一种主动式的网格工作流可靠性保障方法*

张利永[1,2]，韩燕波[1]
（1. 中国科学院计算技术研究所网格与服务计算研究中心，北京 100190
2. 中国科学院研究生院，北京 100039）

摘 要：针对中国国家网格（CNGrid）环境，曾尝试通过一种工作流元调度机制（VINCA 抽象工作流），在为用户提供单一入口和屏蔽细节的同时，优化利用已有的流程引擎能力。在此基础上提出一种主动式的工作流可靠性保障方法，根据流程引擎在最近一段时间间隔内的失效率和负载增长率两种特征参数主动预测其将来成功处理请求的概率，并据此将 VINCA 抽象工作流中的复合活动（实现为一个子流程）调度到“最可靠”的工作流引擎上。文章旨在从整体上提高工作流执行的成功率和稳定性，有效地避免基于“事后”被动恢复模式所带来的时间开销和实施上的复杂性。最后，通过场景示例作出了定性分析，表明该方法在大规模持续执行流程时，能充分利用工作流引擎能力，有效地保证工作流执行的可靠性。

关键词：网格工作流；可靠性；保障方法
中图分类号：TP311 **文献标识码**：A **文章编号**：0529-6579（2008）06-0093-07

近年来，网格工作流作为一种综合利用网格资源求解应用问题的“编程”技术得到越来越多的研究，由于其结合了工作流的协同能力与网格计算的集成能力和高效性，特别适合于构建跨组织、跨边界、跨学科的复杂问题求解环境，在科学计算领域已经得到了广泛的应用。

随着网格工作流应用范围的扩大，如何保障其可靠性成为急需解决的问题。已有工作大多采用重试、重调度、检查点恢复、冗余复制等容错机制保障工作流执行的可靠性[1]。比如，Taverna[2] 采用重试执行任务和重调度任务到其他资源的方式进行异常处理；工作［3］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[4] 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；工作［5－6］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[9] 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去调用相应的补偿活动。尽管上述方法的出发点和工作原理不尽相同，但他们都可归结为通过异常处理、故障恢复等手段进行被动地“事后”处理的容错方法，可能带来较大的时间开销和技术实施上的复杂性。

在中国国家网格 CNGrid（http：//www. cngrid. org/）环境中，存在着 BPEL、JSDL[10]、VINCA[11] 等多种类型的引擎，同一类型的引擎又可能有多份部署在多个网格节点上。我们曾尝试通过 VINCA 抽象工作流定义语言和一种工作流元调度机制（VINCA Meta-Workflow），为网格用户屏蔽底层资源细节并提供单一入口的、一体化的网格工作流虚拟执行环境的同时，优化利用已有的流程引擎能力和流程应用资源[7]。本文在已有工作［7］的基础上，主要研究如何提高 VINCA 抽象工作流执行时的可靠性。多份引擎的存在，使得我们可以将 VINCA 抽象工作流中的复合活动（实现为 BPEL 流程、JSDL 任务流或 VINCA 抽象子流程）优先调度到最有可能成功执行此活动的引擎上。然而，由于网格环境的复杂性和引擎状态的动态性，使得预测工作流是否可成功执行成为难题。本文借鉴传统的错误预测的基本模型和原理，选取最近一段时间间隔内引擎的失效率和负载增长率作为引擎运行状态的两种特征参数，并建立特征参数到引擎状态的映射关系，主动预测其将来成功处理请求的概率，并据此在“事前”将 VINCA 抽象流程中的复合活动

* 收稿日期：2008－09－10
基金项目：国家“863”高科技研究发展计划基金资助项目（2006AA12Z202）；国家自然科学基金资助项目（70673098）；中国科学院奥运科技基金资助项目（KACX1－03）
作者简介：张利永（1980年生），男，博士研究生；E-mail：zhangliyong@ software. ict. ac. cn

调度到最可靠的流程引擎上，从而从整体上提高流程执行的成功率和稳定性。有别于前述已有的相关工作，本文所提方法是一种主动式的以“错误预防”为主要手段的工作流可靠执行的保障方法。

