工作流失败恢复的研究

张 猛[1],侯秀萍[1],王 烨[2],赵云峰[3]
(1.长春工业大学 计算机科学与工程学院,吉林 长春 130012;
2.长春工业大学 计算机信息网络中心,吉林 长春 130012;
3.中国人民解放军装甲兵技术学院 控制系,吉林 长春 130117)

摘 要:工作流在某一点发生错误,导致工作流运行失败。通过调用失败路径上节点的补偿逻辑,以便将工作流恢复到先前的一个上下文环境。因为补偿节点是相当复杂和耗时的工作,减少工作流补偿节点的数量可大大提高工作流的补偿效率。基于工作流的数据和流程定义者所提供信息,文中提出了多种补偿范围,以便在不同条件下有效地减少补偿节点的数量。

关键词:工作流;补偿;依赖
中图分类号:TP311 **文献标识码**:A **文章编号**:1673-629X(2009)04-0065-04

Research on Workflow Failure Recovery

ZHANG Meng[1], HOU Xiu-ping[1], WANG Ye[2], ZHAO Yun-feng[3]
(1. School of Computer Science and Engineering, Changchun University of Technology, Changchun 130012, China;
2. Computer Information Network Center, Changchun University of Technology, Changchun 130012, China;
3. Department of Controlling Engineering, Armor Technique Institute of PLA, Changchun 130117, China)

Abstract: Workflow at some point error occurred, resulting in workflow failure. By recovery the failure of nodes on the path of compensation, workflow returns to a previous context environment. Because the compensation node is a very complex and time-consuming work, workflow to reduce the number of nodes compensation can greatly improve workflow efficiency of the compensation. Based on a detailed analysis of dependencies history information and process designers to provide certain knowledge about workflow processes, presents a wide range of compensation, under different conditions in order to effectively reduce the amount of compensation nodes.

Key words: workflow; compensation; dependencies

0 引 言

工作流是一个年轻却又具有良好发展前景的研究方向。它源于计算机软件的商业应用,是一个多学科交叉的新领域,涉及计算机科学与管理科学的多种原则、方法与技术。它巨大的潜在市场和广阔的应用领域吸引了来自全世界研究机构及软件厂商越来越多的关注[1]。

经过二十多年的研究发展,工作流技术取得了很大的发展并得到了日益广泛的应用。随着网络的普及,Internet的快速发展,工作流系统的外部环境发生了巨大的改变,原来集中式的、同质的、面向桌面的环境逐渐朝着分布的、异质的、面向网络的方向发展。随着环境的复杂化,对工作流稳定性、可靠性的要求更加强烈。工作流失败恢复已经成为设计与编制高可靠软件系统的重要的方法[2~4]。

工作流失败恢复:当工作流在某一点发生错误,导致工作流运行失败。通过调用失败路径上节点的补偿逻辑,将工作流恢复到先前的一个上下文环境,使工作流基于先前的上下文环境继续向下执行[5]。

近几年来,人们提出的几个工作流补偿方式,都可以称作一种完整补偿方式[6]。在大多情况下,这些方式是从数据库观点发展来的,数据库的操作都是一些简单的磁盘读写操作,采用完整补偿方式对系统性能的影响并不大。工作流通常依赖的不只是数据库系

收稿日期:2008-07-18
基金项目:吉林省科技发展计划重大科技攻关项目(吉科合字20040305)
作者简介:张 猛(1981-),男,硕士,研究方向为软件工程;侯秀萍,硕士,教授,研究方向为软件工程。

统,还有应用系统和人工干预。各个活动节点都包含了复杂的逻辑控制,而不是简单的数据读写操作[7]。所以,工作流节点的恢复是相当复杂和耗时的。

定制工作流补偿范围,以便减少不必要的补偿节点,是笔者针对这个问题提出的解决方案。

1 一些关于工作流的描述

定义 1 工作流。

工作流是由点的有穷非空集合和点之间边的集合组成,表示为:$P = (N, A, D)$,其中,P 表示一个工作流;N 是工作流中点的集合:N 中包括两种点:工作节点 W(执行任务);条件节点(控制流程走向的)C;A 是 P 中边的集合;D 是工作流数据的集合[8]。

