靠性代价的 60%和 70%左右.

关键词 异构分布式系统;可靠性;容错;有向无环图;任务复制

中图法分类号 TP302 **DOI 号** 10.3724/SP.J.1016.2013.02019

DAG Reliability Model and Fault-Tolerant Algorithm for Heterogeneous Distributed Systems

XIE Guo-Qi LI Ren-Fa LIU Lin YANG Fan

(*Key Laboratory for Embedded and Network Computing of Hunan Province, National Supercomputing Center in Changsha, Hunan University, Changsha* 410082)

Abstract The performance of heterogeneous distributed systems has been improved significantly, but caused increased failures dramatically. Tolerant scheduling in heterogeneous distributed systems with DAG (Directed Acyclic Graph) task model becomes a research focus. Widely used fault-tolerant algorithms based on task replication have following problems: (1) there are some deficiencies and lack of rigorous proof on constraint between DAG task reliability requirement and DAG reliability requirement; (2) only one backup copy of each task, which not enough to cope with potential repeated failures; (3) blindly to tolerate ε faults of each task with $\varepsilon+1$ backup copies, which improved the reliability of system, but caused high redundancy and resources consumption. Firstly, task dependencies of DAG are analyzed, then the DAG task reliability probability model is determined and based on this, the DAG reliability model is constructed. Secondly, lower limit of task duplication algorithm, economic task duplication strategy algorithm and greedy algorithm for task replication strategy are presented to meet the reliability target of DAG and

收稿日期:2013-06-26;最终修改稿收到日期:2013-08-31.本课题得到国家自然科学基金重点项目(61133005)、国家自然科学基金面上项目(61173036,61070057,61272061)、国家自然科学基金青年科学基金项目(61202102)、国家"八六三"高技术研究发展计划项目基金(2012AA01A301-01)资助.**谢国琪**,男,1983 年生,博士研究生,中国计算机学会(CCF)学生会员,主要研究方向为计算机系统结构、容错计算.E-mail: xgqman@126.com.**李仁发**,男,1957 年生,博士,教授,博士生导师,中国计算机学会(CCF)杰出会员,主要研究领域为计算机系统结构、容错计算、计算机网络.**刘 琳**,女,1988 年生,硕士研究生,主要研究方向为计算机系统结构、容错计算.**杨 帆**,男,1985 年生,博士研究生,中国计算机学会(CCF)学生会员,主要研究方向为计算机系统结构、容错计算、嵌入式建模.

achieve precise quantification for each task's replicas. Finally, the OPDFT (Optional Policy on DAG Fault-Tolerant) algorithm is proposed based on above 3 algorithms. Experiments show that the reliability cost of economic policy and greed policy of OPDFT algorithm is about 60% and 70% of blind strategy respectively.

Keywords heterogeneous distributed systems; reliability; fault-tolerant; DAG; task replication

1 引 言

计算机系统日益走向异构化与分布式.作为与人们生活息息相关的汽车,其内部的总线系统是典型的异构分布式嵌入式计算系统,部分轿车中的电子控制单元(Electronic Control Unit,ECU)的数量超过100个,并通过CAN、FlexRay、LIN、MOST等多种车辆总线互联[1],系统规模和复杂性骤增.近年来,将CPU与具有超高的计算性能和性能功耗比的GPU集成构建异构分布式系统已成为超级计算机发展的新趋势之一[2].

然而,处理器制造工艺的发展,使得单芯片上集成的晶体管尺寸越来越小且晶体管的数量呈指数性增长,在处理器性能得到大幅度提高的同时,系统的故障率急剧增加.汽车行驶过程中存在的各种干扰也容易出现瞬时故障.安全关键系统(如防抱死制动系统)如果因为故障而错过截止期限,则会造成灾难性的后果[3].在提供强大的计算性能的同时,GPU集成了大量的功能部件且运行时温度较高,容易出现瞬时故障,可靠性大大低于传统CPU[2].因此必须采用一定的容错手段来保障异构分布式系统的可靠性,使得在出现局部故障的情况下,仍然能够产生正确的结果.由于异构分布式系统的调度在各处理器上执行时间不同以及处理器间存在通信开销,各任务之间还存在优先级约束和数据依赖关系,因而可以将应用映射成有向无环图DAG[4].以DAG任务模型研究计算机系统的容错问题已成为当前研究的热点[5-13].

2 相关研究

在异构分布式系统中,任务复制技术是实现容错的主要手段,最具代表性的主/副版技术(Primary/Backup Copy,P/B)广泛应用于容错调度方法[14].它通过在备份处理器上执行备份任务来实现容错.P/B复制技术有3种执行方式:主动复制方式(Active Backup Copy)[10-13]、被动复制方式(Passive Backup Copy)[5-9]和混合复制方式(P/B Overlapping Backup Copy)[14].目前学术界对DAG任务容错调度的研究也都是基于任务复制机制,针对副版数复制数量来区分,主要有2种复制方式.

(1)每个任务仅有一个副版.

文献[5]提出了DAG可靠性代价驱动的eFRCD(efficient Fault-tolerant Reliability Cost Driven)算法,该算法对DAG中的每一个任务都有一个分配在不同处理器上的副版任务.为了提高性能,对于主版不在同一个处理器上的多个任务,系统允许在同一个处理器上的这些任务的副版可以重叠.然而,这种方法必须假设这些任务之间是相互独立的,不能满足DAG中有优先级约束任务的需要,因此在eFRCD算法基础上,作者在文献[6]又提出了改进的eFRD(efficient Fault-tolerant Reliability Driven)算法,该算法采用主副版重叠机制,即允许任务的副版与此任务的所有后继任务的主版重叠,可以进一步降低调度长度.文献[7]在eFRD算法的基础上提出了基于最早完成时间的最小复制开销的MRC-ECT(Minimum Replication Cost with Early Completion Time)算法和基于最小复制开销的最小完成时间的MCT-LRC(Minimum Completion Time with Less Replication Cost)算法分别对DAG中的非独立任务和独立任务进行容错调度.

首先,上述研究针对可靠性问题,假设某一个时刻最多只有一个处理器出现故障,且在下个故障出现时,前一个故障已经排除,假设较为理想导致实用性不强.同时也只考虑DAG的可调度性,没有考虑可靠性目标.Lin等人在文献[8]中给出了可靠性目标的定义,即系统任务集里的每个任务都有ε个副版,一个任务成功分配的条件是该任务的ε个副版分配到不同的处理器上,且没有导致这些处理器的利用率超过1,在满足任务副版时间约束和系统高可靠性条件的基础上,最大化成功的分配任务.

其次，上述算法都是采用被动复制方式，只有当主版任务调度失败后才能启动副版任务，被动复制在主版任务失效时，需要选择一个新副版任务恢复到失效前状态，造成失效恢复时间较长. 因此从截止期限和失效恢复时间考虑，主动复制优于被动复制[9]. 主动复制能够在运行失效时直接屏蔽失效的任务版，失效恢复时间几乎接近于零，调度长度也相对较短.

(2) 每个任务有多个副版.

文献[10]提出基于主动复制的 FTSA(Fault Tolerant Scheduling Algorithm)算法，此算法是经典的非任务复制的 DAG 调度算法 HEFT(Heterogeneous Earliest Finish Time)的扩展. FTSA 容忍 ε 个错误的发生，并且有 $\varepsilon+1$ 个版本允许在不同的处理器上，但 FTSA 算法每次只选择一个优先级最高的就绪任务调度. 作者又在文献[11]中提出了同样基于主动复制的 CAFT(Contention Awareness and Fault-Tolerant)算法，与 FTSA 算法每次只选择一个优先级最高的就绪任务不同，CAFT 选择一组就绪任务，在同一决策过程中分配其所有副版到相应的处理器，这样能够产生更好的负载均衡. 但是 FTSA 和 CAFT 算法为了使系统能够达到容忍多个故障，采用了盲目的复制策略，即对于每个任务需要复制的版本个数，并没有精确的量化，而是盲目地使每个任务拥有 $\varepsilon+1$ 个副版，容忍系统中任务可能存在的 ε 个故障. 虽提高了系统的可靠性却易造成任务冗余过高，使得调度过程中既付出了昂贵的计算资源，又造成调度长度过长而可能错过任务的截止期限. 文献[12]则将盲目的复制策略应用到多个 DAG，更加剧了系统的冗余程度过高.

