高技术通讯 2012 年 第22 卷 第7 期:689 ~ 696

doi:10.3772/j.issn.1002-0470.2012.07.004

RCSM:多主体系统中的可靠关键子系统模型①

张冬蕾②[*,**] 史忠植[**] 王 鹏[*] 赵 亮[*]

([*]61226 部队 北京 100079)

([**]中国科学院计算技术研究所智能信息处理重点实验室 北京 100190)

摘 要 针对多主体系统由其自治的软件主体构成带来的故障处理上的困难,研究了多主体系统的可靠性问题,并尝试从主体组层面出发对系统中的故障进行协调处理,提出了一个可靠关键子系统模型(RCSM)。RCSM 包含三类主体,即中继主体、控制中心主体以及工作主体。此模型使用中继主体封装由工作主体组成的关键子系统,使子系统的拓扑结构对外界不可见,从而避免故障的大范围传播。在子系统内部采用基于任务分配的一致性检查点设置技术实现工作主体的故障恢复。对模型中的控制中心主体则采用了复制技术与团队协作相结合的手段来提高其可靠性。通过仿真实验对 RCSM 的故障恢复效率及其对系统正常运行效率的影响进行分析,说明了该模型的实用价值。

关键词 多主体系统(MAS),可靠性模型,关键子系统,检查点,复制技术

0 引 言

近 50 年来,人们对主体(agent)、多主体(multi-agent)进行了大量的研究,特别是从 20 世纪 90 年代以来逐渐形成了主体技术研究的热潮。主体技术与多主体系统(multi-agent system, MAS)已成为人工智能甚至计算机科学的研究热点。多主体系统由于其高度的模块化、自主性、智能性,使其非常适合运行在一个开放的分布式的环境中,属于不同组织的主体间可以通过网络交互合作来完成更为复杂的任务。但是多主体系统作为一个复杂的分布式系统,它同样面临着所有分布式软件系统会面临的故障问题,如主体软件实现中没有恰当处理异常情况,支持主体运行的环境、计算机的崩溃,网络链接中断等软硬件故障等,这些故障都可能导致多主体系统的瘫痪,如何应对这些故障是多主体系统面临的严峻挑战。由于多主体系统由具有社会属性的有自治能力的软件主体构成,这使得多主体系统中的故障处理要比传统的分布式系统中的故障处理更为困难,当系统中某点发生故障后,由于主体间的频繁交互,使得故障更容易扩散,其影响范围难以得到有效的控制。为了提高多主体系统的可靠性,需要使用有效手段来应对各种可能发生的故障。为此,本文提出了一个多主体系统可靠性模型——可靠关键子系统模型(reliable critical subsystem model,RCSM),并对其功能进行了较详尽的论述。

1 研究现状

目前在多主体系统可靠性领域,已经进行了大量研究工作,归纳起来可分为基于故障预防和基于故障恢复两大类。实践证明,利用故障预防技术来提高系统的可靠性,一般最多使系统平均无故障时间增加一个数量级,要想进一步提高系统可靠性就必须采用故障恢复技术。

基于故障恢复的技术需要引入冗余来实现,因此可以根据冗余引入的不同划分为以下两类[1]:

(1) 基于复制(replication)的技术:多个复制体同时在系统中运行,其中一个作为当前活跃主体响应外界的交互请求,在其崩溃时,从其余复制体中挑选出新的活跃主体继续之前的工作。

(2) 基于检查点(checkpoint)的技术:主体在运行过程中,将执行状态保存在稳定的存储空间中,当

① 973 计划(2007CB311004)和国家自然科学基金(61035003,60970088)资助项目。

② 女,1979 年生,博士;研究方向:智能主体技术和多主体系统;联系人,E-mail: dongleizhang@gmail.com

(收稿日期:2011-04-06)