定义 2 流程数据 $D(P)$。

$D(P) = \{SD(P) \cup PD(P) \cup AD(P)\}$

(1) 进程专用数据 $SD(P)$:用于工作流内部。

(2) 进程关系数据 $PD(P)$:用于工作流内部和工作流外部应用程序。

(3) 应用专用数据 $AD(P)$:用于工作流外部应用程序,对于工作流内部是不可见的。

定义 3 节点依赖。

通常,先后执行的两个节点有两种依赖关系。

第一种叫执行依赖,如图 1 所示。

图 1 执行依赖

前一个节点只是决定流程经过后面那一个节点。

定义 4 如果 $N_i, N_k \in N$, $End(N_i) < Start(N_k)$①, $N_i, N_k \in R(P)$(表示在同一条执行路径上),用 $N_i \xrightarrow{ed} N_k$ 表示执行依赖。

第二种叫数据依赖,如图 2 所示。

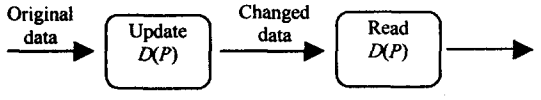

图 2 数据依赖

后面的节点要读取前一个节点改变后的流程数据。它们在流程补偿并重新执行后都会产生不同结果。

定义 5 $N_i, N_k \in N$, $End(N_i) < Start(N_k)$, N_k 读取了 N_i 所改写的数据,用 $N_i \xrightarrow{dd} N_k$ 表示数据依赖。

两种依赖关系都满足用 $N_i \xrightarrow{*} N_k$ 表示。

①$End(W_i)$ 表示 W_i 执行的完成时间;$Start(W_i)$ 表示 W_i 执行的开始时间。

2 补偿范围设计与分析

工作流节点的恢复是相当复杂和耗时的,如何通过一定的筛选原则,去除一些并不需要补偿的节点,以便减少补偿节点的数量,进而提高补偿效率。

筛选原则:

(1)如果节点之间存在的是执行依赖关系,那么前一个节点的执行只是决定流程走向,并不会改变工作流中的数据或将自身的数据持久化,而影响工作流中的其它节点,所以这样的节点是不需要补偿的。

(2)当一个工作流中存在着同步工作流分支时,一条路径中的节点与另一条路径的节点不存在数据依赖关系时,只需补偿存在保存点的那条路径。但如果它们之间存在数据依赖关系,它们便均需要补偿。

(3)流程定制者要提供必要的工作流信息:工作流通常依赖的不只是数据库系统,还有应用系统和人工干预。所以定制工作流范围所需要的不仅仅是工作流内部的信息,而且还需要流程定制者提供必要的工作流信息。工作流在进行补偿时可以通过这些信息来筛选补偿节点。同时对于一些存在于同一个工作流中的同步工作流,如果可以事先确定它们之间并不存在数据依赖关系,可以更进一步简化补偿算法,提高补偿效率。

(4)所有的定制的补偿范围均要包含与保存节点存在数据依赖关系的节点。

依据上述原则文中提出如下补偿范围。

2.1 补偿范围

2.1.1 瞬时补偿范围

瞬时补偿范围包含的是在保存节点之后,所有之前启动的节点。瞬时补偿范围只是依赖于流程结构和先前的执行过程,这样就很容易通过搜索日志来决定补偿范围,既保障了可计算性又保证了补偿完整性,但它的缺点是补偿了不必要节点。