本文以下第1部分简要介绍了VINCA Meta-Workflow的相关概念，并给出了本文所关注的可靠性的定义；第2部分详细叙述了基于主动预测的可靠性保障机制；第3部分给出了具有可靠性保障功能的VINCA Meta-Workflow系统架构；第4部分通过场景案例对此方法的效果进行了定性分析；最后，总结并展望了未来的工作。

1 CNGrid网格工作流介绍

1.1 元调度机制—VINCA Meta-Workflow

在文献［7］中，我们给出了一种优化利用已有工作流引擎能力和工作流应用资源的概念模型。该模型通过能力描述层和能力抽象层屏蔽底层工作流引擎和工作流应用资源的多样性和异构性，通过VINCA抽象工作流定义语言提供编排工作流应用资源和网格服务资源以构建复杂的网格应用的能力，从而为网格用户提供了业务级的工作流建模和执行环境。

图1给出了CNGrid中可用的工作流引擎的层次关系，其中，如文献［7］中所述，图中的“VINCA元引擎”中的“元”的是指VINCA元工作流不仅可以组合编排网格服务资源，而且更重要的还可以组合其他类型的工作流资源（如BPEL流程）或类工作流资源（如JSDL任务流）。

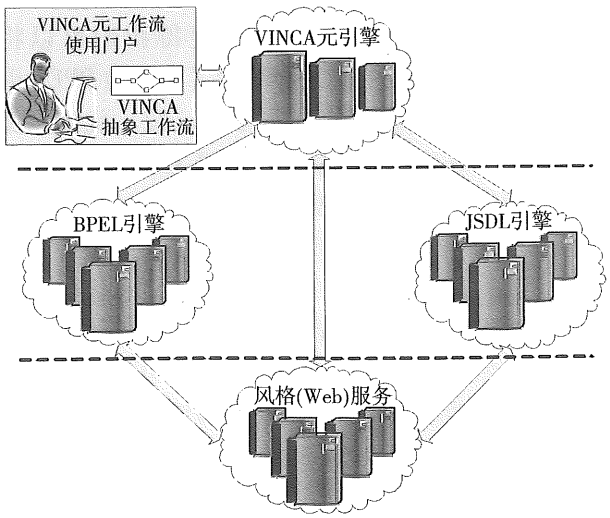

图1 CNGrid中工作流引擎层次关系

Fig. 1 Hierarchy of workflow engines in CNGrid

1.2 相关概念

（1）VINCA抽象工作流定义语言：是一种层次化的以业务服务[12]为基本建模元素的工作流定义语言，其基于XPDL[13]并扩展了XPDL中活动的定义类型。如图2所示，业务服务可分为复合活动和原子活动两类。执行时，复合活动可以具体绑定到BPEL流程或JSDL任务流，或者嵌套的定义为另一个VINCA抽象工作流；原子活动则绑定到具体的网格服务或Web服务。

（2）VINCA元引擎：对VINCA抽象工作流进行解析，绑定活动到具体实现流程或服务，通过调度底层的工作流引擎或直接调用服务提供执行VINCA抽象工作流的能力。需要说明的是，由于服务的自治性，本文只关注复合活动的可靠调度，所以，不适用于流程定义全部为原子活动的情况。