故障发生时,主体能够从记录检查点的位置恢复运行。

利用上述技术,研究者提出了一些多主体系统的可靠性模型。Hägg 将哨兵机制引入了多主体系统[2],为系统的每一部分安排一个哨兵主体,通过监控主体间的交互信息,建立这些主体对外部世界的认知模型,并将这些模型与哨兵主体预先保存的模型进行比对,不相符则认为有故障发生。哨兵起着管理系统中某部分功能的作用,它本身也是一个故障引入点,但在其模型中没有任何机制保证哨兵主体的可靠性,而且在故障恢复过程中哨兵需要与故障主体进行交互,这种要求在故障主体已经崩溃的情况下是不可能实现的。Fedoruk 等提出了一个 Proxy 模型[3],使用一个 Proxy 主体作为对外接口来管理一组完成相同功能的主体,在任意时刻组内只存在一个 Active 主体对外提供能力,当其出现故障时 Proxy 主体负责挑选新的 Active 主体继续工作。Taesoon[4] 和 Wu[5] 等将 Proxy 主体的概念进行了扩展,模型由一组完成相同工作的主体组成,这些主体中不存在 Proxy 这种起控制作用的主体,而是分为 Primary 和 Slave 两种主体。Primary 主体负责全部的计算工作,当其出现故障时,所有 Slave 主体投票从 Slave 主体中选出新的 Primary 主体继续工作。

在这些现有的工作中,多主体系统可靠性模型基本围绕单个主体提出,然而在大规模多主体系统中,通常以主体组(agent group)作为系统的基本构成单元,如子整体层次模型、联合模型、团队模型、联邦模型、混合模型等[6]。主体组划分出一个主体交互活动最为频繁的边界,也是主体发生故障时影响最密切的范围。仅孤立地提高系统中某个或某些主体的可靠性存在较大的局限性,而从主体组的层次出发,全局协调组内主体的故障恢复,对整个系统的可靠性提高有着更为显著的影响。

为了提高主体组的可靠性,本文从主体组内部的任务分配执行机制入手,建立了一个多主体系统可靠性模型——可靠关键子系统模型(RCSM)。在模型中使用封装思想来处理主体组与外部主体间的关系,避免故障在更大范围内传播。在主体组内部采用复制技术与检查点技术相结合的思想,通过任务的重新分配与执行,实现故障在主体组中的协调恢复,使故障的影响对外界主体透明化,可以有效地解决由主体崩溃、主机崩溃以及网络中断等异常情况所导致的故障。

2 多主体系统中的关键子系统

由于系统中的资源是有限的,系统设计者必须合理地利用有限的资源最大限度地提高多主体系统的可靠性,因此需要对多主体系统的组织结构进行分析,找出那些一旦发生故障所造成的影响范围最大的主体进行重点保护,而主体依赖图正是一个非常合适的分析工具。在多主体系统中,每个主体都被定义为具有自治能力的实体。但是,并不是所有主体都拥有执行任务所需要的全部能力或资源,这时该主体将寻求其他主体的支持,来完成自己的任务。为了描述这种主体间的相互依赖关系,研究者从社会关系学研究领域引入了依赖图这一分析工具[7-9]。我们对依赖图的定义进行了扩展,将系统中的一个主体组用图中的一个节点表示,节点间的有向边和标签代表主体组间的依赖关系与权值,使用类似的方法分析以主体组为基本单元的多主体系统。

定义 1 关系依赖图(relationship dependence graph,RDG) 将多主体系统中的每个主体组用一个节点表示,由这些节点组成的带有标签的有向图 G 即为主体间关系依赖图 RDG(N, L, W)。其中 $N = \{N_i\}_{i=1,\cdots,n}$ 为图中节点集合,$L = \{L_{ij}\}_{i=1,\cdots,n;j=1,\cdots,n}$ 为有向边集合,$W = \{W_{ij}\}_{i=1,\cdots,n;j=1,\cdots,n}$ 为有向边上的标签集合。L_{ij} 是从节点 N_i 到节点 N_j 的有向边,$i \neq j$,即没有从一个节点出发指向自身的边。W_{ij} 是有向边 L_{ij} 上的标签,它是一个取值范围在[0, 1]区间的实数,是表示主体组 Group_i 对主体组 Group_j 依赖程度的一个权值,如为 0 则表示两个主体组之间没有依赖关系,在依赖图中表现为没有边将两个节点连接起来。

通过分析一个多主体系统对应的关系依赖图 RDG(N, L, W),可以得出每个主体组 Group_i 在系统中的重要程度 W_i,计算公式为 $W_i = \sum_{j=1,j\neq i}^{n} W_{ji}$,其中 n 为依赖图中节点的数量。

由此可以将系统中的主体组按照 W_i 的大小从高到低进行排序,选择排位靠前的主体组进行保护。如果所有的 W_i 基本相同,这说明组间的依赖关系均匀地分布在整个系统中,保护其中任意主体组对系统可靠性的提高效果都是一样的。然而在实际应用中,很多系统中的依赖关系分布是不均匀的,一些主体组拥有系统中唯一或较少的资源或能力,并且该