针对主动复制采用 $\varepsilon+1$ 个副版容忍 ε 个故障的情况，文献[13]首次指出：拥有更多的副版并不意味更高的可靠性. 其提出的 MaxRe 算法针对不同的任务，基于可靠性目标而采用不同的副版次数，在满足系统的可靠性目标的前提下，能够最小化资源的使用. 但是该算法认为每个任务的可靠性目标为 $r=\sqrt[n]{R}$，其中 R 为 DAG 的可靠性需求，n 为任务个数，但是由于 DAG 中的任务存在优先级约束，对于有前驱的任务，需要考虑其直接前驱任务的影响，不能依靠简单的开方就能断定.

本文旨在通过 DAG 可靠性概率模型的有效建模以及在可靠性目标的约束条件下，计算DAG任务复制下限值，提出有效的任务复制选择策略，精确量化分析每个任务需要的版本个数，在满足可靠性目标的前提下，减少任务冗余，最小化可靠性代价，降低调度长度以提高性能.

3　可靠性模型

3.1　DAG 任务模型

DAG 任务模型中的结点表示任务，有向边表示任务间的依赖和通信关系[15]. 用 $G=\{N,E,W,C\}$ 表示 DAG. 其中 $N=\{n_1,n_2,\cdots,n_i\}$ 表示图中结点的集合，也就是任务的集合. $E=\{e_{1,2},e_{2,3},\cdots,e_{i,j}\}$ 表示图中有向边的集合，边表示任务间的优先级与数据依赖关系；$P=\{p_1,p_2,\cdots,p_k\}$ 表示异构处理器的集合. $\boldsymbol{W}$ 是一个 $|N|\times|P|$ 的矩阵，$w_{i,k}$ 表示任务 n_i 在 p_k 上的计算时间开销. $c_{i,j}$ 表示任务 n_i 与 n_j 间的通信开销，如果将 2 个任务分配到同一个处理器，则通信开销为 0. $pred(n_i)$ 表示结点 n_i 的直接前驱结点集合，$ind(n_i)$ 表示 n_i 的入度，也就是其直接前驱结点集合的个数，当前任务只有在它全部前驱结点的数据到达后才能执行. $succ(n_i)$ 表示结点 n_i 的直接后继结点集合，$outd(n_i)$ 表示 n_i 的出度，也就其直接后继结点集合的个数. 没有前驱结点的结点为入口结点，记为 n_{entry}. 没有后继结点的结点为出口结点，记为 n_{exit}. 如图 1 所示为一个包含 7 个任务结点的 DAG，表 1 为相应的计算开销矩阵.

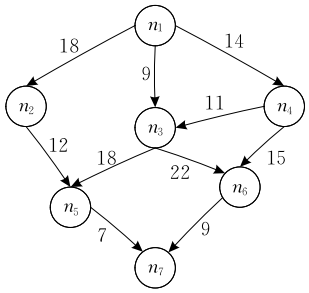


图 1　DAG 任务模型实例 *DAG_1*

表 1　*DAG_1* 计算开销矩阵

任务	p_1	p_2	p_3
n_1	14	16	9
n_2	13	19	18
n_3	11	13	19
n_4	13	8	17
n_5	12	13	10
n_6	13	16	9
n_7	7	15	11

3.2 DAG 任务可靠性模型

任务的可靠性为系统中任务无故障运行的概率.一般地,DAG 任务发生故障的概率服从泊松分布(Poisson Distribution)[5-14].泊松分布是一种离散型概率分布,用以描述在时间段$[0,t]$内 $\gamma(\gamma\geq 0)$个事件发生的概率.它假定事件以一个恒定速率 λ 到达且事件是相互独立的,那么在时间段$[0,t]$内,出现 $\gamma(\gamma\geq 0)$个事件的概率为

$$Error_{\gamma}(t)=\frac{e^{-\lambda t}\times(\lambda t)^{\gamma}}{\gamma!} \tag{1}$$

若 $\gamma=0$,即在时间段$[0,t]$内事件没有发生,根据式(1)可得事件没有发生故障的概率可表示为

$$Error_{\gamma=0}(t)=e^{-\lambda t} \tag{2}$$

若任务 n_i 发生故障的均值为 λ_i,那么运行在处理器 p_u 上的任务 n_i 在其执行时间 $w_{i,u}$ 内无故障概率可表示为

$$NError(n_i,p_u)=e^{-\lambda_i w_{i,u}} \tag{3}$$

由于任务相互之间存在依赖关系,且当前任务只有在它全部前驱任务结点的数据到达后才能执行,可得出任务的可靠性不是由式(3)所能表示的,还必须与它所有的前驱结点相关.

(1)无容错机制下的任务可靠性

在无容错机制下,系统中只有每个任务的主版存在,对于没有前驱结点的入口结点 n_{entry} 而言,其可靠性仅由式(3)决定;对于入度 $ind(n_i)$ 不为零的任务 n_i,其可靠性计算还需要考虑到其直接前驱结点的影响,因此无容错机制下,DAG 任务 n_i 的可靠性计算公式为

$$\begin{cases}CR(n_{\text{entry}},p_u)=e^{-\lambda_{\text{entry}}\times w_{\text{entry},u}}\\ CR(n_i,p_u)=\\ \qquad NError(n_i,p_u)\times\prod\limits_{n_x\in pred(n_i),p_u\in P}CR(n_x,p_u)\end{cases} \tag{4}$$

(2)容错机制下的任务可靠性

在容错机制下,系统中不仅有每个任务对应的主版任务,还有它们的副版任务.我们定义 k_i 为任务复制因子,表示任务 n_i 拥有的副版任务个数为 k_i,相应的主副版任务集合为 $\{n_i^0,n_i^1,n_i^2,\cdots,n_i^{k_i}\}$,其中 n_i^0 为主版任务,剩余的为副版任务.在主副版任务集合中,只要有一个主版或副版任务执行成功就代表任务 n_i 在其运行时间段内没有发生故障,那么任务的无故障概率可表示为

$$NError'(n_i)=1-\prod_{l_i=0,p_u\in P}^{k_i}(1-NError(n_i^{l_i},p_u)) \tag{5}$$

对于没有前驱结点的入口结点任务 n_{entry},其可靠性可由式(5)表示;对于入度 $ind(n_i)$ 不为零的任务 n_i,可靠性由 $NError'(n_i)$ 和它所有的直接前驱结点任务共同决定.因此容错机制下,DAG 中任务 n_i 的可靠性计算如下:

$$\begin{cases}CR'(n_{\text{entry}})=1-\prod\limits_{l_i=0,p_u\in P}^{k_i}(1-NError(n_i^{l_i},p_u))\\ CR'(n_i)=NError'(n_i)\times\prod\limits_{n_x\in pred(n_i)}CR'(n_x)\end{cases} \tag{6}$$

3.3 DAG 可靠性模型

文献[13]中的 DAG 可靠性模型中,任务可靠性需求与 DAG 可靠性需求的约束计算公式忽略了对于有前驱任务的任务,还需要考虑其前驱任务对此任务的可靠性造成的影响.而 DAG 任务模型本身以任务之间优先级约束为特点,因此本节充分考虑 DAG 的任务优先级约束,创新性地从理论证明和实例验证 2 个方面提出和验证 DAG 可靠性模型.

定理 1. DAG 可靠性概率模型.若 DAG 中每个任务 n_i 拥有副版任务的个数为 k_i,那么该 DAG 可靠性计算的概率模型可表示为

$$GR_{\text{DAG}}=\prod_{n_i\in N}(NError'(n_i))^{Road_i} \tag{7}$$

其中 $Road_i$ 表示从任务结点 n_i 到出口任务结点 n_{exit} 所有路径的个数,$Road_i$ 的值可以通过深度优先遍历(Depth-First-Search,DFS)算法[16]计算而得;$NError'(n_i)^{Road_i}$ 定义为可靠性贡献因式,即 DAG 可靠性是各个任务的可靠性贡献因式的乘积.