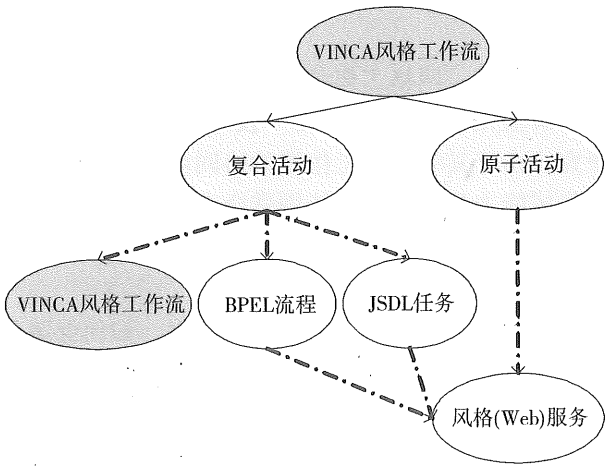

图2 VINCA抽象工作流模型层次关系

Fig. 2 VINCA abstract workflow model

1.3 可靠性概念界定

可靠性问题已在计算机科学的各个领域中得到广泛的深入的研究，网格系统的可靠性也得到越来越多的关注[8]。一般来说，“可靠性”（Dependability）是一个用来刻画系统属性的综合性概念，这些属性包括可依赖性（Reliability）、可用性（Availability）、防危性（Safety）、安全性（Security）、耐受性（Survivability）、一致性（Integrity）、可维护性（Maintainability）等[14]。

本文所说的可靠性只关注于网格工作流系统的可依赖性和可用性，网格工作流系统是可靠的是指其能通过一定的技术手段避免或减少工作流执行时错误的发生，提供稳定的、正确的执行工作流的能力，保证工作流的成功执行。

1.4 前提假设

网格工作流的执行需要使用多种分布的网格资

源，而网格管理域的存在可能使得某些工作流引擎无权访问所需的资源，不利于引擎能力的充分利用。本文对 CNGrid 环境进行简化，作出以下假设：

（1）网格工作流的执行是节点无关的，可以在任意节点的工作流引擎上执行，这要求活动的执行不使用某一网格节点上的特有资源，或者可以通过资源访问服务远程使用该资源；

（2）同一类型的工作流引擎其能力完全相同，以保证复合活动可调度到任一相应类型的引擎上执行；

（3）不考虑复合活动所对应的具体流程的规模的差异，假定执行复合活动所需的计算能力是相当的，以保证调度复合活动时只需考虑工作流引擎自身的属性，而与复合活动的实现无关；

（4）假设用户提交的工作流定义都是正确的、可被正常执行的，若执行时出现错误，则是由引擎或节点的原因造成的。

2　可靠性预测机制

2.1　基本原理

为了提高系统的可靠性，常用的方法大体可以分为错误预防（fault prevention）、错误预报（fault forecasting）、错误移除（fault removal）、容错（fault tolerance）4 类[14]。

其中前两种方法均属于错误出现前主动预防的可靠性保障模式，后两种方法则属于错误出现后被动修复的模式。本文结合前两种方法，根据流程引擎在最近一段时间间隔内的失效率和负载增长率两种特征参数主动预测其将来成功处理请求的概率（即 2.3 中定义的预知可靠度），并据此将 VINCA 抽象工作流中的复合活动调度到相应类型的“最可靠”的工作流引擎上。此处，“最可靠”的含义为相对于其他备选的流程引擎，此流程引擎最可能成功的执行完成复合活动。引擎失效率、负载增长率以及据此预测出的可靠度实际上是对引擎能力的实时建模和预测，既可以对错误进行“预警”，并据此调度复合活动，也可作为出错后进行重调度和流程调整的参考指标。本文只关注根据这些参数进行工作流活动的初始调度。

预测引擎将来可靠性的基本原理见图 3。

工作流引擎的特征空间 Y 由引擎的所有特征向量 $y(t_i)$ 组成，$y(t_i)$ 为在时刻 t_i 引擎的一组特征参数的监测值；引擎的状态空间 S 由引擎的状态向量 $s(t_i)$ 组成，$s(t_i)$ 为在时刻 t_i 引擎的一组状态参数的监测值。特征空间与状态空间之间存在以下两种映射关系：

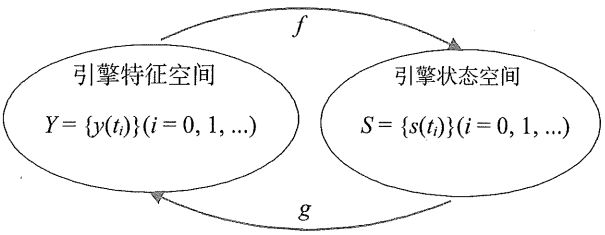

图 3　可靠性预测基本原理

Fig. 3　Principles of dependability predication

（1）f：$Y \rightarrow S$，定义了特征空间 Y 到状态空间 S 的一对一映射，表示根据工作流引擎在时刻 t_i 的特征向量 $y(t_i)$ 可唯一计算出此时引擎的状态向量 $s(t_i)$；

