VINCASim:一种网格工作流可靠性仿真工具

赵小伟[1] 张利永[2] 韩燕波[2]
(山东科技大学信息科学与工程学院 青岛 266510)[1]
(中国科学院计算技术研究所网络科学重点实验室 北京 100190)[2]

摘 要 网格工作流作为综合利用网格资源求解问题的"编程"技术已得到广泛应用,其可靠性保障研究得到越来越多的关注。然而,面对网格环境固有的复杂性和不确定性,如何对可靠性保障方法有效、方便地进行评测,是一个有待深入探索的挑战性问题。以VINCA网格工作流为参考原型,对影响其可靠性的因素进行了系统的分析,抽象出了网格工作流系统组件模型和可靠性属性模型,并基于GridSim实现了一个可配置、易扩展的网格工作流可靠性仿真工具包——VINCASim。此工具可以通过配置方便地模拟节点失效、工作流引擎失效、网络连接失效、流程执行异常等出错情形,提供扩展接口支持以编程的方式引入各种可靠性保障方法,为评测不同方法的有效性提供可控的、可重复的实验平台。通过场景示例说明了该工具的扩展性和易用性。

关键词 网格工作流,可靠性,仿真工具
中图法分类号 TP311 **文献标识码** A

VINCASim: A Simulation Toolkit for Evaluating Dependability of Grid Workflows

ZHAO Xiao-wei[1] ZHANG Li-yong[2] HAN Yan-bo[2]
(College of Information Science and Engineering, Shandong University of Science and Technology, Qingdao 266510, China)[1]
(Key Laboratory of Network Science and Technology, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China)[2]

Abstract As a special kind of "programming" technology for constructing problem-solving applications on the basis of grid resources, grid workflow has been widely applied. Methodologies for ensuring ddependability of grid workflows have attracted attention. However, it remains a challenge how to evaluate the effectiveness of these methodologies due to the complexity and uncertainty of grid environments. Based on VINCA grid workflow, key factors that affect the depen dability were systematically analyzed, and a general component model and dependability attributes model for grid workflow systems were established. A configurable and extensible simulation toolkit called VINCASim for evaluating the dependability of grid workflows was developed based on GridSim. The toolkit can simulate various failures raised from grid nodes, workflow engines, network connection and workflow execution in a configurable manner, and supports incorporating different dependability ensuring methods programmatically. Thus, a controllable and repeatable experiment platform was provided for evaluating different methods. The usability was demonstrated by a use case scenario.

Keywords Grid workflow, Dependability, Simulation

1 引言

近年来,网格工作流作为一种利用网格资源求解应用问题的"编程"技术,在构建跨组织、跨学科的复杂问题求解环境方面得到广泛应用。由于网格环境的自治性和动态性,网格工作流的可靠性保障方法得到越来越多的关注。

由于底层网格平台的差异和所求解的领域问题的不同,网格工作流尚没有公认的、统一的描述和运行规范,其可靠性保障技术也不尽相同。已有的方法包括任务重试执行[1]、任务重调度[1]、流程检查点恢复[2]、冗余复制执行[3]以及基于可靠度预测的任务调度[4]等。

在中国国家网格(CNGrid)环境中,存在多种不同类型的网格工作流引擎(BPEL, XPDL, JSDL等),引擎采取的可靠性保障技术也不同。我们曾开发了支持元调度的VINCA网格工作流系统[5],并提出了基于引擎可靠度预测的主动式可靠性保障机制[4],试图利用多个网格节点上引擎的执行能力,通过增加冗余引擎的方式,构建一个可靠、稳定的网格工作流执行环境。

如何有效、方便地对可靠性保障方法进行评测,成为迫切需求。然而,网格工作流的执行需要协调利用分布在不同物理域、管理域的多种网格资源,网格环境的分布性和复杂性使得难以甚至不可能在真实环境下对各种可靠性保障方法进行

到稿日期:2008-12-15 返修日期:2009-02-25 本文受国家"八六三"高科技研究发展计划(2006AA12Z202)及中国科学院奥运科技项目(KACX1-03)资助。

赵小伟(1984—),男,硕士生,主要研究方向为网格工作流可靠性、业务流程管理等,E-mail:mathzxw2002@gmail.com;张利永(1980—),男,博士生,主要研究方向为网格工作流可靠性、业务流程管理等;韩燕波(1962—),男,博士生导师,主要研究方向为服务计算等。

研究。

仿真技术作为对真实环境的抽象和模拟，可以提供可控的、可重复的实验平台，有针对性地对问题进行深入研究，在众多研究中被广泛应用。网格计算方面也已有多种模拟工具用于任务调度、性能评测等。