资源或能力是其他主体组为了完成各自任务所必需的。这将导致某些主体组的 W_i 值远远高于系统中的其他组,在依赖图中表现为若干以这些主体组为核心的星形结构,将这类主体组称为系统中的关键主体组。相对整个多主体系统而言,一个主体组可以看成一个由主体构成的子系统,因此将关键主体组定义为关键子系统。

定义 2 关键子系统(critical subsystem) 多主体系统中的关键子系统 $CS = \{S_i\}$ 定义为满足下列条件的子系统 S_i 的集合:$W_i = \sum_{j=1,j\neq i}^{n} W_{ji} > \Lambda$,其中 W_i 表示子系统 S_i 作为一个整体对外界的重要程度,n 为依赖图中节点的数量,Λ 为预先设置的重要程度阈值。

对于关键子系统,可以根据其是否依赖系统中其他主体组分为强关键子系统和弱关键子系统。强关键子系统经常作为系统中的某种功能的提供者存在,其作用就是支持系统中其他模块的运行。显然,使用有限的资源来提高强关键子系统的可靠性能够最大程度地提高整个多主体系统的可靠性。为了表述方便,在后面的讨论中如无特殊说明,关键子系统即指强关键子系统。

3 可靠关键子系统模型(RCSM)

3.1 模型整体结构

图 1 所示为 RCSM 的结构图。由于外部主体在寻求关键子系统的支持时与子系统内部主体直接交互,当内部主体出现故障时,很可能将故障通过交互操作传播到外部主体中,这对故障的恢复十分不利。因此引入一个中继主体,借鉴面向对象软件设计中的封装思想,将整个关键子系统映射成为一个中继主体,使这个中继主体成为关键子系统对外的唯一接口。引入中继主体后,子系统中的主体将不再把各自的能力注册到系统的目录服务中,而是由中继主体在目录服务中注册子系统中所有主体的能力。通过这种方式,使得在其他任务请求主体看来,关键子系统成为一个拥有强大能力的单个主体,所有的交互请求都通过中继主体进行处理。中继主体将子系统中的能力实际提供主体以及子系统内部的拓扑结构对外部主体屏蔽,避免故障由于外部主体与内部主体间的直接交互而传播出去,扩散到无法控制的程度。

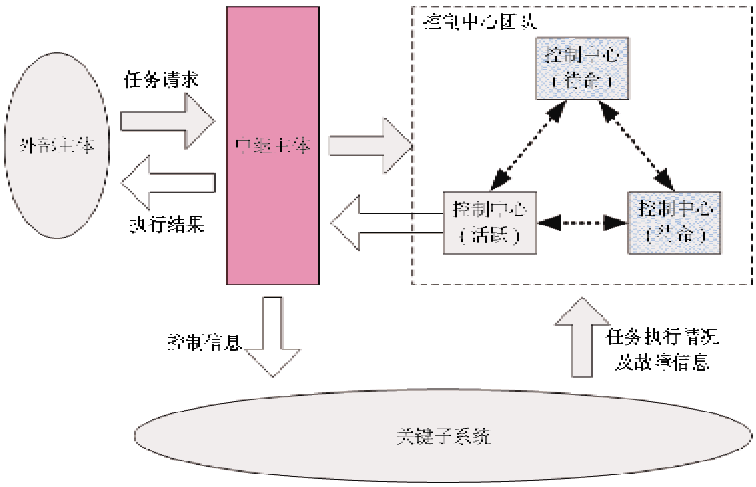

图 1 可靠关键子系统模型 RCSM 结构图

由于中继主体是关键子系统与系统中其他主体间的联系桥梁,中继主体如果崩溃,则子系统与外界的联系将完全中断,因此保证中继主体的可靠性是模型要解决的首要问题。软件的可靠性与其复杂程度成反比,越复杂的软件其崩溃的概率越高。从这个基本认识出发,为了将中继主体设计得尽可能简单,在模型中引入另外一个功能主体——控制中心主体(简称控制中心),它将负责关键子系统的管理工作,而中继主体仅仅负责消息的转发以及接受控制中心主体的注册。中继主体负责转发的消息包括两类,即外部主体与控制中心间的消息以及控制中心发往关键子系统的消息。