（2）g：$S \rightarrow Y$，g 定义了状态空间 S 到特征空间 Y 的一对一映射，表示引擎处于 $s(t_i)$ 所表示的状态时可监测到唯一一组特征值 $y(t_i)$；

本文中，引擎的特征向量由引擎在时刻 t_i 的失效率和负载增长率组成，引擎的状态则只关注其在时刻 t_i 的可靠度，通过 f 映射可建立引擎可靠度主动预测方法，下面给出这几个特征参数和状态参数的详细定义。

2.2　特征参数定义

定义 1：$R(t-\Delta t, t)$ 为由时刻 $t-\Delta t$ 到时刻 t 的 Δt 时间间隔内引擎收到的执行流程的请求数；

定义 2：$S(t-\Delta t, t)$ 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；

定义 3：$E(t-\Delta t, t)$ 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；

定义 4：

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

为由时刻 $t-\Delta t$ 到时刻 t 的 Δt 时间间隔内引擎的失效率；

定义 5：

$$\lambda(t-\Delta t,t) = \frac{R(t-\Delta t,t) - S(t-\Delta t,t) - E(t-\Delta t,t)}{\Delta t}$$

为由时刻 $t-\Delta t$ 到时刻 t 的 Δt 时间间隔内引擎的负载增长率；

定义 6：假设引擎收到第一个请求的时刻记为 t_0，则在时刻 t_n 引擎处于空闲状态是指在时刻 t_n 满足下面关系：

$$R(t_0,t_n) = S(t_0,t_n) + E(t_0,t_n)$$

即，在时刻 t_n，引擎收到的请求或者被成功处理，或者执行时发生了错误而被终止。另外，引擎处于

空闲状态也可等价的定义为在时刻 t_n 满足下面关系：

$$\sum_{i=1}^{n}\lambda(t_{i-1},t_i)=0$$

其中，$t_i(1\leqslant i\leqslant n-1)$ 为介于 t_0 到 t_n 的时刻点。

以上定义均从某一侧面反映了引擎的运行状态。直观来看，引擎处于空闲状态表示引擎的当前负载为0，一般而言，此时最适于将流程执行请求调度到此引擎上，除非该引擎的失效率大于某个不可接受的阈值。下面第3部分的调度算法中首先对此种情况进行了处理。

$\mu(t-\Delta t,t)$ 直接反映了在时刻 t 之前的 Δt 时间间隔内引擎执行流程的可靠程度，$\mu(t-\Delta t,t)$ 只可能取非负数值。大体上，引擎的可成功执行流程的概率应与 $\mu(t-\Delta t,t)$ 成反比关系。

$\lambda(t-\Delta t,t)$ 为引擎在 $t-\Delta t$ 到 t 的 Δt 时间间隔内引擎的负载变化，由于工作流执行需要耗费一定的时间，则此参数可以间接地反映相对于当前时刻 t，引擎在将来一段时间内的负载情况，其取值可能为以下3种情况：

(1) $\lambda(t-\Delta t,t)>0$，表示 Δt 时间段内引擎收到的请求数多于引擎完成的请求数，则相对于时刻 $t-\Delta t$，引擎的负载处于上升趋势，此时不宜将新的复合活动调度到此引擎上执行；

(2) $\lambda(t-\Delta t,t)=0$，表示 Δt 时间段内引擎收到的请求数等于引擎完成的请求数，则相对于时刻 $t-\Delta t$，引擎的负载处于稳定状态，此时可以将新的复合活动调度到此引擎上执行；

(3) $\lambda(t-\Delta t,t)<0$，表示 Δt 时间段内引擎收到的请求数小于引擎完成的请求数，则相对于时刻 $t-\Delta t$，引擎的负载处于下降趋势，此时将新的复合活动调度到此引擎上执行是适合的；