本文以 VINCA 网格工作流[5]为参考原型，对影响网格工作流可靠性的因素进行了系统的分析，抽象出了网格工作流系统组件模型和可靠性属性模型，通过扩展 GridSim[6]实现了一个可配置、易扩展的针对网格工作流可靠性评测的仿真工具 VINCASim。该工具可以通过配置方便地模拟节点失效、工作流引擎失效、网络连接失效、流程执行异常等出错情形，提供扩展接口支持以编程的方式引入各种可靠性保障方法，为评测不同方法的有效性提供可控的、可重复的实验平台。以 VINCA 网格工作流为例，对文献[4]中所提出的方法进行了模拟，并进一步分析了本工具的扩展性和易用性。

本文第 2 节分析了相关工作，并重点介绍了 GridSim；第 3 节对影响网格工作流可靠性的因素进行了系统的分析；第 4 节给出了 VINCASim 仿真工具的架构及具体实现；第 5 节针对文献[4]提出的方法展示了 VINCASim 的使用方法；最后，对未来工作进行了展望。

2 相关工作

2.1 可靠性保障方法

目前，网格工作流领域比较有代表性的可靠性保障方法有 Taverna[11]采用重试执行任务和重调度任务到其他资源两种方式进行异常处理；文献[11]中提出一种流程实例迁移机制，支持从一个引擎上导出出错流程实例相关的状态数据、业务数据，并导入到其他引擎中，使得流程实例可以继续执行；DAGMan[2]提供任务级的检查点恢复机制，任务出错时透明地选择其他资源并从检查点重新执行；文献[3]提出的方法通过冗余执行任务的方式保证至少有一个任务副本可以成功返回结果；在 BPEL[12]中，用户可针对某一类异常自定义相应的补偿活动，异常被捕获后引擎自动调用相应的补偿活动。

2.2 网格模拟器

目前已有一些网格系统的仿真工具，如 GridSim[6]，OptorSim[7]，SimGrid[8]，MicroGrid[9]等。OptorSim 作为欧洲数据网格的一部分，主要用于评估数据网格中的数据副本管理策略(用于优化数据访问效率)，可以方便地配置网格拓扑结构、初始文件分布、网络带宽、作业列表等，能够模拟不同节点的负载以及网络拥塞程度等。SimGrid 是一种基于 C 语言开发的仿真工具，为在分布异构环境下进行应用程序调度研究提供了一个仿真环境，并提供了一系列核心函数用于生成资源模型(时间共享)和任务模型，其中资源模型包含处理机和网络连接，任务模型包含消耗处理机资源的计算任务和消耗网络连接的数据传输资源，任务间通过依赖关系建立执行的次序；MicroGrid 试图利用现有的物理资源(比如一个 cluster)来模拟一个虚拟的网格环境，应用程序运行在真实系统之上的虚拟层次，模拟的结果比较精确，有助于优化网格系统的设计以及评估网格系统的性能，但应用程序需要真正运行完成，因此模拟的效率不高，并且 MicroGrid 是面向 Globus 环境的，对通用调度算法有一定局限性。

GridSim 是一种被广泛使用的网格系统仿真工具，可以对基于虚拟组织的资源分配、工作流调度等进行仿真和评估，从而优化网格资源调度算法，可以对网格环境中的异构资源(时间共享和空间共享)、用户、应用程序、资源调度器等进行抽象和建模。

GridSim 提供了良好的扩展机制，通过继承等方式可以抽象、模拟网格环境中的其他元素。文献[10]通过扩展 GridSim，抽象出了具有一定失效概率的网格资源、网格用户、失效检测器等元素，对网格环境中资源失效、错误检测等进行了仿真，可以将网格资源的可用性情况反馈给用户，从而保证用户将任务提交给一个可用的网格资源来处理。但是文献[10]并不能满足网格工作流的可靠性仿真需求：除了考虑网格资源的失效外，还要考虑节点失效、工作流引擎失效、网络连接失效、流程执行异常等出错情形以及如何监控引擎状态、接入流程请求调度策略等。

综上，目前已有的网格仿真工具主要是从性能方面对网格系统进行仿真，可靠性方面的仿真尚不能满足网格工作流的需求。由于 GridSim 具有良好的可扩展性，因此本文通过扩展 GridSim 的方式，在对网格工作流可靠性进行系统分析的基础上，抽象出可靠性相关组件，开发了 VINCASim。基于 VINCASim 可以对网格工作流系统进行仿真，从而优化可靠性调度算法。