由于中继主体的功能已经非常简单,其可靠性可以得到极大的保证,这种简单的转发及控制中心

的注册功能甚至可以用专门的硬件来完成,以进一步降低崩溃的可能性。并且由于中继主体的无记忆性运行,即它并不依赖之前的运行结果,仅需要将收到的消息转发出去,即使它崩溃了,恢复时也无需考虑数据的一致性问题,能够快速简单地投入运行。这一切可以保证模型在系统中拥有一个较为稳固的连接点。至此,将可靠性模型的基本思想扩展为可靠关键子系统模型(RCSM)。

定义 3 可靠关键子系统模型(RCSM) 可靠关键子系统模型是一个五元组,RCSM = (RA, {CCT}, {WA}, $\{R_{cw}\}$, $\{R_{co}\}$),其中 RA 为中继主体,{CCT}为控制中心主体集合,{WA}为构成关键子系统的工作主体集合。$\{R_{cw}\}$为控制中心与工作主体之间的控制与通信关系集合,$\{R_{co}\}$为控制中心与模型外部主体之间的通信关系集合。

在 RCSM 中包含三类主体:负责消息转发的中继主体(relay agent);负责管理关键子系统的控制中心主体(control center agent);关键子系统中负责实际执行任务的主体,模型中将其定义为工作主体(worker agent)。这些主体间的关系以及消息传递的方向如图 1 所示。

中继主体的可靠性如何得到保证已在本节做了说明,下面讨论如何提高控制中心主体和工作主体的可靠性,实现提高关键子系统可靠性的目标,并通过这些讨论来说明可靠关键子系统模型的运转机制。

3.2 RCSM 中工作主体可靠性

关键子系统中的工作主体作为任务的真正执行者,其结构一般较为复杂,需要较强的计算能力,而且在某些情况中一个任务的执行将化费较长的时间,例如科学计算等。此外子系统中的主体之间经常为了协作完成某项任务而进行大量的交互通讯。这些特点都限制了复制技术在其上大量使用的可行性,因此在本节中介绍如何使用检查点技术来提高工作主体的可靠性。

检查点技术需要配合回卷恢复策略来消除崩溃主体对系统的影响。回卷恢复策略用于消除崩溃主体在检查点到崩溃点之间对其他主体造成的影响,使所有相关主体都回退到某个一致状态,以保证崩溃主体能够使用之前保存的检查点信息正确地恢复运行。为了实现回卷策略,在主体的正常运行过程中,需要将影响到其他主体的操作以日志形式保存到稳定存储空间中,在主体发生崩溃时使用这些信息进行回卷,如 Strasser 使用的 Message Logging[10]。主体在正常运行过程中,记录同步的日志会导致大量额外开销,降低其运行效率。在关键子系统中各主体联系紧密,交互频繁,进一步加剧了这种日志开销,因此这种检查点/日志相结合的技术不适合在关键子系统中应用。

为了避免使用日志文件进行低效率的回卷操作,我们提出了一种基于任务分配的一致性检查点设置方案。该方案可以保证不同工作主体间检查点的一致性,当某个主体出现故障而崩溃时,能够直接使用对应的检查点重新运行之前崩溃的任务,无需进行回卷操作,在很大程度上降低了故障恢复的复杂性。对于如何记录一个主体的检查点信息目前已有大量相关文献进行介绍[11-13],由于检查点信息的记录与主体的具体实现相关,所以在本文并不讨论检查点的记录方式。为了能够使用一致性检查点设置方案,首先对关键子系统的属性进行一些假设:

(1)子系统中的资源分布在不同的主体中,不同主体控制的资源不可重叠,其他主体想要访问该资源只能通过控制资源的主体,且主体对所控制资源拥有完全的控制权;

(2)外部主体仅在任务的协商过程中与控制中心有交互操作,一旦任务提交给控制中心,任务执行过程将由控制中心控制,外部主体仅等待任务执行结果,与控制中心之间不再有交互操作;

(3)工作主体的自适应能力可以保证执行任务过程中与其他主体交互操作的正确性,运行在不同主体上的有交互操作的任务,其开始执行的前后顺序并不会对结果造成影响;

(4)工作主体在没有执行任务时,处于一个稳定的可以被记录的状态。

设某个时刻关键子系统接收到的任务为图 2 左侧所示的任务树,假设关键子系统由 3 个工作主体即 Agent1 – Agent3 组成,则该任务树在子系统中可能的任务分配和执行过程如图 2 右侧所示。