总之，$\lambda(t-\Delta t,t)$ 取值越小，一定程度上表示将请求调度到此引擎上执行时可能因负载过重而导致引擎崩溃失效的概率越小，从而流程应用成功执行的可能性越大。因此，将 $\lambda(t-\Delta t,t)$ 作为预测引擎将来一段时间内可靠度的评估参数是合适的。

综合考虑 $\mu(t-\Delta t,t)$ 和 $\lambda(t-\Delta t,t)$，可以给出预测引擎可靠度的映射关系 f，下面给出详细定义。

2.3 状态参数定义

定义7：

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

$(0\leqslant\alpha\leqslant1)$

为由时刻 $t-\Delta t$ 到时刻 t 的 Δt 时间间隔内引擎的参考可靠度。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。

Δt 为预测引擎将来的可靠度所依赖的引擎运行历史的参考时间间隔，其取值直接影响预测算法的精确度和可信度。如果 Δt 取值过小，$\mu(t-\Delta t,t)$ 和 $\lambda(t-\Delta t,t)$ 可能较多的体现引擎的偶然特征，使得计算得到的可靠度的可信性降低。而如果 Δt 取值过大，则 $\mu(t-\Delta t,t)$ 和 $\lambda(t-\Delta t,t)$ 不能实时地反映引擎的当前状态特征，使得 $\rho(t)$ 缺乏时效性。所以，Δt 取值需要综合考虑这两种极端情况对预测引擎可靠度造成的影响，不宜取固定的常量值。本文给出基于流程平均执行时间的 Δt 取值策略，及引擎在时刻 t 的可靠度的计算方法。

实际应用中，复合活动对应的具体流程的规模大小各异，执行时间也长短不一。根据引擎运行历史可以计算出流程的平均执行时间，表示为 $\bar{t}$，则 Δt 可分别取值为 $\Delta t_n=n\cdot\bar{t}(n\geqslant1)$，据此，进一步给出如下定义：

定义8：

$$\rho(t)=\frac{1}{n}\sum_{i=1}^{n}\rho(t-\Delta t_i,t)$$

为引擎在时刻 t 的预知可靠度，即，计算时刻 t 之前的多个时间间隔内的可靠度 $\rho(t,\Delta t)$，求均值作为引擎的在时刻 t 的可靠度，同时也作为引擎在时刻 t 开始的一段时间内可成功执行流程的概率。

需要说明的是，由于采用了基于历史统计的方法预测引擎将来的可靠度，这就需要在时间间隔 Δt 内有足够多的参考样本，样本量越大，预测的效果越好。所以，此方法更适合于在大规模并发执行流程情况下保障网格工作流系统的可靠性。

3 系统设计与实现

3.1 系统架构

以前述方法为指导，我们设计了具有可靠性保障机制的 VINCA Meta-Workflow 系统，其整体架构如图4所示。

由图4可见，该系统可分为4层，分别为：

(1) 系统门户：提供基于 B/S 结构的网格工作流使用环境，是网格用户提交、执行流程应用的直接入口。通过配置，系统门户可以连接到任意一个后台的 VINCA 元引擎；

(2) VINCA 元引擎：提供解析、调度、执行 VINCA 抽象流程的能力。其内部除了传统的流程解析、运行监控、数据管理和持久化等功能模块外，还增加了以下可靠性保障模块：