3 可靠性因素分析

本节将以 VINCA 网格工作流为参考原型对影响流程正常执行的因素进行系统的分析。

3.1 VINCA 网格工作流

VINCA 网格工作流是一种分布式、多引擎架构的网格工作流系统。在每个网格节点上除存在大量网格资源外，还存在不同类型的流程执行引擎。

VINCA 网格工作流具有层次式结构：网格资源层、流程执行引擎层、流程请求调度器层。其拓扑结构如图 1 所示。

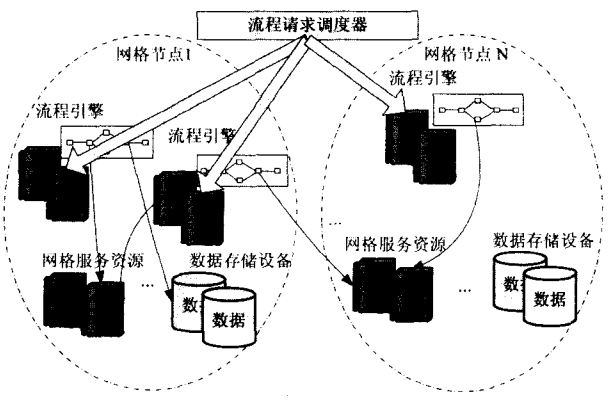

图 1 VINCA 网格工作流拓扑结构

网格资源层由网格(Web)服务资源、数据存储资源等构成，可以承载一定的计算任务和数据存储任务等。

流程执行引擎层包括以下组件：引擎状态监控器、流程执行引擎、任务执行引擎。引擎状态监控器监控引擎的状态信息，如接受的流程请求数、当前负载、执行失败的流程实例数、执行成功的流程实例数等。流程执行引擎负责执行流程实例，并通过任务引擎将任务调度到网格资源上。

请求调度器层负责接受用户执行流程的请求，然后按照一定的可靠性保障机制(如基于引擎可靠度预测的主动式可

靠性保障机制)将流程请求调度到合适的流程执行引擎上,来提高流程执行成功的概率。

3.2 影响可靠性的因素

由 VINCA 网格工作流的拓扑结构可知,影响流程执行可靠性的因素可以发生在以下层:

(1)网格资源层。网格资源具有动态、自治的特点,当网格资源发生错误,不能正常工作或不能被访问时,会导致依赖该网格资源的流程执行失败。此时称网格资源失效导致流程执行失败。

特别地,由于流程执行时依赖的网格资源需要通过流程执行引擎来进行调度,所以当流程执行引擎对某流程依赖的网格资源没有访问权限时,也会引发流程执行失败,此时也称网格资源失效导致流程执行失败。

(2)流程执行引擎层。引擎的状态也具有动态性,引擎崩溃、引擎重启等会引起其上运行的流程执行失败。此时称流程引擎失效导致流程执行失败。

在 VINCA 网格工作流环境中,引起流程引擎失效的原因主要有引擎宕机、引擎重启以及引擎崩溃(如超过最大负载)等。

(3)流程请求调度器层。当流程请求调度器宕机或发生错误时,会导致其无法成功响应流程请求,从而导致整个系统崩溃。

图 2 表示了 VINCA 网格工作流环境下影响流程正确执行的情况。1,2 表示网格资源失效的情况,3 表示流程引擎失效的情况,4 表示流程请求调度器无法响应流程请求的情况。

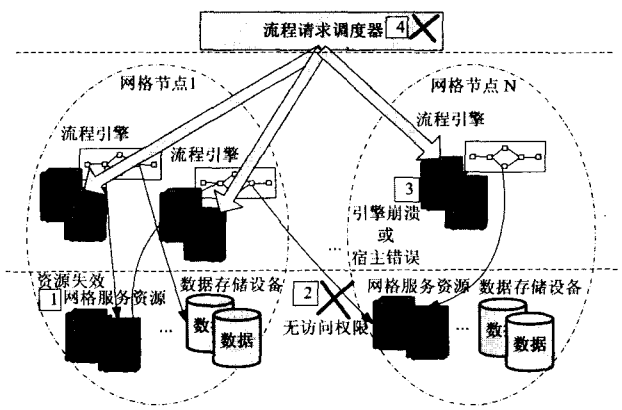

图 2 影响流程执行可靠性的因素

另外,影响 VINCA 网格工作流可靠性的因素还有网络连接等。当网络连接发生错误的时候,可以认为与该网络连接的网格资源或者流程引擎失效。

4 VINCASim 架构及实现

4.1 VINCASim 架构

为了方便、有效地模拟网格工作流可靠性评测环境,需要:

1)模拟网格工作流中的实体(网格资源、流程执行引擎、流程请求调度器等)以及实体异常情况;

2)能够方便地设置实体的可靠性属性(如网格资源失效概率、流程执行引擎的最大负载等);

3)能够方便地生成各实体之间的拓扑关系;

4)方便地监控各实体的状态以及异常情形;

5)易扩展,允许方便地接入不同的流程请求调度策略;

6)方便地统计仿真结果。

为了满足上述需要,本节抽象出了网格工作流的系统组件模型和可靠性属性模型。

4.1.1 流程请求调度器

可以模拟流程请求的到达、流程请求的长度等,并按照一定的请求调度策略将流程请求调度到流程执行引擎上去。由以下组件构成:

(1)流程请求生成器

① 可以按照一定的概率分布生成流程请求的到达时间;

② 可以随机生成流程请求,其中流程的执行时间长度符合一定的概率分布。

(2)流程请求调度策略

通过引擎状态监控器获取各流程执行引擎的状态信息,并按照一定的流程请求调度策略将流程请求调度到流程执行引擎上。用户可以按照接口规范设计不同的流程请求调度算法。

4.1.2 流程执行引擎

负责流程实例的执行。可以模拟流程实例的执行情况,通过任务引擎进行流程任务调度。可以设置以下属性:

①最大流程实例数(最大负载)

这里假定引擎的实例数越大,表示引擎的执行能力越强。当引擎超过最大负载的时候,所有接收到的流程请求都不会成功响应。流程实例可以设置流程实例标识、流程长度、起止时间、运行状态等属性。

②引擎重启之后,流程实例是否可继续执行

如果设置为否,则在引擎宕机后所有正在运行的流程实例会被置于执行失败的状态。

③执行流程失效概率

可以按一定概率模拟流程实例执行失败的情况。

4.1.3 任务引擎

负责将应用任务调度到网格资源上。可以采取一定的任务调度策略将任务调度到可用的网格资源上。不同的任务引擎采取的任务调度策略可能不同,从而导致应用任务执行失败的概率也不同。

4.1.4 网格资源

负责执行应用任务或者存储数据等。

4.1.5 网络连接

通过设置不同组件之间的网络连接,可以生成网格工作流的拓扑结构。

4.1.6 状态监控器

分为网格资源状态监控器和流程执行引擎状态监控器。可以按一定的概率分布模拟流程执行引擎(或网格资源)的宕机,同时可以监控流程执行引擎(或网格资源)的状态。

VINCASim 架构如图 3 所示,由于文献[10]已经对网格资源失效以及状态监控做了一定的工作,因此 VINCASim 是在文献[10]的工作基础上进行的。

仿真开始,VINCASim 根据网络连接配置生成网格工作流组件以及拓扑结构,流程请求调度器随机生成流程请求并按照调度策略进行调度,流程执行引擎负责流程实例的执行,并按一定出错概率返回流程实例执行结果;仿真结束,可以对仿真过程中的信息进行统计,便于直观地分析仿真过程中生

成的数据，优化调度算法。

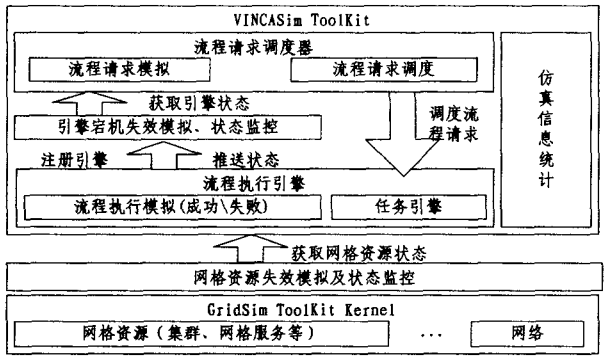

图 3　VINCASim 架构图

VINCASim 具有模块化设计，可以对不同的模块按照需求进行依赖注入，方便扩展。

4.2　模拟具体实现

本节主要介绍 VINCASim 如何对流程请求的到达、流程执行时间长度、流程执行引擎的宕机以及执行流程失效等进行模拟。

4.2.1　“发生次数”与“等待时间”的关系

在本文中，“发生次数”表示流程请求的到达个数、引擎的宕机次数、引擎执行流程实例失败的次数等，“等待时间”表示两次请求之间的时间间隔、两次引擎宕机之间的时间间隔或者两次执行流程实例失败的时间间隔等。