假设 Agent2 在执行任务 T3 的过程中由于某种异常而导致崩溃,由于 Agent1 所运行的任务 T2 与任务 T3 有交互关系,因此 Agent1 也可能由于 Agent2的崩溃而崩溃或者陷入无限等待中,造成任务 T1 的执行前提条件无法满足,最终导致整个任务树执行失败。对于这种情况的简单处理办法是重新启动这 3 个主体,将该组任务分配给它们重新执行。然而当任务的执行需要较长时间时,这种完全从头

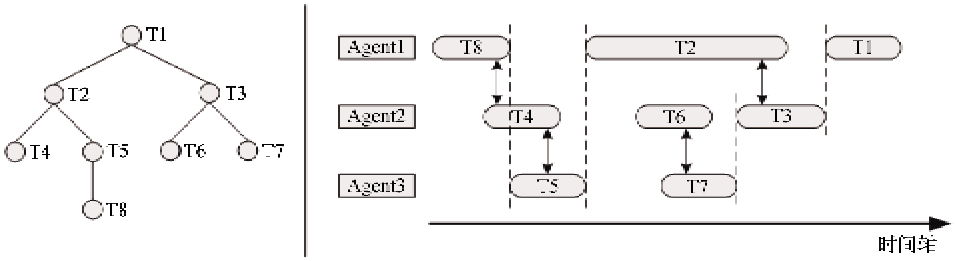

图2　到达子系统的任务树及执行过程示例

开始执行的处理方式十分低效,并且如果系统运行在一个不稳定的环境中,即工作主体执行任务过程中崩溃概率较大时,这组任务有可能永远也无法执行完成。对于这种情况,如果能够部分地利用之前任务的运行结果,使故障主体从一个中间结果的基础上恢复运行,将会大大提高运行效率。

为了实现这一目标,在RCSM中,工作主体在执行两个任务之间保存一个检查点,其中包含了主体自身的状态以及主体所控制的资源的状态。由于主体在运行过程中记录了多个检查点,在崩溃后需要挑选出满足一致性要求的最近的检查点。如果主体发生崩溃时所运行的任务与其他主体没有交互操作,说明这次崩溃的影响仅局限在该主体内部,则选择该任务执行前所记录的检查点就可以满足要求,否则就需要其他交互主体同时读取检查点来消除崩溃主体对它们造成的影响。下面的两个算法用于为所有受故障影响的主体选择最近的满足一致性要求的检查点进行故障恢复。

算法1　故障影响范围检查算法:

(1)设需要重新执行的任务集合为S,发生崩溃的任务为$Task_0$,将S的初始值设为$S=\{Task_0\}$,将$Task_0$标记为未处理;

(2)如果S中存在一个没有处理过的任务$Task_i$,对于任何任务$Task_j$,若其与任务$Task_i$有交互关系,或任务$Task_j$在任务$Task_i$之后在同一个主体上运行,并且$Task_j$不在集合S中,则将任务$Task_j$加入集合S,并标记为未处理,最后将任务$Task_i$标记为已处理;

(3)如果集合S中所有任务都标记为已处理,则算法结束,返回S;否则重复执行步骤(2)。

算法2　一致性检查点选择及任务重新分配算法:

(1)假设受故障影响的工作主体共有N个,为$AG_1,AG_2,\cdots,AG_N$,将受影响任务集合S中的任务以主体为单位划分为N个任务子集:$S_1,S_2,\cdots,S_N$;

(2)将这N个集合中的任务按照执行的先后顺序进行排序;

(3)对于每个任务集合S_i,使用其中最早运行的任务对应的检查点信息来设置主体AG_i的状态以及对应资源的状态,同时将任务集合S_i分配给主体AG_i;

(4)当所有主体及资源的状态都设置完毕,对应任务集合也分配完毕,重新运行全部受故障影响主体。故障恢复过程结束。

在RCSM中,控制中心主体结合任务树信息,使用上面的两个算法来进行关键子系统中工作主体的故障恢复。

3.3　RCSM中控制中心的可靠性

控制中心主体的功能包括如下两点:(1)接收从外部主体发来的任务请求,分解任务请求并将子任务分配给关键子系统中的主体,协调各工作主体执行任务,将结果汇总发往外部主体;(2)在工作主体发生故障时启动故障恢复机制消除故障影响,使关键子系统恢复到正常运行状态。