证明. 在 DAG 中,相关任务之间存在着优先级依赖关系.当前任务只有在它的全部前驱结点的数据完成之后才能执行.这就意味着若某个任务成功执行了,那么它所有的直接前驱结点任务也执行成功了.以此类推,可知对整个 DAG 而言,只要出口结点任务 n_{exit} 运行成功,那么此 DAG 就是可靠的.因此,出口结点任务 n_{exit} 的可靠性代表着整个 DAG 的可靠性.若 DAG 中每个任务 n_i 拥有副版任务的个数为 k_i,根据式(5)可知,DAG 的可靠性概率 GR_{DAG} 可表示为

$$\begin{aligned}GR_{\text{DAG}}&=CR'(n_{\text{exit}})\\&=NError'(n_{\text{exit}})\times\prod_{n_x\in pred(n_{\text{exit}})}CR'(n_x)\end{aligned} \tag{8}$$

对式(8)逐步进行扩展:

$$
\begin{aligned}
GR_{\mathrm{DAG}} &= NError'(n_{\mathrm{exit}}) \times CR'(n_{i-1}) \times \\
&\quad CR'(n_{i-2}) \times \cdots \times CR'(n_{i-a}) \\
&= NError'(n_{\mathrm{exit}}) \times NError'(n_{i-1}) \times \\
&\quad \prod_{n_{x1} \in pred(n_{i-1})} CR'(n_{x1}) \times NError'(n_{i-2}) \times \\
&\quad \prod_{n_{x2} \in pred(n_{i-2})} CR'(n_{x2}) \times \cdots \times NError'(n_{i-a}) \times \\
&\quad \prod_{n_{xa} \in pred(n_{i-a})} CR'(n_{xa}) \\
&= NError'(n_{\mathrm{exit}}) \times NError'(n_{i-1}) \times \\
&\quad NError'(n_{i-b}) \times \prod_{n_{xc} \in pred(n_{i-b})} CR'(n_{xc}) \times \cdots \times \\
&\quad NError'(n_{i-2}) \times NError'(n_{i-d}) \times \\
&\quad \prod_{n_{xe} \in pred(n_{i-d})} CR'(n_{xe}) \times \cdots \times NError'(n_{i-a}) \times \\
&\quad NError'(n_{i-f}) \times \prod_{n_{xg} \in pred(n_{i-f})} CR'(n_{xg}) \times \cdots \times \\
&\quad NError'(n_3) \times \cdots \times NError'(n_2) \times \cdots \times \\
&\quad NError'(n_1) \\
&= NError'(n_{\mathrm{exit}}) \times NError'(n_{i-1})^{Road_{i-1}} \times \\
&\quad NError'(n_{i-2})^{Road_{i-2}} \times \cdots \times \\
&\quad NError'(n_{i-q})^{Road_{i-q}} \times \\
&\quad NError'(n_{i-q-1})^{Road_{i-q-1}} \times \cdots \times \\
&\quad NError'(n_4)^{Road_4} \times NError'(n_3)^{Road_3} \times \\
&\quad NError'(n_2)^{Road_2} \times NError'(n_1)^{Road_1} \\
&= \prod_{n_i \in N} (NError'(n_i))^{Road_i} \qquad (9)
\end{aligned}
$$

最终得到整个 DAG 的可靠性计算的概率模型为式(7)的形式. 证毕.

接下来通过图 1 所示的 *DAG_1* 的可靠性计算过程实例来验证上述证明. 图 1 表明了任务之间的依赖关系,其中任务结点 n_1 为入口结点,任务结点 n_7 为出口结点 n_{exit}. 由图中任务结点的边关系可知,如果出口任务结点 n_7 执行成功,则整个 DAG 就执行成功.

一方面,假设在容错模式下每个结点任务拥有一定数量的副版任务,依据式(6),其可靠性的计算过程如表 2 所示. 如表中步 2,n_7 运行成功则代表它的前驱任务结点 n_5 和 n_6 也执行成功;n_5 依赖 n_2 和 n_3 执行成功,如步 3;同样如步 4,n_6 需要 n_3 和 n_4 执行成功,步 5～7 表示了 n_2 和 n_4 依赖于入口结点 n_1 执行成功,n_3 依赖于前驱 n_1 和 n_4.

表 2　*DAG_1* 可靠性计算过程

步骤	结点	*DAG_1* 的可靠性
1	$n_{\mathrm{exit}}=n_7$	$CR'(n_7)$
2	$pred(n_7)=\langle n_5,n_6\rangle$	$NError'(n_7)\times CR'(n_5)\times CR'(n_6)$
3	$pred(n_5)=\langle n_2,n_3\rangle$	$NError'(n_7)\times NError'(n_5)\times CR'(n_2)\times CR'(n_3)\times CR'(n_6)$
4	$pred(n_6)=\langle n_3,n_4\rangle$	$NError'(n_7)\times NError'(n_5)\times CR'(n_2)\times (CR'(n_3))^2\times NError'(n_6)\times CR'(n_4)$
5	$pred(n_2)=\langle n_1\rangle$	$NError'(n_7)\times NError'(n_5)\times NError'(n_2)\times CR'(n_1)\times (CR'(n_3))^2\times NError'(n_6)\times CR'(n_4)$
6	$pred(n_3)=\langle n_1,n_4\rangle$	$NError'(n_7)\times NError'(n_5)\times NError'(n_2)\times CR'(n_1)\times (NError'(n_3)\times CR'(n_1)\times CR'(n_4))^2\times NError'(n_6)\times CR'(n_4)$
7	$pred(n_4)=\langle n_1\rangle$	$(NError'(n_1))^6\times NError'(n_2)\times (NError'(n_3))^2\times (NError'(n_4))^3\times NError'(n_5)\times NError'(n_6)\times NError'(n_7)$

另一方面,DAG 中各个任务结点到出口结点 n_7 的路径以及个数 $Road_i$ 可以通过深度优先遍历算法计算得出,计算结果如表 3 所示.

表 3　*DAG_1* $Road_i$ 计算过程

任务	结点	路径条数
n_1	$n_1\to n_2\to n_5\to n_7$, $n_1\to n_3\to n_5\to n_7$, $n_1\to n_3\to n_6\to n_7$, $n_1\to n_4\to n_3\to n_5\to n_7$, $n_1\to n_4\to n_3\to n_6\to n_7$, $n_1\to n_4\to n_6\to n_7$	6
n_2	$n_2\to n_5\to n_7$	1
n_3	$n_3\to n_5\to n_7$, $n_3\to n_6\to n_7$	2
n_4	$n_4\to n_3\to n_5\to n_7$, $n_4\to n_3\to n_6\to n_7$, $n_4\to n_6\to n_7$	3
n_5	$n_5\to n_7$	1
n_6	$n_6\to n_7$	1
n_7	$n_7\to n_7$	1

根据定理 1,将路径个数 $Road_i$ 代入可靠性计算式(7),即可得到实例 *DAG_1* 的可靠性为

$$
\begin{aligned}
GR_{DAG\_1} &= (NError'(n_1))^6 \times NError'(n_2) \times \\
&\quad (NError'(n_3))^2 \times (NError'(n_4))^3 \times \\
&\quad NError'(n_5) \times NError'(n_6) \times \\
&\quad NError'(n_7) \qquad (10)
\end{aligned}
$$

将按照定理 1 获知的可靠性 GR_{DAG_1} 的表达式与表 2 中实际计算所得到的可靠性表达式对比,计算结果一样. 该实例的验证与定理 1 得出的结论一致,验证完毕.

4 容错算法

4.1 DAG 任务复制下限值算法

(1)任务复制次数下限值分析

定义 1. DAG 可靠性约束(DAG Reliability Constraint). 假定用户对系统的可靠性目标为 ω 且 $\omega<1$. 若 $GR\geq\omega$,则表示系统的容错方法已达到了用户的可靠性目标需求,即可认为系统是可信的. 那么将 DAG 的可靠性约束定义为

$$GR_{\mathrm{DAG}}=\prod_{n_i\in N}(NError'(n_i))^{Road_i}\geq\omega \quad (11)$$

为了满足 DAG 的可靠性目标值,采用主动任务复制的方式,那么必定存在一个任务复制次数的下限值 k_i^{down},使得每个任务实际复制的次数 k_i 必须满足 $k_i\geq k_i^{\mathrm{down}}$. 这里 k_i^{down} 是满足 $NError'(n_i)^{Road_i}\geq\omega$ 的最小值.

定理 2. 若系统中任意一个任务 n_i 的复制次数 $k_i<k_i^{\mathrm{down}}$,那么 $NError'(n_i)^{Road_i}<\omega<1$,则无论系统中其它任务复制次数 k_j 取何值($i\neq j$),系统的可靠性目标难以达到,也就是会造成 DAG 的可靠性值 $\prod_{n_i\in N}(NError'(n_i))^{Road_i}<\omega$.

证明. 已知条件为任务 n_i 的复制次数 $k_i<k_i^{\mathrm{down}}$,那么 $NError'(n_i)^{Road_i}<\omega$. 在此条件下,DAG 中其它任务的 $n_j\in N(i\neq j)$ 对应的 $NError'(n_j)^{Road_j}$ 之积 $\prod_{n_j\in N}(NError'(n_j))^{Road_j}$ 存在两种情况:

情况 1. 若 $\prod_{n_j\in N}(NError'(n_j))^{Road_j}<\omega$,在不等式两边同时乘以 $NError'(n_i)^{Road_i}$,可得 $GR<\omega\times NError'(n_i)^{Road_i}$,已知 $NError'(n_i)^{Road_i}<\omega$,则

$$GR<\omega\times NError'(n_i)^{Road_i}<\omega^2<\omega \quad (12)$$

即在第 1 种情况下 DAG 的可靠性 GR 小于 ω.