假设时刻 $t-\Delta t$ 到时刻 t 的 Δt 时间间隔内“发生次数”ξ 为 k 的概率为 $P(\xi=k)=\alpha(\Delta t)$，$k=0,1,2,\cdots$，其中 α 为 Δt 的函数表达式且有 $0\leqslant\alpha(\Delta t)\leqslant 1$，则“等待时间”$\eta$ 的分布函数为 $F(\Delta t)=P(\eta\leqslant\Delta t)=1-P(\eta>\Delta t)=1-P(\xi=0)$。

特别地，当“发生次数”符合泊松分布时，即 $F(\xi=k)=\frac{(\lambda\Delta t)^k}{k!}e^{-\lambda\Delta t}$，$k=1,2,3\cdots$ 时，“等待时间”符合指数分布 $F(\Delta t)=P(\eta<\Delta t)=\begin{cases}1-e^{-\lambda\Delta t}, & \Delta t>0\\ 0, & \Delta t\leqslant 0\end{cases}$。

4.2.2　模拟实现方式

(1)流程请求模拟

仿真开始，设置流程请求的个数，然后按一定概率随机生成流程请求的到达时间，同时随机生成流程请求的长度。然后，从引擎状态监控器获取可用的引擎，并选择合适的引擎调度流程请求。若没有可用的引擎，则等待一定的仿真时间再进行调度。用户可以实现不同的请求模拟器，从而模拟不同的流程请求负载。

(2)引擎宕机模拟

引擎状态监控器负责模拟流程执行引擎的宕机。仿真开始会按照一定的概率分布生成引擎的宕机数目，相应地随机生成引擎的宕机时刻。在每个的宕机时刻，随机选择一个流程，执行引擎宕掉。引擎的宕机时间长度也需要符合一定的概率分布。引擎会在宕机时间长度到达时重新注册到引擎状态监控器中。

(3)流程执行失效模拟

首先引入失效强度的概念。

失效强度：假定流程的执行失效是由网格资源的失效引起的，不同的网格资源引起的流程执行失效数不一样。同时，由于不同的流程执行引擎采取的任务调度策略不同，因此不同的引擎每次流程执行失效都会导致不同数目的流程实例执行失败。

每个流程执行引擎会按照一定概率分布生成一定数量的失效数，同时会随机生成失效时刻。在每个失效时刻，引擎按照失效强度随机选择失效的流程实例。

所以，在假设时刻 $t-\Delta t$ 到时刻 t 的 Δt 时间间隔内流程执行失败概率相等的情况下，流程执行时间越长，执行失败的概率越大。

根据引擎的实现方式不同，引擎宕机对流程执行失效也会产生不同的影响。本文考虑两种情况：若引擎具有在重启之后继续执行流程实例的能力，则流程实例不会因为引擎的宕机而失败；另一种情况是，引擎重启之后，所有正在运行的流程实例都会执行失败。

仿真过程中，失效次数、请求次数、失效时间间隔、请求时间间隔等需要符合 4.2.1 节中的关系。

当前，VINCASim 支持超指数分布、Γ 分布、威布尔分布等。

4.3　引擎状态检测

VINCASim 采用“PUSH”的方式检测流程执行引擎的失效以及状态信息。即每间隔一定的时间，各引擎向状态监控器发送“I'm alive”消息表示没有失效，同时将自己的状态信息推送给引擎状态监控器。若间隔一定时间状态监控器没有收到该消息，则认为该引擎宕机。

为了对不同的可靠性调度算法提供支持，VINCASim 中的引擎状态表示引擎接收的请求数、引擎成功执行的流程实例数、失败的流程实例数等。

假设 $R(t-\Delta t,t)$ 为时刻 $t-\Delta t$ 到时刻 t 的 Δt 时间间隔内引擎收到的执行流程的请求数；$S(t-\Delta t,t)$ 为由时刻 $t-\Delta t$ 到时刻 t 的 Δt 时间间隔内引擎成功执行的流程数；$E(t-\Delta t,t)$ 为由时刻 $t-\Delta t$ 到时刻 t 的 Δt 时间间隔内引擎失败执行的流程数。

基于以上信息，可以得出引擎的其他特征信息。

(1)引擎的负载

时刻 $t-\Delta t$ 到时刻 t 的 Δt 时间间隔内引擎的负载可表示为接收到的请求数减去完成(成功的与失败的)的请求数，公式为 $l(t-\Delta t,t)=R(t-\Delta t,t)-S(t-\Delta t,t)-E(t-\Delta t,t)$。