为了提高控制中心主体的可靠性,在RCSM中使用了复制技术,即同时运行控制中心主体的多个副本,其中一个副本为活跃控制中心,其余为待命控制中心,可以在活跃控制中心出现故障时接替其工作。为了监控活跃控制中心的状态,活跃控制中心需要定期发送消息给中继主体,以表明其运行状态正常。当中继主体在规定时间内没有收到该消息则认定活跃控制中心发生故障,中继主体将指定新的活跃控制中心。

使用复制技术需要解决的关键问题是如何保证信息在多个主体副本间的一致性。为此使用团队合作(teamwork)思想[14]将所有控制中心副本组成一个团队,来应对各种导致信息不一致的情况。

使用联合意图理论定义的控制中心团队的联合目标为:(1)当有外部事件到达时,团队中各控制中心需要对接收到该事件达成共识,要求活跃控制中

心去处理该事件，并对事件的处理结果达成共识；(2)团队要保持一定数量的成员，这是为了应对成员崩溃，维持团队稳定的一个隐含团队目标。

由于团队中的成员分为活跃控制中心及其待命副本，各成员功能不同，因此由联合目标得出不同的成员目标。

对于活跃控制中心主体，其个体目标包括：(1)处理外部事件：把外部任务分解分配给工作主体，对子任务分配情况与其他成员达成共识；外部任务执行结束返回结果时，与团队成员达成共识，表明该任务已结束；故障发生时，对于要处理的内容与团队成员达成共识；(2)当控制中心副本数量不够时，生成新的控制中心副本，并与团队成员达成共识。

对于待命控制中心副本，其个体目标包括：(1)在发现有事件超时没有被处理时，与所有成员达成“该事件没有丢失且未被处理”的共识；(2)发现团队中成员数量不够时，达成共识后通知活跃控制中心处理。

在活跃控制中心崩溃后，由中继主体指定的新活跃控制中心在开始运行前需要与团队中其他成员就目前执行任务情况达成共识。设团队中所有成员记录的任务完成情况构成集合 $\{T_1, \cdots, T_N\}$，N 为团队成员数量，对于每一个成员记录的任务完成情况 T_i 又分为已完成任务集合 T_{i_c} 与未完成任务集合 T_{i_o}，设新活跃控制中心需要跟踪的关键子系统未完成任务集合为 T_{open}，则确定 T_{open} 的算法如下：

算法 3 关键子系统任务执行情况一致性算法：

(1)令 $T_S = \bigcap_{i=1,N} T_{i\_o}$，$T_L = \bigcup_{i=1,N} T_{i\_o}$，显然 $T_S \subseteq T_{\text{open}} \subseteq T_L$；

(2)如果 $T_S == T_L$，则 $T_{\text{open}} = T_S$，算法结束；

(3)否则令 $T_{\text{open}} = T_S$，$T_{\text{diff}} = T_L - T_S$，令 $T_C = \bigcup_{i=1,N} T_{i\_c}$，对于 $\forall x \in T_{\text{diff}}$，若 $x \notin T_C$，则将 x 加入 T_{open}，算法结束。T_{open} 就是新活跃控制中心需要去跟踪的子系统未完成任务集合。

4 仿真实验

为了分析 RCSM 引入的冗余操作对系统运行效率的影响，使用多主体平台 MAGE 进行了若干仿真实验。多主体平台 MAGE 是一个面向主体的软件开发、集成和运行环境[15]，主要基于智能主体和多主体技术，为用户提供一种面向主体的软件开发和系统集成模式，以主体为最小粒度，封装和自动化实现了主体的一般性质，应用程序可以通过添加特殊的主体行为方便地实现自己的特定功能。

仿真实验由三部分构成：原始关键子系统、采用 RCSM 封装后的关键子系统、测试用任务集合。实验所用的计算环境为 CPU P4 2.8G、内存 1G。

实验参数设置：子系统中工作主体能够执行的原子任务一共有 20 种；子系统中包含 40 个工作主体，每个主体被随机赋予一部分原子任务的执行能力，并设定所有主体的运算能力相同；每个主体需要保存的检查点数据的大小为 1M 字节。