情况 2. 若 $\omega\leq\prod_{n_j\in N}(NError'(n_j))^{Road_j}$,在不等式两边同时乘以 $NError'(n_i)^{Road_i}$,可得 $\omega\times NError'(n_i)^{Road_i}\leq GR$. 已知 $GR=NError'(n_i)^{Road_i}\times\prod_{n_j\in N}(NError'(n_j))^{Road_j}$,则

$$\omega\times NError'(n_i)^{Road_i}\leq GR<NError'(n_i)^{Road_i}\times 1<\omega \quad (13)$$

综合以上两种情况可知,如果任务复制次数 k_i 小于下限值 k_i^{down}($k_i<k_i^{\mathrm{down}}$),都会造成 $\prod_{n_i\in N}(NError'(n_i))^{Road_i}<\omega$ 无法达到系统的可靠性目标. 证毕.

由定理 2 可知复制次数下限值 k_i^{down} 取 $NError'(n_i)^{Road_i}\geq\omega$ 满足的最小值,可推出任务 n_i 的可靠性 $NError'(n_i)\geq\sqrt[Road_i]{\omega}$,即

$$1-\prod_{l_i=0,\varphi_l\in\varphi}^{k_i}(1-NError(n_i^{l_i},\varphi_l))\geq\sqrt[Road_i]{\omega} \quad (14)$$

(2)任务复制次数下限值算法

接下来根据此表达式来设计获取任务复制次数下限值 k_i^{down} 的算法,使得不等式(14)成立. 该算法需引入几种优先级队列和集合,依次为

① 任务无故障概率优先级队 $NError\_Quque$,每个任务都包含一个此队列. 该队列是任务 n_i 在处理器 p_u 上无故障概率值 $NError(n_i,p_u)=\mathrm{e}^{-\lambda_i w_{i,u}}$ 的降序排列;

② $Duplication\_Set$ 集合$=\{k_1,k_2,\cdots,k_i\}$,该集合包含每个任务的副版复制个数,初始默认次数都为 0;

③ 集合 $Road\_Set=\{road_1,road_2,\cdots,road_i\}$,该集合包含每个任务到出口任务 n_{exit} 的路径个数;

④ 任务的可靠性贡献因式优先级队列 $Contribution\_Quque$,该队列是任务 n_i 对 DAG 可靠性贡献因式 $NError'(n_i)^{Road_i}$ 的降序排列.

公理 1. 处理器分配原则. 为了使得任务复制次数最小化,需从任务的无故障概率优先级队列 $NError\_Quque$ 中选择最大 $\mathrm{e}^{-\lambda_i w_{iu}}$ 且未分配任务 n_i 主版及其副版的处理器给当前任务版本.

算法 1. DAG 任务复制下限值算法.

输入:$Duplication\_Set$, $Contribution\_Quque$

输出:$Duplication\_Set$, $Contribution\_Quque$

for(var $i=0$; $i<|N|$; $i++$){ //遍历所有任务

初始任务 n_i 的复制个数 $k_i^{\mathrm{down}}=0$;

获取任务 n_i 发生故障的概率

$$fail=\prod_{l_i=0,p_u\in P}^{k_{\mathrm{down}}}(1-NError(n_i^{l_i},p_u));$$

while $(1-fail<\sqrt[Road_i]{\omega})\&(k_i<|U|)$ do

为 n_i 增加一个副版,$k_i^{\mathrm{down}}=k_i^{\mathrm{down}}+1$;

依公理 1 分配处理器;

更新 $fail=\prod_{l_i=0,p_u\in P}^{k_{\mathrm{down}}}(1-NError(n_i^{l_i},p_u))$;

end while

更新 n_i 对 DAG 的可靠性的贡献因式 $NError'(n_i)^{Road_i}=(1-fail)^{Road_i}$,并更新到 $Contribution\_Quque$ 中;

更新 n_i 的复制次数 k_i^{down},更新到 $Duplication\_Set$ 中;

}//end for

return $Duplication\_Set$,$Contribution\_Quque$.

任务复制次数下限值算法的时间复杂度为 $O(n\times p^2)$. 其中，n 为任务数，p 为处理器数.

证明.　遍历完一个 DAG 内的所有任务的时间复杂度为 $O(n)$，由于任务复制次数的大小不能超过处理器的数目 p，所以任务复制的时间复杂度为 $O(p)$，计算 $fail$ 值需要遍历处理器，所以其复杂度为 $O(p)$. 那么，任务复制次数下限值算法的时间复杂度为 $O(n\times p^2)$.　　证毕.

4.2　贪婪的任务复制策略算法

定义 2.　可靠性代价(Reliability Cost)[17]. 它是指因任务复制冗余执行而消耗的计算资源，DAG 任务每复制一次都会产生一定的可靠性代价. 设 $\boldsymbol{X}$ 是一个 $|N|\times|P|$ 的矩阵，$x_{i,u}$ 表示任务 n_i 是否存在主版任务或者副版任务分配在处理器 p_u 上运行. 若存在任务在处理器 p_u 上调度，则 $x_{i,u}=1$；否则 $x_{i,u}=0$. 在对任务分配处理器时，要求任务的主版任务和所有的副版任务被分配到不同的处理器上执行以保证当主版执行失败的时候，副版仍能在截止期前产生正确的结果. $w_{i,u}$ 是任务 n_i 在处理器 p_u 上的计算时间开销. 那么在一个 DAG 图中因为任务副版的执行而增加的可靠性代价表示为

$$GRC_{\mathrm{DAG}}=\sum_{i=0}^{|N|-1}\sum_{u=0}^{|U|-1}w_{i,u}\times x_{i,u} \tag{15}$$

因此，针对文献[10-12]中存在的高可靠性代价问题，即为容忍 ε 个故障盲目地使每个任务拥有 $\varepsilon+1$ 个副版，在结合以上知识的基础上，本文接下来提出相应策略的任务复制算法，算法以满足系统可靠性目标和降低系统可靠性代价为目的，在系统可靠性目标的驱使下动态地量化各个任务需要冗余复制的次数.

定义 3.　可靠性贡献值. 假设任务 n_i 当前拥有的副版数为 k_i，那么任务 n_i 可靠性贡献值 ΔGR_{n_i} 表示任务 n_i 复制一个副版之后(其副版数变为 $k_i^{\mathrm{new}}=k_i+1$)其 DAG 的可靠性值 GR_{DAG}^i 减去复制之前 DAG 的可靠性值 GR_{DAG}，即

$$\Delta GR_{n_i}=(GR_{\mathrm{DAG}}^i-GR_{\mathrm{DAG}}) \tag{16}$$

其中 GR_{DAG}^i 的计算公式为

$$GR_{\mathrm{DAG}}^i=\prod_{n_i\in N}(NError'(n_i))^{Road_i}$$

$$=\prod_{n_i\in N}\Big(1-\prod_{k_i=0,p_u\in P}^{k_i^{\mathrm{new}}}(1-NError(n_i^{k_i},p_u))\Big)^{Road_i} \tag{17}$$

贪婪(Greedy)的任务复制策略(Greedy Policy)是指对所有任务进行一次预复制，并从计算出的任务可靠性贡献值集合 $\{\Delta GR_{n_1},\Delta GR_{n_2},\cdots,\Delta GR_{n_{|N|}}\}$ 中找到当前贡献值最大的那个任务 n_i，并为其增加一个主动方式的副版任务.

算法 2.　贪婪的任务复制策略算法.

输入：*Duplication_Set*，*Contribution_Quque*

输出：*Duplication_Set*，*Contribution_Quque*

初始化：通过算法 1 获取 *Duplication_Set*，*Contribution_Quque*，其中 *Duplication_Set* 存储每个任务的最小复制次数 k_i，*Contribution_Quque* 存储每个任务的可靠性贡献因式 $NError'(n_i)^{Road_i}$；

通过式(11)计算 DAG 的初始可靠性值 GR_{DAG}；

while $GR_{\mathrm{DAG}}<\omega$ do

对每个任务预复制一次(已达到满复制的任务不再复制)，并通过式(16)和公理 1 得出每个任务的可靠性贡献值 ΔGR_{n_i} 和相应处理器，并组成可靠性贡献值集合 $\{\Delta GR_{n_1},\Delta GR_{n_2},\cdots,\Delta GR_{n_{|N|}}\}$；

从可靠性贡献值集合 $\{\Delta GR_{n_1},\Delta GR_{n_2},\cdots,\Delta GR_{n_{|N|}}\}$ 中选择最大值 n_i；

并将 GR_{DAG}^i 作为新的 DAG 可靠性值，即 $GR\leftarrow GR_{\mathrm{DAG}}^i$；

为 n_i 增加一个副版，$k_i=k_i+1$；

依处理器分配原则，分配处理器；

更新 n_i 对 DAG 的可靠性的贡献因式 $NError'(n_i)^{Road_i}$，并更新到 *Contribution_Quque* 中；

更新 n_i 的复制次数 k_i，并更新到 *Duplication_Set* 中；

end while

贪婪任务复制策略的时间复杂度为 $O(n^2\times p^2)$. 其中，n 为任务数，p 为处理器数.