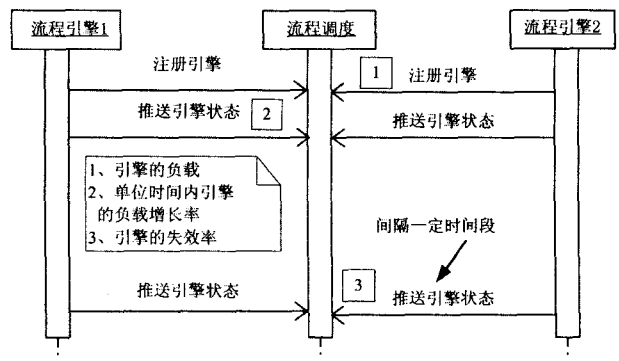

图 4　流程引擎状态检测时序图

(2)引擎的负载增长率

其中单位时间内引擎的负载增长率可表示为

$$\lambda(t-\Delta t,t)=\frac{l(t)-l(t-\Delta t)}{\Delta t}$$

(3)引擎执行流程失效率

时刻 $t-\Delta t$ 到时刻 t 的 Δt 时间间隔内引擎执行流程的失效率可以表示为

$$\mu(t-\Delta t,t)=\frac{E(t-\Delta t,t)}{\Delta t}$$

图 4 表示了流程引擎的状态检测时序图。状态检测器中维护着流程执行引擎的最新状态信息。

5 场景示例及分析

本节将使用 VINCASim 搭建一个示例场景，模拟 VINCA 网格工作流的运行时环境，从而对文献[4]中的可靠性调度算法进行模拟并对其结果进行分析。

5.1 可靠性算法介绍

首先，对文献[4]中的可靠性调度算法——“基于引擎可靠度预测的主动式可靠性保障算法”进行简单介绍。首先引入“引擎可靠度”的定义。

定义 1 引擎可靠度：

时刻 $t-\Delta t$ 到时刻 t 的 Δt 时间间隔内引擎的可靠度为

$$\rho(t-\Delta t,t)=\frac{1}{\alpha\cdot e^{\mu(t-\Delta t,t)}+(1-\alpha)\cdot e^{\lambda(t-\Delta t,t)}}(0\leqslant\alpha\leqslant1)$$

其中，α 为系数因子，用于调节 $\mu(t-\Delta t,t)$ 和 $\lambda(t-\Delta t,t)$ 对可靠度的影响权重。

由引擎可靠度定义可知，流程执行引擎的可靠度与引擎在一段时间内的负载增长率、执行流程失效率有关。通常情况下，如果引擎的执行流程失效率比较大，则不宜将流程请求调度到该引擎上；如果引擎的负载增长率比较高，也不宜将流程请求调度到该引擎上。

该可靠性算法会基于引擎运行的历史信息计算引擎在 t 时刻之前的 Δt 时间段内的负载增长率以及执行流程失效率，从而计算引擎在 t 时刻的可靠度。将流程请求调度到“最可靠”的流程执行引擎上，从而保障 VINCA 网格工作流的可靠性。

5.2 场景介绍

假设 $P(\lambda)$ 表示泊松分布，其中 λ 表示均值，表示一段时间平均“发生次数”为 λ。$H(\mu,\varepsilon)$ 表示超指数分布，其中 μ 表示均值，ε 表示方差，表示平均“等待时间”为 μ。以下所指的时间均为仿真时间。

假设有 3 个网格节点，每个网格节点都有 1 个流程执行引擎。其中流程执行引擎的参数如表 1 所列。

表 1 引擎参数表

	流程失效数	失效强度	最大负载
引擎 1	$P(10)$	$P(10)$	20
引擎 2	$P(15)$	$P(10)$	20
引擎 3	$P(20)$	$P(10)$	20

假设流程执行引擎在时刻 $t-\Delta t$ 到时刻 t 的 Δt 时间间隔内宕机次数符合泊松分布 $P(15)$，宕机失效间隔符合超指数分布 $H(25,200)$，宕机之后恢复正常的时间长度符合超指数分布 $H(25,50)$。

假定两次随机的流程请求的间隔时间符合分布 $H(5,200)$，每个流程的长度符合分布 $H(100,110)$。

由以上设置可知，流程请求以较快的速度到达。同时，与引擎两次宕机之间的“等待时间”相比较，流程的长度也较长，这主要是为了增大流程实例执行期间引擎宕机的概率。

5.3 仿真结果分析

假设可靠度计算公式中的 $\Delta t=1000$，调整 α 的值，分别令 α 的值为 0.9，0.2，0.1。图 5 表示当请求数为 500 的时候，取不同的 α 值实验 10 次成功执行的流程请求数。