测试用任务集合设置：使用 20 种原子任务随机生成 100 个任务树，分 10 组，每组 10 个任务树。

本实验的目的是分析使用 RCSM 对关键子系统运行效率的影响，要观测的对比实验结果是随机生成的 10 组任务中每组任务运行的总时间。首先设置工作主体执行不同原子任务所需时间。

第一次设置：随机选取 1 ~ 2s 之间的值作为 20 个原子任务的执行时间。

需要注意的是该时间仅为工作主体执行原子任务的净时间，不包括主体之间执行任务时可能的同步等待时间。原始子系统的任务执行时间与 RCSM 封装后子系统任务执行时间对比结果见图 3。

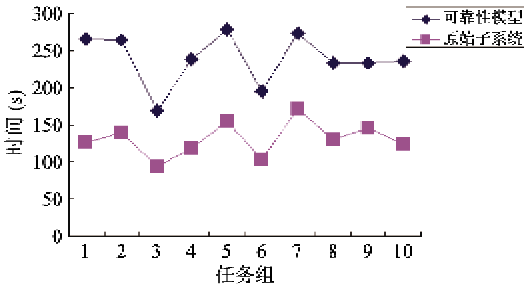

图 3 第一次对比实验结果

第二次设置：随机选取 5 ~ 10s 之间的值作为 20 个原子任务的执行时间。对比执行结果见图 4。

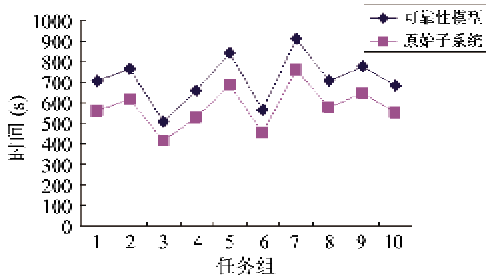

图 4 第二次对比实验结果

通过两次对比实验可以发现,当原子任务的运行时间较短时,模型会对系统的运行效率带来较大的负担,对于第一次设置的情况,最高影响达到了将近100%。然而由于主体记录检查点信息所产生的花费是基本固定的,在原子任务运行时间较长的情况下,RCSM对于系统的影响将大为降低,在第二次设置的情况下,冗余操作对系统运行效率的影响最高在30%左右。可以看出,可靠性模型比较适用于原子任务的运行需要较长时间的情况,对于原子任务运行时间较短的情况,也可以通过降低检查点的记录频率,即间隔若干任务记录一个检查点的方法来平衡系统运行效率和系统可靠性之间的关系。此外,利用各种技术减少检查点信息的大小也是提高系统运行效率的可行手段。

5 结 论

本文对多主体系统可靠性问题进行了研究,在分析了现有围绕单一主体提出的可靠性模型的不足的基础上,从主体组层面的任务分配执行机制出发,结合多种可靠性技术,提出了一个多主体系统可靠性模型——可靠关键子系统模型(RCSM)。在此模型中使用透明化封装思想来处理关键子系统与外部主体间的关系,将故障的传播限制在可控范围内。在子系统内部采用基于任务分配的一致性检查点设置技术实现工作主体的故障恢复,以减少使用检查点技术的复杂性,提高恢复效率。对模型中的控制中心主体则采用了复制技术与团队协作思想相结合的手段来提高其可靠性。最后通过仿真实验对RCSM对系统正常运行效率的影响进行了分析,说明了该模型的实用价值。

下一步的工作将集中在两个方面:(1)优化关键子系统中的任务分解与分配算法,工作主体故障的恢复效率同当前关键子系统所执行的任务是否具有阶段性有很大关系,因此可以扩展现有的任务分解与分配算法,使其在分解分配任务时能够考虑任务的执行过程是否具有阶段性,以达到平衡任务执行效率与故障恢复效率的目的;(2)RCSM对于故障检测手段并没有具体的要求,之前我们已经开发出了一个基于模型的主体软件故障诊断系统eHealer[16],下一步将把RCSM与该诊断系统结合起来,开发出一个面向多主体系统的故障协同诊断恢复系统。

参考文献

[1] Qu W Y, Shen H, Defago X. A survey of mobile agent-based fault-tolerant technology. In: Proceedings of the 6th International Conference on Parallel and Distributed Computing Applications and Technologies, Dalian, China, 2005. 446-450

[2] Hägg S. A sentinel approach to fault handling in multi-agent systems. In: Proceedings of the 2nd Australian Workshop on Distributed AI, in conjunction with the 4th Pacific Rim International Conference on Artificial Intelligence, Cairns, Australia, 1996. 181-195