证明.　遍历完一个 DAG 内的所有任务的时间复杂度为 $O(n)$. 计算 GR_{DAG}^i 值首先需要选择预复制任务的处理器，复杂度为 $O(p)$；其次要计算预复制任务的可靠性贡献值，需要遍历处理器，复杂度为 $O(p)$；最后得出 GR_{DAG}^i 需要遍历的所有任务，其复杂度为 $O(n)$. 所以，贪婪的任务复制策略的时间复杂度为 $O(n^2\times p^2)$.　　证毕.

4.3　经济的任务复制策略算法

贪婪的任务复制策略基于可靠性贡献值来复制任务，没有考虑因任务复制而消耗的计算资源，因而可以进一步优化.

定义 4.　可靠性贡献经济比值. 假设任务 n_i 当前拥有的副版数为 k_i，那么任务 n_i 可靠性贡献经济比值 ΔGRE_{n_i} 表示任务 n_i 复制一个副版之后，其可靠性贡献值 ΔGR_{n_i} 与该副版任务消耗的计算资源 w_{i,p_s} 的比值(s 为选择的处理器编号)，即

$$\Delta GRE_{n_i}=\frac{\Delta GR_{n_i}}{w_{i,p_s}} \tag{18}$$

经济的(Economic)任务复制策略是指对所有任务进行一次预复制，并从计算出的任务可靠性贡献经

济比值集合$\{\Delta GRE_{n_1},\Delta GRE_{n_2},\cdots,\Delta GRE_{n_{|N|}}\}$中找到当前贡献值最大的那个任务$n_i$.作为选择的结果任务$n_i$将增加一个主动方式的副版任务.该策略不但考虑了不同的任务对系统的可靠性贡献大小不同外,还将任务因复制冗余而消耗的计算资源作为指标,因此相比贪婪的任务复制策略,其付出的可靠性代价更少.

算法 3. 经济的任务复制策略算法.

输入:*Duplication_Set*,*Contribution_Quque*

输出:*Duplication_Set*,*Contribution_Quque*

通过算法 1 获取 *Replication_Set*,*Contribution_Quque*,其中 *Replication_Set* 存储每个任务的最小复制次数k_i,*Contribution_Quque* 存储每个任务的可靠性贡献因式$NError'(n_i)^{Road_i}$;

通过式(11)计算得出 DAG 的初始可靠性值GR_{DAG};

while $GR_{\mathrm{DAG}}<\omega$ then

对每个任务预复制一次(已达到满复制的任务不再复制),并通过式(18)和公理 1 计算得出每个任务的可靠性经济贡献值集合$\{\Delta GRE_{n_1},\Delta GRE_{n_2},\cdots,\Delta GRE_{n_{|N|}}\}$和相应处理器;

从可靠性经济贡献值集合中$\{\Delta GRE_{n_1},\Delta GRE_{n_2},\cdots,\Delta GRE_{n_{|N|}}\}$中选择最大值$n_i$,并将$GR^i_{\mathrm{DAG}}$作为新的 DAG 可靠性值,即$GR\leftarrow GR^i_{\mathrm{DAG}}$;

为n_i增加一个副版,$k_i=k_i+1$;

依处理器分配原则,分配处理器;

更新n_i对 DAG 的可靠性的贡献因式$NError'(n_i)^{Road_i}$,并更新到 *Contribution_Quque* 中;

更新n_i的复制次数k_i,并更新到 *Duplication_Set* 中;

end while

经济的任务复制策略的时间复杂度为$O(n^2\times p^2)$.

证明. 遍历完一个 DAG 内的所有任务的时间复杂度为$O(n)$.计算ΔGRE_{n_1}值首先需要选择预复制任务的处理器,复杂度为$O(p)$;其次要计算预复制任务的可靠性贡献值,需要遍历处理器,复杂度为$O(p)$;最后得出ΔGRE_{n_1}需要遍历的所有任务,其复杂度为$O(n)$.所以,经济的任务复制策略的时间复杂度为$O(n^2\times p^2)$. 证毕.

4.4 DAG 容错算法

通过理论分析,得出 DAG 的可靠性概率模型;明确为了达到 DAG 的可靠性目标,任务的复制次数需满足$k_i\geq k_i^{\mathrm{down}}$的限制,且通过任务复制次数下限值算法可获知下限值$k_i^{\mathrm{down}}$.针对文献[10-12]中盲目任务复制策略造成的高可靠性代价问题,提出了贪婪的任务复制策略算法和经济的任务复制策略算法.基于上述模型和算法,提出可选策略的 DAG 容错(Optional Policy on DAG Fault-Tolerant,OPDFT)算法.虽然通过任务复制策略得出了任务的主副版本和相应分配的处理器,但必须对任务的优先级进行排序,本文采用在 DAG 任务容错调度中常用的任务向上排序值(upward rank value)[10-11]的降序排列作为任务优先级排序标准,其计算公式为

$$\begin{cases} rank_u(n_{\mathrm{exit}})=\overline{w_{\mathrm{exit}}} \\ rank_u(n_i)=\overline{w_i}+\max\limits_{n_j\in succ(n_i)}\{\overline{c_{i,j}}+rank_u(n_j)\} \end{cases} \tag{19}$$

其中$\overline{w_i}$表示任务n_i的平均计算开销,OPDFT 算法流程如算法 4 和图 2 所示.

算法 4. OPDFT 算法.

输入:DAG 参数信息

输出:DAG 调度结果

1. 读取 DAG 的相关信息,如任务结点、任务结点之间的有向边和通信开销以及在处理单元上运行的计算开销等,确定系统参数集;

2. 按照深度优先遍历算法(Depth First Search,DFS)[16],求取各 DAG 中各任务到出口任务的路径个数$Road_i$;

3. 按照式(2)计算 DAG 中各任务在各个处理器上,在其执行时间内无瞬时性故障的概率 *Nerror*,并放入降序的优先级队列 *NError_Quque*;

4. 调用任务复制次数下限值算法;

5. 判断当前系统的可靠性值是否已满足系统可靠性目标,如果满足,则跳转至步 8;如果不满足,则进入下一步.

6. 若系统考虑计算资源消耗的经济性,则采用经济的任务复制选择策略,继续选择任务进行复制,否则采用贪婪的任务复制策略进行复制;

7. 获取最终各任务的复制和所分配的处理器;

8. 基于向上排序值对任务优先级排序并调度;

9. 返回调度结果,算法结束.

OPDFT 算法的时间复杂度主要考虑任务复制下限值算法和任务复制策略算法,如果通过任务复制次数下限值算法能够满足 DAG 的可靠性目标,那么无需调用任务复制策略算法,OPDFT 算法的时间复杂度就为$O(n\times p^2)$.否则需调用任务复制策略算法,使 OPDFT 算法的时间复杂度达到$O(n^2\times p^2)$.因此,这也说明了与直接调用相关任务复制策略算法相比,预先调用任务复制下限值算法,确定每个任务的复制次数下限值,能够为 OPDFT 算法产生更好的效率.

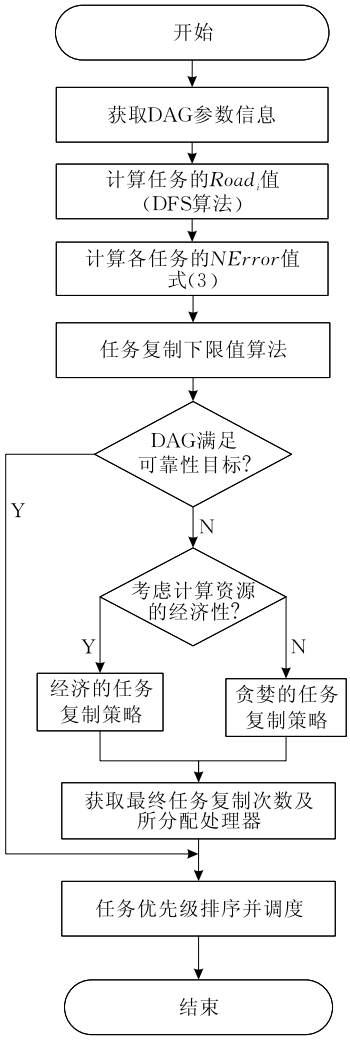


图 2　OPDFT 算法流程

5　实　验

5.1　实验平台及样本

实验的硬件环境为一台具有奔腾双核处理器(3.2 GHz/2.0 GB RAM)的 Windows XP 机器，所使用的软件工具有 VC++6.0 和 Highcharts，DAG 任务图生成工具 TGFF 3.5 (Task Graphs for Free)[18]. 实验流程步骤分为系统参数生成模块、算法处理模块、实验数据分析这 3 步.