定义 6 瞬时补偿范围。

$TCS(N_i) = \{N_k \in N \mid End(N_i) < Start(N_k)\}$

2.1.2 无条件的补偿范围

无条件的补偿范围只补偿于过去执行的历史数据中的工作节点,不补偿条件节点。

定义 7 无条件的补偿范围。

$NCS(N_i) = \{W_k \in W \mid End(N_i) < Start(N_k)\}$

无条件的补偿范围依据的是筛选原则(1),去除了所有条件节点。

2.1.3 完整补偿范围

完整补偿范围补偿所有直接或间接被最终的保存节点所影响的活动实例。

定义 8 完整补偿范围。

$ACS(N_i) = \{W_k \in W \mid End(N_i) < Start(N_k), W_i \xrightarrow{*} W_k\}$

完整补偿范围依据的是筛选原则(1)和(2),去除了所有条件节点。也避免补偿一些与保存点不存在数据依赖关系的节点。

2.1.4　可计算的补偿范围

可计算的补偿范围依赖于人为给出的一些信息,这些信息存在于SD(P)和PD(P)之中。工作流根据这些流程信息来判断是否补偿节点。

定义9　可计算的节点。

$Computable(W_i) = \{W_i \in W \mid W_i \in application(W), W_i \in HumanInterfere(W)\}$

定义10　可计算的补偿范围。

$CCS(N_i) = \{W_k, W_j \in W \mid W_i \xrightarrow{*} W_k, End(N_i) < Start(W_k, W_j), W_i \notin Computable(W), W_j \in Computable(W)\}$

可计算补偿范围依据的是筛选原则(3),由于确定应用节点是否应该补偿主要依据的是进程专用数据,这对工作流来说是不可见的。如果流程定制者事先给出一定的信息,使工作流能够根据这些信息自行判断应用节点是否需要补偿,便能减少补偿节点的数量。

2.1.5　有限补偿范围

有限补偿范围依赖于工作流结构和工作流运行的特定时间。有限补偿范围是比较容易计算的,但流程定制者必须确信没有其它的节点与有限补偿范围内的点有数据依赖关系[9]。

定义11　有限补偿范围。

$FCS(N_i)\{W_k \in W \mid \exists W_j \in W, W_k \in R(P_a), W_j \in R(P_b), N_i \xrightarrow{*} W_k, \neg \exists W_k \xrightarrow{*} (\forall W_s \in R(P_a)), End(N_i) < Start(W_k, W_j), N_i \notin R(P_b), W_j \in HumanInterfere(R(P_b))\}$

有限补偿算法依据的是筛选原则(2)、(3)。由于流程定制者事先给出的信息,大大地减小了待补偿的范围,所以更进一步提高了补偿效率。

2.1.6　最小补偿范围

最小补偿范围依赖于过去执行的历史数据和人为给出的一些信息,包括PD(P)和AD(P)。例如:$W_2 \in N$工作流补偿节点在W_2,基于W_2节点的最小补偿范围表示为MCS(N_2),它是W的子集包括所有符合$W_2 \xrightarrow{dd} W_k$定义的$W_k \in R(P)$的集合。

定义12　最小补偿范围。

$MCS(N_i) = \{W_k \in W \mid W_i \xrightarrow{dd} W_k, End(W_i) < Start(W_k), W_k \in CCS(N_i)\}$

最小补偿范围依据的是筛选原则(1)、(2)、(3),它只补偿了必要补偿节点。从补偿节点数量上来说它是最少的。

2.2　补偿接口的设计与实现

并不是所有的活动都是可补偿的,在某些情况下,活动也许不可能用有合理的补偿逻辑。那Wait活动来说,它仅仅是等待计时器的触发。在计时器触发之后,Wait活动也就完成了,没有任何办法可以改变计时器已经触发了这个事实。而在另一些情况下,人们并不希望使用补偿机制[6]。

所以补偿要作为一个可选项出现,为给活动添加可补偿性,活动必须实现ICompensatableActivity接口:

```
public interface ICompensatableActivity{
  ActivityExecutionStatus Compensate (ActivityExecutionContext context, HumanInterfere hi);
}
```

ICompensatableActivity接口定义了一个方法,即Compensate。一个活动可以在Compensate方法中完成补偿逻辑,并返回ActivityExecutionStatus.Closed。