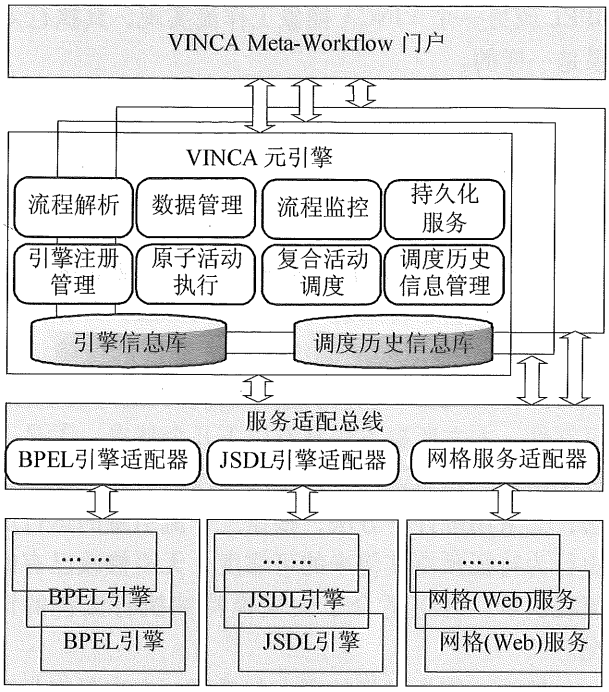

图 4　VINCA Meta-Workflow 系统架构

Fig. 4　Architecture of VINCA Meta-Workflow

a）引擎注册管理：通过持久化的引擎信息库，管理 CNGrid 网格环境中的各种工作流引擎的注册信息；

b）调度历史信息管理：维护复合活动的调度和执行信息，为前述的可靠度预测算法提供所需的底层引擎的统计数据；

c）复合活动调度：根据复合活动所对应的具体流程的类型，根据前述算法计算备选工作流引擎的可靠度，并据此将复合活动对应的具体流程调度到“最可靠”的工作流引擎上；

（3）服务适配总线：通过引擎适配器和网格服务或 Web 服务适配器，对上为 VINCA 元引擎提供统一的调用底层工作流引擎的接口，对下接入管理各种工作流引擎和网格服务或 Web 服务；

（4）引擎资源及网格（Web）服务：包括 BPEL 引擎、JSDL 引擎、网格服务和 Web 服务。

3.2　基于引擎可靠度的调度策略

VINCA 抽象工作流执行过程中，复合活动的调度策略是本文关注的重点。下面给出 VINCA 元引擎实现中处理流程活动的步骤：

（1）如果活动为原子活动，查找、调用绑定的网格服务或 Web 服务，结束返回；否则，

（2）活动为复合活动，查找该复合活动绑定的具体流程 P；

（3）根据流程 P 的类型，查找当前可用的备选的工作流引擎，假设得到的引擎集合为 ES；

（4）如果 ES 为空集合，出错返回；否则，

（5）根据工作流执行的历史信息，计算流程平均执行时间，并据此选取特定的几个时间间隔；根据经验值，设置前述定义 7 中的 α 取值；

（6）对集合 ES 中的每一个引擎，按照第 2 部分的定义分别计算其在选定的时间间隔内的失效率、负载增长率及预知可靠度；

（7）首先根据引擎是否处于空闲状态进行调度：如果 ES 中只有一个引擎处于空闲状态，且（6）中计算出的失效率小于最大可容忍的失效率（此值亦为经验值），则该引擎即为目标调度引擎 TE；否则，

（8）选取 ES 中预知可靠度值最大的引擎作为目标调度引擎 TE；

（9）新建 TE 的调度历史记录 TE，调度复合活动对应的流程到 TE，并触发目标引擎执行该流程；

（10）如果 TE 成功执行了流程，更新 TE 的成功执行流程的次数；否则，更新 TE 的失败执行流程的次数；

（11）当前活动处理完毕，按照上述步骤处理下一个活动。

4　场景示例及分析

本文所提出的基于引擎可靠度预测的工作流可靠性保障方法作为初步的尝试在 VINCA Meta-Workflow 系统的开发实施中尚处于起步阶段，本节以一个简单的流程为例定性地说明应用该方法后对工作流执行可靠性的影响。

在图 5 所示的示例场景中，VINCA 抽象工作流 P 中定义了 2 个并发活动 A1、A2，它们在执行时均被绑定到 JSDL 任务流。假设有 3 个可用的 JSDL 引擎，在初始时刻 t_0 它们均处于“空闲状态”。为简化计算，只以 t_0 作为样本时间间隔的开始，计算定义 8 中的引擎的可靠度。假设 VINCA 元引擎并发的执行 3 次流程 P，则会创建 6 个 JSDL 任务。

对下面几种情况的工作流执行的可靠性分别进行考察：

（1）假设 JSDL 引擎均只能成功执行 A1 或 A2，即引擎失效率为 0，且 3 个 JSDL 引擎执行 A1、A2 所需时间相同，忽略调度算法自身的时间开销。此时，应用前述的调度策略，可得到 6 个 JSDL 任务最终的调度结果为：