本文将采用 TGFF 随机生成具有不同特征的大量随机 DAG 作为仿真测试中的任务样本空间，并在一个由 15 个异构处理器构成的处理器平台上进行任务分配，任务故障概率取值范围为 $(2\sim10)\times10^{-3}$ 中的随机数. 其中，TGFF 中生成 DAG 随机 DAG 的参数设置为：任务个数 $m=\{10,20,30,40,50,60,70,80,90,100,110,120,130,140,150\}$，DAG 最大出度集 $\beta=\{2,3,4,5\}$，DAG 最大入度集 $\gamma=\{2,3,4,5\}$，任务在不同处理器上执行时间的差异度 $\eta=\{0.1,0.5,1.0\}$，平均通信时间与平均计算时间的比值 $CCR=\{0.1,0.5,1.0,5.0,10.0\}$，假设 w_i 表示任务 n_i 的平均计算开销，那么 n_i 在处理器 p_k 上的计算开销可以通过公式产生，即

$$w_i\times(1-\eta/2)\leq w_{i,k}\leq w_i\times(1+\eta/2) \quad (20)$$

5.2　评价指标

本文采用 DAG 的可靠性(Reliability)、可靠性代价(Reliability Cost)和加速比(Speedup)作为评价指标.

可靠性即 DAG 的可靠性概率模型，采用式(7)计算，通过可靠性与可靠性目标之间的关系来衡量容错算法的有效性.

可靠性代价是指因任务复制冗余执行而消耗的计算资源，采用式(15)计算. DAG 的可靠性代价越低，说明冗余任务越少，容错算法越高效.

加速比即在一个处理器上串行执行 DAG 图中的所有任务时使用的最少时间与算法实际调度长度的比值. 调度算法产生的加速比越大，说明容错算法越高效，加速比计算公式为

$$Speedup=\frac{\min\limits_{p_u\in P}\left\{\sum\limits_{n_i\in V}w_{i,u}\right\}}{makespan} \quad (21)$$

5.3　实验结果及分析

实验 1.　针对多个 DAG 样本，探究它在不同的可靠性目标 ω 和不同的任务复制策略条件下，系统最终所获取的实际可靠性值和为此而付出的可靠性代价. 本次实验从样本空间中随机选取 20 个 DAG，并设定系统的可靠性目标 ω 在 0.80～0.99 区间变化，以 0.01 递增. 取 100 次实验所得数据的平均值作为最终的仿真结果. 实验结果如图 3 和图 4 所示.

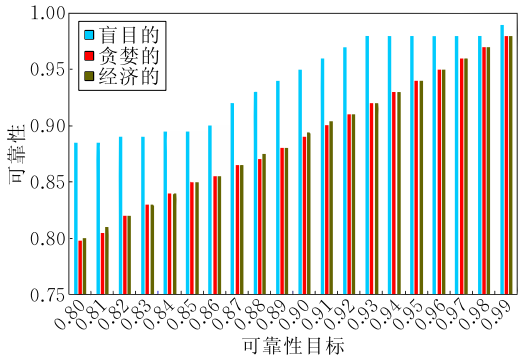


图 3　3 种复制策略的可靠性值随目标值的变化比较

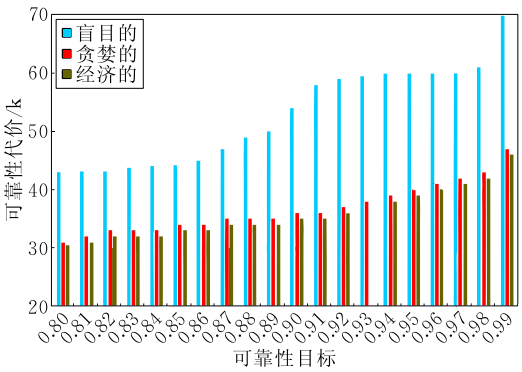


图 4 3 种复制策略的可靠性代价随可靠性目标的变化比较

图 3 表示利用 3 种任务复制策略进行容错设计之后最终所得的可靠性值与用户设定的可靠性目标之间的关系. 从图中可以看出,运用 3 种容错策略均能使系统可靠性目标得到满足. 其中,利用盲目的复制策略最终得到的可靠性值最大,这是因为较之另外两种策略,它缺乏有效的分析而盲目地运用了大量的冗余任务副版. 另外,经济的任务复制策略与贪婪的任务复制策略得到的结果较为接近. 但随着系统可靠性目标的不断提高,运用 3 种策略所得到的可靠性值之间差距逐渐缩小. 也就是说,当对系统可靠性要求较高时,盲目复制策略在最终得到可靠性值较大方面的优势在减弱.

图 4 表示 DAG 在不同的可靠性目标和任务复制策略下,为满足可靠性目标而付出的可靠性代价. 从图中可以看出,随着系统可靠性目标的变大,3 种复制策略所付出的可靠性代价都在增长,这是因为需增加任务副版数目以满足可靠性需求的增长. 但是在同一个可靠性目标下,本文所提出的经济的任务复制策略所付出的可靠性代价最低,贪婪的复制策略次之,盲目的任务复制策略的可靠性代价最大.

另外,对照图 3 与图 4,可以看出当系统可靠性目标值较大时,特别是当 $\omega=0.99$ 时,3 种复制策略最终所得到的可靠性值均满足目标且相差无几,但是盲目的任务复制策略付出了最大的可靠性代价,而本文所提出的经济的任务复制策略可靠性代价最低. 这是因为它是在对系统可靠性建模的基础上,兼顾不同的任务对系统可靠性的贡献与所带来的可靠性开销之间的关系,然后做出任务复制选择. 可见本文所提出的算法较之以往算法能够在一定程度上降低可靠性代价.

实验 2. 假定用户对系统的可靠性目标 $\omega=0.99$,评估任务数量规模不同的 DAG 在相同的可靠性目标条件下,分别采用以上 3 种任务复制策略所付出的可靠性代价. 本次实验中系统规模,即 DAG 任数在 10～150 之间变化. 实验结果如图 5 所示.

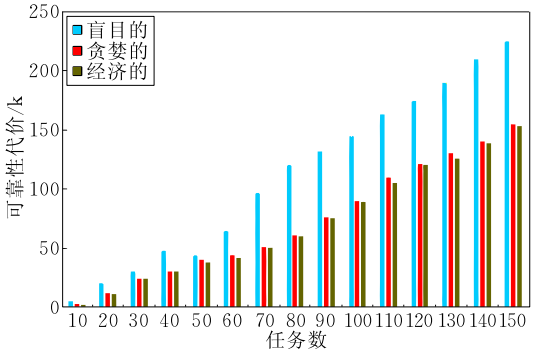


图 5 3 种复制策略的可靠性代价随 DAG 任务数的变化比较

由图 5 可看出当 DAG 任务规模增大时,可靠性代价值也在增大. 这是由于系统中含有的任务数量规模增加,会导致任务副版集合的增大,故而需要更多的计算资源来运行这些任务. 但从图中可以看出,无论 DAG 规模为多少,本文所提出的基于经济任务复制策略的容错算法对应的可靠性代价最少,大约是盲目的任务复制策略所带来的可靠性代价的 60%.

实验 3. 假定用户对系统的可靠性目标 $\omega=0.99$,评估任务数量规模不同的 DAG 在相同的可靠性目标条件下,分别采用以上 3 种任务复制策略所产生的的加速比. 本次实验中系统规模,即 DAG 任务数量在 10～150 之间变化. 实验结果如图 6 所示.

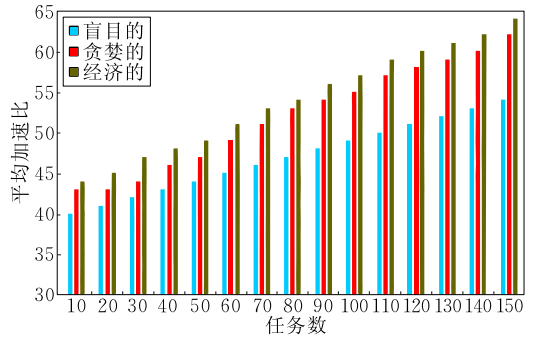


图 6 3 种复制策略的平均加速比随 DAG 任务数的变化比较

由图 6 可看出,当 DAG 任务规模增大时,加速比也在增大,本文所提出的基于经济任务复制策略的容错算法对应的加速比最大,贪婪复制策略次之,大约分别是盲目的任务复制策略所带来的加速比的 1.2 和 1.3 倍左右. 这是因为随着 DAG 任务规模增大,经济任务复制策略复制的副版任务比盲目复制策略要少,因此相比调度长度更短,从而加速比更高.