图 6 表示当流程请求数为 100，300，500，700 的情况下，α 取不同值的时候成功执行的流程数(测量 10 次，取均值)。

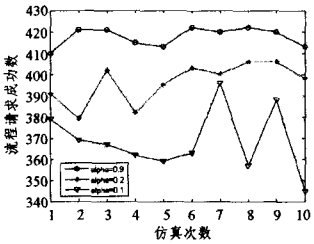

图 5 流程请求数为 500，α 取不同的值

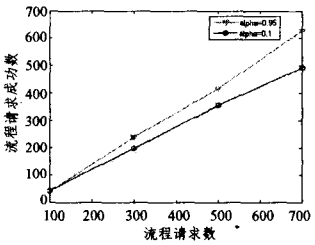

图 6 对不同的流程请求数 α 取不同的值

为了便于分析该可靠性算法，还可以监控流程请求调度器进行请求调度的时刻各引擎的状态信息。表 2 为 Δt 取 1000、α 取 0.1 的情况下，在时刻 $t=43.6832421912243$ 引擎 3 的状态信息。

表 2 引擎 3 在时刻 t 的状态信息

实例标识	开始时间	结束时间	流程长度	执行状态
fa81c64a	15.86997837	16.540023	0.670044628	成功
fa81c654	32.88077755	40.90021057	50.27004767	失败
fa84375a	38.08321303	40.90021057	112.552782	失败
fa81c650	29.85400372	40.90021057	604.127013	失败
fa81c64c	23.20695723	34.93655027	11.72959304	成功
fa843756	36.61619748	40.90021057	97.10917635	失败

由可靠性公式可得引擎的可靠度为 0.920553934449901。

结束语 本文在对影响网格工作流可靠性因素进行充分分析的基础上，抽象出了网格工作流系统组件模型和可靠性属性模型。基于 GridSim 实现了一种可配置、易扩展的网格工作流可靠性仿真工具 VINCASim。基于 VINCASim 可以模拟流程执行引擎宕机、流程执行失败、流程请求调度等。同时 VINCASim 提供了相应的状态检测机制来检测流程执行引擎的状态。VINCASim 为优化网格工作流中的可靠性调度算法提供了一种可控的、可重复的实验平台。

然而，VINCASim 还有一些需要完善的地方。如①VINCASim 只能以 API 的形式创建组件模型、设置可靠性属性等，尚需开发可视化工具；②没有单独考虑网络连接失效的情况，而是将其归结为网格资源失效或者流程执行引擎失效的情况。

参 考 文 献

[1] Taverna User Manual[EB/OL]. http://www.mygrid.org.uk/usermanual1.7/,2008

[2] Couvares P, Kosar T, Roy A, et al. Workflow Management in Condor[J]. Workflows for e-Science, 2007: 357-375

[3] Abawajy J H. Fault-Tolerant Scheduling Policy for Grid Computing Systems[C]// 18th International Parallel and Distributed Processing Symposium (IPDPS'04). Santa Fe, New Mexico, USA, April 2004: 238-244

[4] 张利永. 一种主动式的网格工作流可靠性保障方法[J]. 中山大学学报, 2008

(下转第 185 页)

有的面向语法的服务匹配所存在的主要缺点，然后分析了面向语义的服务匹配方式。面向语义的服务匹配能很好地解决服务之间的语义问题，但同时引入了逻辑推理的复杂度和资源的高需求；在此基础上针对普适计算环境，给出基于语义的服务匹配所需要的匹配的效率、资源约束性问题的解决方案，并给出了上下文感知的服务匹配以及面向服务组合的服务匹配相关的研究。目前在普适计算环境下有关基于语义的服务匹配的研究还比较少。本文通过比较相关的研究方法，找出需要解决的问题，以便下一步设计出适合普适计算环境的服务匹配算法。

参 考 文 献

[1] Weiser M. The computer for the 21st century [J]. Scientific American, 1991, 265(3): 94-104

[2] Arnold K, O'Sullivan B, Scheifler R W, et al. The Jini specification[M]. Addison-Wesley, 1999

[3] Universal Plug and Play (UPnP) [EB/OL]. http://www.upnp.org

[4] Guttman E, Perkins C E, Veizades J, et al. Service Location Protocol[R]. Version 2, IETF, RFC 2608, June 1999

[5] Miller B A, Pascoe R A. Salutation Service Discovery in Pervasive Computing Environments[R]. IBM Pervasive Computing White Paper, February 2000

[6] Lee C, Helal A, Desai N, et al. Konark: A System and Protocols for Device Independent, Peer-to-Peer Discovery and Delivery of Mobile Services [J]. IEEE Transactions on Systems, Man and Cybernetics, 2003, 33(6): 682- 696