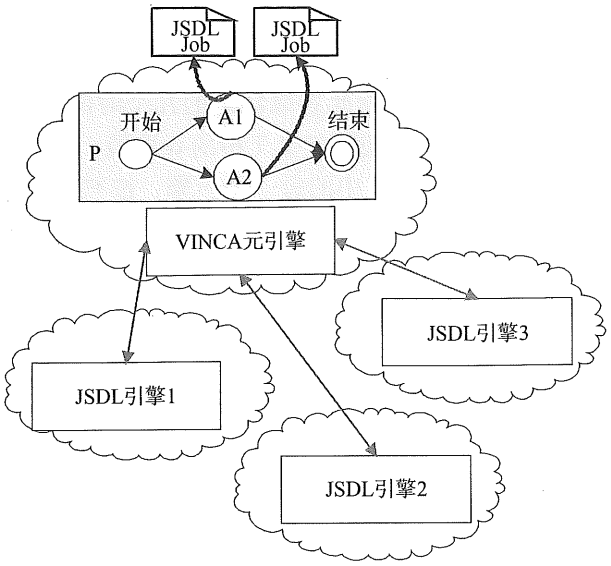

图5 示例场景

Fig. 5 An example scienario

P1	目标	P2	目标	P3	目标
A1	引擎1	A1	引擎3	A1	引擎2
A2	引擎2	A2	引擎1	A2	引擎3

（2）有别于（1），假设引擎3所在节点计算能力远远超过其他两个节点，其执行A1、A2的时间也小于其他两个引擎，可以假设启动第4个流程时引擎3上的P2_A1已经完成，按照算法，此时会将P4_A1（或P4_A2）调度到引擎3上。

（3）考虑JSDL引擎执行任务时可能出错的情况，假设按照（1）中调度结果，引擎1在执行P1_A1任务时出现错误，而引擎2和引擎3均没发生错误，假设启动第4个流程时每个引擎上均尚有一个任务未完成，根据可靠度计算公式，由于引擎1的失效率>0，使得其可靠度降低，故P4的任务会优先调度到引擎2和引擎3上；

对以上的结果进行分析，由（1）、（2）可以看出，在不考虑引擎出错的情况下，调度算法总是选择负载增长率最小的引擎，这表明，文中所提的调度算法能够充分利用所有的引擎资源，避免个别引擎负载过重导致系统崩溃，从而有利于提高流程应用执行的可靠性。由（3）可以看出，文中的调度算法可以避免将任务调度到失效率较高的引擎上，最大程度的提高任务执行的成功率，从而提高流程应用执行的可靠性。

值得说明的是，尽管上述示例较为简单，由于调度算法只针对一个复合活动，而与流程中的其他活动无关，故示例能真实反应实际应用时的效果。另外，将示例中的复合活动由JSDL实现替换为由BPEL或另一个VINCA抽象工作流实现，其执行效果是一样的。

5 总结及展望

本文针对CNGrid中的多工作流引擎环境，借鉴传统的错误预测模型和原理，通过对一段时间内引擎失效率和负载增长率两种特征变量的统计分析，建立了引擎将来的可靠度预测算法，并据此进行VINCA复合活动的调度，最大程度的保障活动成功执行的概率，从而从整体上提高工作流执行的可靠性。本文所提方法具有以下几个特点：①是一种主动的、以错误预防为主要手段的可靠性保障方法；②采用统计学方法，根据工作流引擎的运行历史行为特征预测其将来的可靠度，无需修改已有的流程引擎；③更适合于大规模持续执行流程的情况。

然而，作为初步的探索与尝试，文中工作尚有许多有待完善和改进之处。未来的工作包括：①除引擎失效率与负载增长率之外，建立和引入其他引擎特征参数；②完善可靠度计算公式；③建立综合网格节点能力、复合活动规模与引擎可靠度的任务调度算法；④建立引擎可靠性的测试模型和测试环境。

参考文献：

[1] YU J, BUYYA R. A taxonomy of workflow management systems for Grid computing [J]. Journal of Grid Computing, 2005, 3 (3): 171 - 200.