实验 4.　随机实验. 实验目的是对一个 DAG 分别运用基于以上 3 种复制策略的容错算法，获取它在可靠性目标 $\omega=0.99$ 的条件下，各个任务对应的复制次数以及所选择处理器情况. 本次实验从样本空间中随机选择 3 个 DAG，并以之为例验证算法功能的有效性.

本次随机实验中，随机从样本空间中选择了分别含有 8 个、5 个和 12 个任务的 DAG. 每个任务在不同的策略下对应的任务版本数如图 7、图 8 和图 9 所示. 这次试验中盲目的任务复制策略带来的可靠性代价为 4513，贪婪的任务复制策略对应的可靠性代价是 3171，经济的任务复制策略最少，可靠性代价是 3046. 各个 DAG 内采用经济任务策略情况下，任务所分配的处理器如表 4 所示.

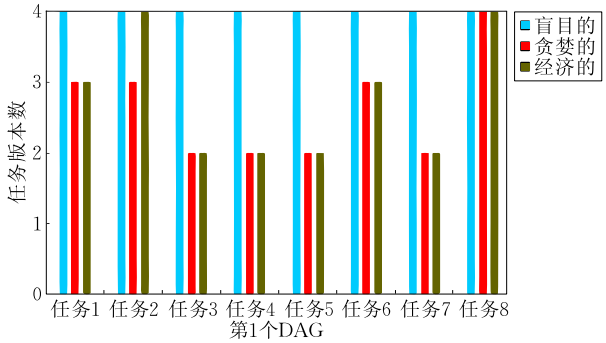


图 7　第 1 个随机 DAG 副版复制数比较

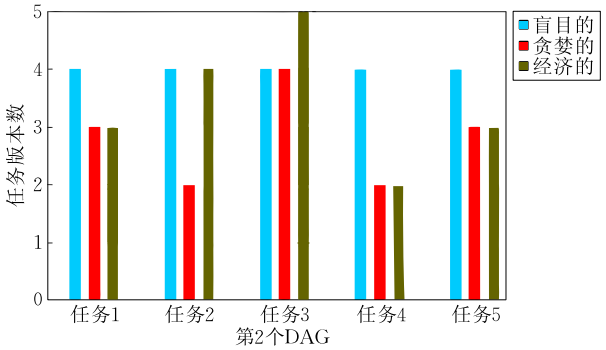


图 8　第 2 个随机 DAG 副版复制数比较

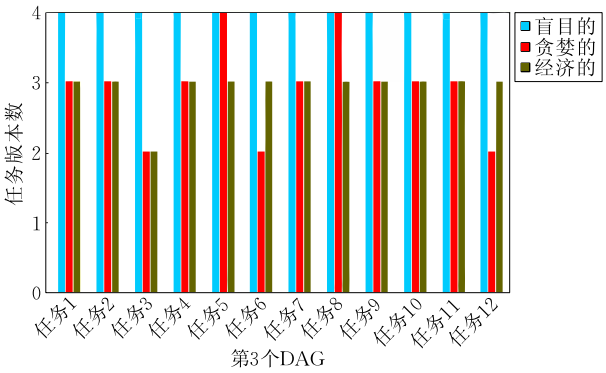


图 9　第 3 个随机 DAG 副版复制数比较

表 4　样本 DAG 处理器分配情况

任务序号	经济任务复制策略的处理器分配情况														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
G_{1_1}			√				√								√
G_{1_2}				√									√	√	√
G_{1_3}		√												√	
G_{1_4}		√		√	√		√								
G_{1_5}		√								√					
G_{1_6}					√		√			√					
G_{1_7}							√		√						
G_{1_8}			√				√	√						√	
G_{2_1}		√			√	√									
G_{2_2}							√				√	√			√
G_{2_3}		√	√			√	√						√		
G_{2_4}													√		√
G_{2_5}								√	√		√				
G_{3_1}				√	√										√
G_{3_2}			√	√										√	
G_{3_3}			√												√
G_{3_4}			√						√				√		
G_{3_5}				√				√		√					
G_{3_6}	√									√			√		
G_{3_7}											√		√		√
G_{3_8}			√									√			√
G_{3_9}			√				√				√				
G_{3_10}			√				√					√			
G_{3_11}			√							√		√			
G_{3_12}				√	√					√					

以上 4 次实验的结果表明，本文所提出的 DAG 任务启发式动态复制容错方法可以根据用户设定的可靠性目标有效地为任务动态地生成任务版本数，为之分配处理器；并且在达到可靠性目标的同时，经济任务复制策略和贪婪任务复制策略在可靠性代价方面分别是盲目策略算法的 60%和 70%左右.

实验 5.　与 MaxRe[13] 算法的 DAG 可靠性模型在可靠性及可靠性代价方面的比较. 图 10 为在基于经济复制策略的情况下，可靠性值随可靠性目标变化的趋势图. 可以看出，本文提出的 OPDFT 算法采用的 DAG 可靠性模型相比 MaxRe 在可靠性目标一定的情况下，具有更高的可靠性值.

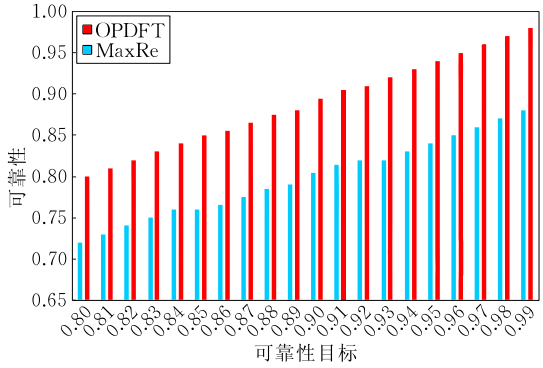


图 10　经济复制策略下实际可靠性值随目标值的变化比较

图 11 表示在经济复制策略下，DAG 在不同的可靠性目标下，为满足可靠性目标而付出的可靠性代价. 从图中可以看出，随着系统可靠性目标的增大，本文所提出的 DAG 可靠性模型所付出的可靠性代价在最坏情况也仅为 MaxRe 算法的 80% 左右.

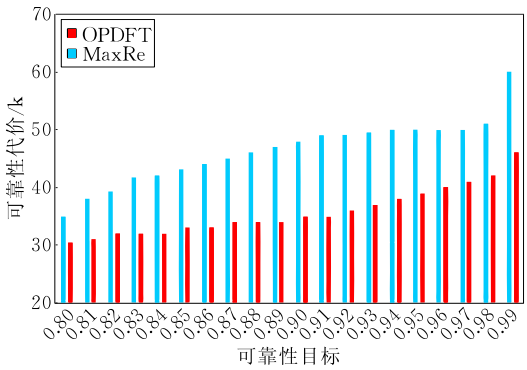


图 11 经济复制策略下可靠性代价随可靠性目标值的变化比较

由于本文从任务优先级约束和可靠性贡献因式出发，建立更加精确的可靠性模型，因而具有更好的系统可靠性值和更低的可靠性代价.

实验 6. 考虑真实汽车电子环境下，端到端最差响应时间(Worst-Case Response Time，WCRT)[19]随消息数变化的情况. 在汽车电子系统中，消息集的 WCRT 定义为消息集的第一个消息所分配的 ECU 触发就绪到传输完毕到达最后一个消息所在 ECU 节点之间的时间段，WCRT 越短，说明加速比越高，算法越高效.

本文采用日本名古屋大学嵌入式系统研究中心提供的单个 CAN 网络环境下的包括 65 个消息任务，并被分配到 14 个 ECU 之中的真实消息任务集[19]. 基于经济复制策略，实验结果如图 12 所示，从结果可以看出，OPDFT 的 WRCT 相比 MaxRe 要短，最好情况下优于 MaxRe 算法 30%以上.