[7] Bluetooth SIG. Specification of the Bluetooth System Core [EB/OL]. Version 1.0B volume 1, 1999. Part E. http://www.bluetooth.com/link/spec/bluetooth_e.pdf

[8] UDDI. The UDDI Technical white paper [EB/OL]. http://www.uddi.org/, 2000

[9] Schumacher M, Helin H, Schuldt H. CASCOM: Intelligent Service Coordination in the Semantic Web[D]. Birkhauser, Basel, September 2008

[10] Mokhtar S B, Preuveneers D, Georgantas N, et al. EASY: Efficient SemAntic Service Discovery in Pervasive Computing Environments with QoS and Context Support[J]. Journal of System and Software, 2008, 81(5): 785-808

[11] Constantinescu I, Faltings B. Efficient matchmaking and directory services [C]// IEEE/WIC International Conference on Web Intelligence (WI'03). Washington: IEEE Computer Society, 2003

[12] Srinivasan N, Paolucci M, Sycara K. Adding owl-s to uddi, implementation and throughput[C]// Workshop on Semantic Web Service and Web Process Composition. California, 2004

[13] Preuveneers D, Berbers Y. Encoding Semantic Awareness in Resource-constrained Devices[J]. IEEE Intelligent Systems, 2008, 23(1): 26-33

[14] Yang S J H, Zhang J, Chen I Y L. A JESS enabled context elicitation system for providing context-aware Web services[J]. Expert Systems with Applications, 2008, 34(4): 2254-2266

[15] Broens T, Pokraev S, van Sinderen M, et al. Context-Aware, Ontology-based Service Discovery[C]// Proceedings of the European Symposium on Ambient Intelligence (EUSAI'04). Germany: Springer, 2004

[16] Suraci V, Aiuto S M. Context-aware Semantic Service Discovery [C]// IST Mobile & Wireless Communications Summit 2007. Washington: IEEE Computer Society, 2007

[17] Lecue F, Delteil A, Leger A. Applying Abduction in Semantic Web Service Composition[C]// Proceedings of IEEE International Conference on Web Services (ICWS 2007). Washington: IEEE Computer Society, 2007

[18] 吕庆聪，曹奇英. 一种普适计算环境下基于语义的服务匹配算法[J]. 计算机应用，2008，28(6)：1578-1581

[19] Bandara A, Payne T R, De Roure D, et al. A Pragmatic Approach for the Semantic Description and Matching of Pervasive Resources[C]// GPC 2008. Germany: Springer, 2008

[20] Urbieta A, Azketa E, Gomez I, et al. Towards effects - based service description and integration in pervasive environments[C]// Proceedings of the 3rd International Workshop on Services Integration in Pervasive Environments. New York: ACM, 2008

(上接第 147 页)

[5] Zhang L Y, Zhao Z F, Li H F. Leveraging Legacy Workflow Capabilities in a Grid Environment[C]// Sixth International Conference on Grid and Cooperative Computing (GCC 2007). Urumchi, Xinjiang, China, 2007: 361-365

[6] Sulistio A, Cibej U, Venugopal S, et al. A Toolkit for Modelling and Simulating Data Grids: An Extension to GridSim[C]// Concurrency and Computation: Practice and Experience (CCPE). New York, USA: Wiley Press, Dec. 2007

[7] Bell W H, Cameron D G, Capozza L, et al. OptorSim - A Grid Simulator for Studying Dynamic Data Replication Strategies[J]. International Journal of High Performance Computing Applications, 2003

[8] Casanova H. SimGrid: A Toolkit for the Simulation of Application Scheduling[C]// Proceedings of the First IEEE/ACM International Symposium on Cluster Computing and the Grid. 2001: 430-437

[9] Song H J, Liu Xin, Jakobsen D, et al. The MicroGrid: a Scientific Tool for Modeling Computational Grids[C]// Proceedings of Super Computing 2000. 2000

[10] Caminero A, Sulistio A, Caminero B. Extending GridSim with an Architecture for Failure Detection

[11] Tan W, Fong L, Bobroff N. BPEL4Job: A Fault-Handling Design for Job Flow Management[C]// Service-Oriented Computing-ICSOC 2007. 2007: 27-42

[12] Andrews T, Curbera F, Dholakia H, et al. Business Process Execution Language for Web Services Version 1.1, BEA Systems, IBM Corporation, Microsoft Corporation, SAP AG, and Siebel Systems (2002), developerWorks (updated February 1, 2005) [EB/OL]. http://www.ibm.com/developerworks/library/specification/ws-bpel