[2] Taverna User Manual. http://www.mygrid.org.uk/user-manual1.7[Z], 2008.

[3] AN W, FONG L, BOBROFF N. BPEL4Job: A fault-handling design for job flow management[C]//Proceedings of the 5th International Conference on Service Oriented Computing (ICSOC'07). Vienna, Austria, 2007: 27 - 42.

[4] COUVARES P, KOSAR T, ROY A, et al. Workflow management in condor[M]. Workflows for e-Science. 2007: 357 - 375.

[5] ABAWAJY J H. Fault-tolerant scheduling policy for grid computing systems[C]//Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS'04). Santa Fe, New Mexico, USA, 2004: 238 - 244.

[6] HWGNG S, KESSELMAN C. Grid Workflow: A flexible failure handling framework for the grid[C]//Proceedings of the 12th IEEE International Symposium on high Performance Distributed Computing (HPDC'03). Seattle, Washington, USA, 2003: 126 - 137.

[7] ZHANG L Y, ZHAO Z F, LI H F. Leveraging legacy workflow capabilities in a Grid environment[C]//Proceedings of the 6th International Conference on Grid and Cooperative Computing (GCC'07), Urumchi, Xinjiang, China, 2007: 361 - 365.

[8] TOWNEND P, XU J. Dependability in Grids[J]. IEEE Distributed Systems Online, 2005, 6(12):1 - 1

[9] ANDREWS T, CURBERA F, DHOLAKIA H, et al. Business process execution language for web services version 1.1[M/OL]. http://www.ibm.com/developerworks/library/specification/ws-bpel, 2005

[10] ANJOMSHOAA A, BRISARD F, DRESCHER M, et al. Job submission description language (JSDL) specific, Version 1.0 [Z],2005.

[11] HAN Y, GENG H, LI H, et al. VINCA-A Visual and personalized business-level composition language for chaining web-based services[C]//Proceedings of the 1st international conference on service-oriented computing (ICSOC'03). Trento, Italy, 2003: 165 - 177.

[12] WANG J W, YU J, HAN Y. A Service modeling approach with business-level Reusability and extensibility [C]//Proceedings of the IEEE International Workshop on Service-Oriented System Engineering (SOSE'2005). Beijing, 2005:23 - 28.

[13] Workflow Management Coalition. Process definition interface-XML process definition language version 1.15 [OL]. http://www.wfmc.org/standards/xpdl.htm, 2005.

[14] AVIZIENIS A, LAPRIE J C, RANDELL B. Fundamental concepts of dependability[C]//Proceedings of the 3rd IEEE Information Survivability Workshop (ISW'2000), Boston, Massachusetts, USA, 2000: 7 - 12.

A Proactive Approach to Ensure Dependability of Grid Workflows

ZHANG Li-yong[1,2], *HAN Yan-bo*[1]

(1. Research Centre for Grid and Service Computing, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China

2. Graduate University of Chinese Academy of Sciences, Beijing 100039, China)

Abstract: As a special kind of "programming" technology for constructing problem-solving applications on the basis of grid resources, grid workflow has attracted attention and made progress. However, how to ensure dependability of grid workflows is still a remaining challenge. For China National Grid (CNGrid), a meta-scheduling mechanism (VINCA Abstract Workflow) was proposed, by which the underling capability of workflow engines can be optimally used and a single entrance can be provided while details are hidden. A proactive approach to ensure dependability of grid workflows is proposed in this paper. The workflow engine's failure rate and workload increasing rate in a certain recent interval are calculated for predicting the probability of successfully handling current execution request. The compound activity (implemented as a sub-process) in a VINCA abstract workflow is scheduled to the most promising engine. In this way, the overall dependability of workflow execution can be enhanced while avoiding time cost and technical complexity caused by the traditional "reactive" rescue approaches. The approach is qualitatively analyzed by an example scenario, which indicates that the approach can ensure the dependability of workflow by fully utilizing the engines' capability when executing workflows cosmically and continuously.

Key words: grid workflow; dependability; ensuring approaches