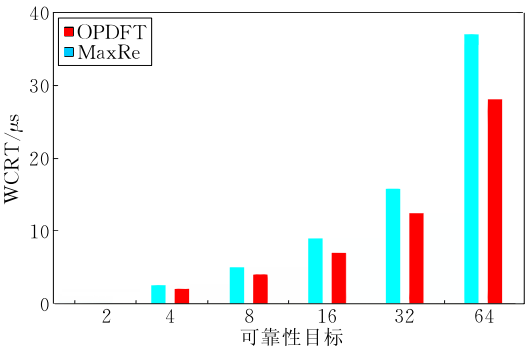


图 12 经济复制策略下 WCRT 随任务消息数的变化比较

6 总 结

本文面向异构分布式计算系统领域进行容错研究，以经典 DAG 任务模型为基础，在对 DAG 任务进行可靠性分析的基础上，提出了 DAG 的可靠性模型并进行了证明和实例验证；然后利用主动任务复制容错方式，给出了系统可靠性的约束条件以及可靠性代价的计算方法，并提出了一种计算 DAG 任务复制次数下限值算法；接着分别提出以降低可靠性代价为目标的贪婪的任务复制策略和经济的任务复制策略算法，在一定程度上降低了对任务的盲目复制；接着通过结合上述 3 个算法，给出了 DAG 可选策略的容错调度算法；最后利用仿真实验将本文提出的算法与盲目复制策略进行比较，结果表明本文所提出的算法可精确量化任务复制个数，在满足系统的可靠性目标的前提下付出更少的可靠性代价并具有更好的调度性能.

致 谢 在此，向湖南大学嵌入式与网络计算重点实验室李仁发教授领导的高性能嵌入式计算讨论班和国家超级计算长沙中心异构并行系统高性能研究组的专家、老师、工程师和同学所提出的意见和建议表示感谢，向本文的审稿专家致以深深的谢意！

参 考 文 献

[1] Xie Yong, Li Ren-Fa, Ruan Hua-Bin, Peng Xin. Optimal configuration algorithm for static segment of FlexRay. Journal of Communications, 2012, 33(11): 33-40(in Chinese)
(谢勇，李仁发，阮华斌，彭鑫. 最优的 FlexRay 静态段配置算法. 通信学报，2012，33(11)：33-40)

[2] Jia Jia, Yang Xue-Jun. Propagation behavior analysis and fault tolerance optimization of hardware fault in heterogeneous systems. Journal of Software, 2011, 22(12): 2853-2865 (in Chinese)
(贾佳，杨学军. 异构系统硬件故障传播行为分析及容错优化. 软件学报，2011，22(12)：2853-2865)

[3] Topcuoglu H, Hariri S, Wu M. Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE Transactions on Parallel and Distributed Systems, 2002, 13(3): 260-274

[4] Luo Wei, Yang Fu-Min, Pang Li-Ping, Tu Gang. A real-time fault-tolerant scheduling algorithm of periodic tasks in heterogeneous distributed systems. Chinese Journal of Computers,

2007，10(10)：1740-1749(in Chinese)
(罗威，阳富民，庞丽萍，涂刚. 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，2007，10(10)：1740-1749)

[5] Qin X，Jiang H，Swanson D R. An efficient fault-tolerant scheduling algorithm for real-time tasks with precedence constraints in heterogeneous systems//Proceedings of the International Conference on Parallel Processing (ICPP'02). Vancouver，Canada，2002：360-368

[6] Qin X，Jiang H. A novel fault-tolerant scheduling algorithm for precedence constrained tasks in real-time heterogeneous systems. Parallel Computing，2006，32(5)：331-356

[7] Zheng Q，Veeravalli B，Tham C K. On the design of fault-tolerant scheduling strategies using primary-backup approach for computational grids with low replication costs. IEEE Transactions on Computers，2009，58(3)：380-393

[8] Lin J，Cheng A M K. Real-time task assignment with replication on multiprocessor platforms//Proceedings of the 2009 15th International Conference on Parallel and Distributed Systems (ICPADS'09). Shenzhen，China，2009：399-406

[9] Gopalakrishnan S，Caccamo M. Task partitioning with replication upon heterogeneous multiprocessor systems//Proceedings of the 12th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'06). San Jose，USA，2006：199-207

[10] Benoit A，Hakem M，Robert Y. Fault tolerant scheduling of precedence task graphs on heterogeneous platforms//Proceedings of the 22nd IEEE International Symposium on Parallel and Distributed Processing(IPDPS 2008). Miami，USA，2008：1-8

[11] Benoit A，Hakem M，Robert Y. Contention awareness and fault-tolerant scheduling for precedence constrained tasks in heterogeneous systems. Parallel Computing，2009，35(2)：83-108

[12] Tabbaa N，Entezari-Maleki R，Movaghar A. A fault tolerant scheduling algorithm for dDAG applications in cluster environments//Proceedings of the International Conference on Digital Information Processing and Communications (ICDIPC 2011). Ostrava，Czech，2011：189-199

[13] Zhao L，Ren Y，Xiang Y，et al. Fault-tolerant scheduling with dynamic number of replicas in heterogeneous systems//Proceedings of the 12th IEEE International Conference on High Performance Computing and Communications (HPCC 2010). Melbourne，Australia，2010：434-441

[14] Guo Hui，Wang Zhi-Guang，Zhou Jing-Li. Load balancing based process scheduling with fault-tolerance in heterogeneous distributed systems. Chinese Journal of Computers，2005，28(11)：1807-1816(in Chinese)
(郭辉，王智广，周敬利. 异构分布式系统中基于负载均衡的容错调度算法. 计算机学报，2005，28(11)：1807-1816)

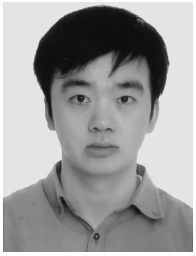
[15] Ilavarasan E，Thambidurai P. Low complexity performance effective task scheduling algorithm for heterogeneous computing environments. Journal of Computer Sciences，2007，3(2)：94-103

[16] Yan Wei-Min，Wu Wei-Min. Data Structures. 3rd Edition. Beijing：Tsinghua University Press，2007：167-170 (in Chinese)
(严蔚敏，吴伟民. 数据结构(C 语言版). 第 3 版. 北京：清华大学出版社，2007：167-170)

[17] Shatz S M，Wang J P，Goto M. Task allocation for maximizing reliability of distributed computer systems. IEEE Transactions on Computers，1992，41(9)：1156-1168

[18] Dick R P，Rhodes D L，Wolf W. TGFF：Task graphs for free//Proceedings of the 6th International Workshop on Hardware/Software Codesign (CODES/CASHE'98). Seattle，USA，1998：97-101

[19] Chen Y，Zeng G，Ryo K. Effects of queueing jitter on worst-case response times of CAN messages with offsets//Proceedings of the Embedded System Symposium in Japan. Tokyo，Japan，2012：119-126



XIE Guo-Qi，born in 1983，Ph. D. candidate. His main research interests include computer architecture and fault-tolerant computing.

LI Ren-Fa，born in 1957，Ph. D.，professor，Ph. D. supervisor. His main research interests include computer architecture，fault-tolerant computing and computer network.

LIU Lin，born in 1988，M. S. candidate. Her main research interests include computer architecture and fault-tolerant computing.

YANG Fan，born in 1985，Ph. D. candidate. His main research interests include computer architecture，fault-tolerant computing and embedded modeling.

Background

Computer systems are increasingly towards heterogeneous and distributed, the performance of heterogeneous distributed systems like automotive electronic systems and supercomputing systems has been improved significantly, but caused increased failures dramatically. Tolerant scheduling in heterogeneous distributed systems with DAG task model becomes a research focuses. Widely used fault-tolerant algorithms based on task replication have following problems: (1) there are some deficiencies and lack of rigorous proof on DAG task reliability model and DAG reliability model; (2) only one backup copy of each task, which no enough to cope with potential repeated failures; (3) blindly to tolerate ε faults of each task which has $\varepsilon+1$ backup copies, which improving the reliability of system, but caused high redundancy and resources consumption.

This paper is aimed to address above problems. Firstly, the authors analyze task dependencies of DAG to determine DAG task reliability probability model and based on this to construct DAG reliability model with rigorous proof and examples validation. Then give the definition of DAG reliability constraint and present lower limit of task duplication algorithm with low complexity to give the lowest replicas of each task. For precise quantification of replicas, economic task duplication strategy algorithm and greedy task replication strategy algorithm are presented to meet the reliability target of DAG. Lastly, present optional policy on DAG fault-tolerant algorithm based on above 3 algorithms to realize our goals.

The authors have done research on Domain-oriented SoC design methodology and application in high-performance embedded systems.

At present, with the support of National Natural Science Foundation of China under Grant Nos. 61133005, 61173036, 61070057, 61272061, 61202102 and the National High-Tech Research and Development Plan of China under Grant No. 2012AA01A301-01 and so on, the authors are doing relevant works on scheduling, fault-tolerant and programs in computer architecture and software.