[3] Fedoruk A, Deters R. Improving fault-tolerance by replicating agents. In: Proceedings of the 1st International Joint Conference on Autonomous Agents and Multi-agent Systems, Bologna, Italy, 2002. 737-744

[4] Taesoon P, Ilsoo B, Hyunjoo K, et al. The performance of checkpointing and replication schemes for fault tolerant mobile agent systems. In: Proceedings of the 21st IEEE Symposium on Reliable Distributed Systems, Osaka, Japan, 2002. 256-261

[5] Wu Z G, Fang B X. Research on extensibility and reliability of agents in Web-based computing resource publishing. In: Proceedings of the 4th International Conference/Exhibition on High Performance Computing in the Asia-Pacific Region, Beijing, China, 2000, 1. 432-435

[6] Horling B, Lesser V. A survey of multi-agent organizational paradigms. *The Knowledge Engineering Review*, 2004, 19(4): 281-316

[7] Castelfranchi C. Decentralized AI, Chapter Dependence Relations in Multi-agent Systems. New York: Elsevier, 1992

[8] Sichman J S, Conte R, Demazeau Y. A social reasoning mechanism based on dependence networks. In: Proceedings of European Conference on Artificial Intelligence, Amsterdam, Netherlands, 1994. 416-420

[9] Sichman J S, Conte R. Multi-agent dependence by dependence graphs. In: Proceedings of the 1st International Joint Conference on Autonomous Agents and Multi-agent Systems, Bologna, Italy, 2002. 483-490

[10] Strasser M, Rothermel K. System mechanism for partial rollback of mobile agent execution. In: Proceedings of the 20th International Conference on Distributed Computing System, Taipei, Taiwan, 2000. 20-28

[11] Zhang Y H, Wong D S, Zheng W M. User-level checkpoint and recovery for LAM/MPI. *ACM SIGOPS Operating Systems Review*, 2005, 39(3): 72-81

[12] Schulz M, Bronevetsky G, Fernandes R, et al. Implementation and evaluation of a scalable application- level

checkpoint-recovery scheme for MPI programs. In: Proceedings of the 2004 ACM/IEEE Conference on Supercomputing, Pittsburgh, USA, 2004, 38-38

[13] Bozyigit M, Wasiq M. User-level process checkpoint and restore for migration. *ACM SIGOPS Operating Systems Review*, 2001, 35(2): 86-96

[14] Cohen P R, Levesque H J. Teamwork. *Special Issue on Cognitive Science and Artificial Intelligence*, 1991, 25 (4): 487-512

[15] Shi Z Z, Zhang H J, Cheng Y, et al. MAGE: an agent-oriented software engineering environment. In: Proceedings of the 3rd IEEE International Conference on Cognitive Informatics, Victoria, Canada, 2004. 250-257

[16] 张冬蕾, 韩旭, 史忠植. 基于主体的软件故障诊断系统 eHealer. 高技术通讯,2010,20(4):379-385

RCSM: a reliable critical subsystem model in multi-agent system

Zhang Donglei[*,**], Shi Zhongzhi[**], Wang Peng[*], Zhao Liang[*]

([*] Unit 61226 of People's Liberation Army, Beijing 100079)

([**] The Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190)

Abstract

It was paid great attention that a multi-agent system is composed of autonomous software agents, which makes it more difficult to deal with the failures in the system than in a traditional distributed system, and the reliability problem of multi-agent systems was studied. This study tried to resolve the failures at the agent group level, and, as a result, a reliable critical subsystem model, called the RCSM for short, was introduced with the aim of improving the reliability of a multi-agent system. The RCSM consists of three types of agent: Relay Agent, Control Center Agent, and Worker Agent. In RCSM, the Relay Agent encapsulates the critical subsystem, which is formed by Worker Agents, to avoid wide spread of failures by hiding the actual topology of critical subsystem. Within the subsystem RCSM adopts a consistent checkpoint mechanism based on task assignment to recover the failures in Worker Agent, and combines replication with teamwork to increase the robustness of Control Center Agent. The fault recovery efficiency of the RCSM and its influence on system load were analyzed by simulation experiments to illuminate the practicability of RCSM.

Key words: multi-agent system (MAS), reliable model, critical subsystem, checkpoint, replication