例:带有补偿逻辑的SendEmail活动

```
public class SendMail implement ICompensatableActivity {
  protected override ActivityExecutionStatus Execute(ActivityExecutionContext context){
    MailService mailService = context.GetService<MailService>();
    MailService.Send(To,From,Subject,Body);
    Return ActivityExecutionStatus.Closed;
    }
  ActivityExecutionStatus Compensate (ActivityExecutionContext context,HumanInterfere hi){
    MailService mailService = context.GetService<MailService>();
    MailService.Send(To,From,hi.getInterfere());
    Return ActivityExecutionStatus.Closed;
  }
}
```

SendEmail活动的Compensate方法,根据人工干预信息和定义在Send方法中的补偿方法,简单地发送了第二封邮件来通知对方原来的那封邮件需要被忽略,并返回ActivityExecutionStatus.Closed。

在WF程序的执行过程中,活动自动机掌握着所有活动的生命周期,并控制着状态的转移。WF运行时处理从一个状态到另一个状态的转移过程,以确保WF程序执行的正确性。如图3所示,通常,一个活动的启始状态为Initialized,一旦开始工作,状态就转移到Executing,当活动节点执行成功或补偿成功,就进

入 Closed 状态，当活动节点执行失败，就进入 Faulting 状态，如需要补偿，转移到 Compensation 状态，如不需要补偿，转移到 Closed 状态。

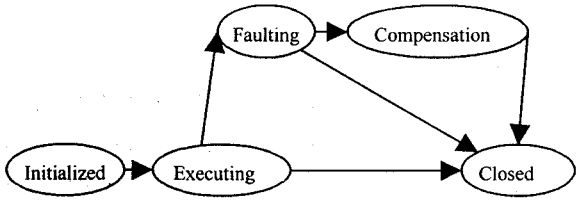

图 3 活动自动机

2.3 动态选择补偿范围算法

根据文中所提出的补偿范围和筛选原则，提出如下算法：

```
ConfirmCompensateScope(Wi){
  Lookup PostNode;
  Lookup PostRunnedNode;
    if(! isHaveHumanInterfere){
      if(! isHaveBranch){
        TCS(Wi);//Temporal Compensation Scope
        }else{
        if(PostRunnedNode∈OnlyOneRunRoute){
        NCS(Wi); //No- condition Compensation Scope
        }else{
        ACS(Wi); //All Compensation Scope
        }
        }else{
      if(isHaveBranch){
        if(HumanInterfere.isHaveDataDepend){
        FCS(Wi);//Finite Compensation Scope
        }else{
        MCS(Wi);//Minimal Compensation Scope
      }
        }else{
        CCS(Wi);//Computable Compensation Scope
        }
      }/* End of ConfirmCompensateScope (Wi)/*
```

2.4 补偿范围比较与分析

图 4 为网上购物工作流。考虑 $R(P)=\{1,2,3,4,5,6,7,8,9,10\}$ 在 1 处保存上下文，在 7 处失败。利用这个流程对上述的补偿范围进行验证与分析得到表 1。

它们之间存在如下关系：

$MCS(N_i)\subset FCS(N_i)\subset CCS(N_i)\subset ACS(N_i)\subset NCS(N_i)\subset TCS(N_i)$

最小范围补偿从补偿节点数量上看是最优的，同可计算补偿范围策略，有限补偿范围一样需要人工干预。瞬时补偿范围和无条件补偿范围补偿方式是比较好实现的，但它补偿的节点也是最多的会影响补偿效率。虽然 MCS(1)补偿路径更高效，但它的算法是最复杂的，在一些特定场景它所消耗的资源可能远远大于完整补偿范围，无条件的补偿范围。

表 1 各种补偿范围的比较

名称	补偿节点	数量	完整	复杂度	人工干预
瞬时	1,2,3,4,5,6,7,8,9,10	10	是	简单	无
无条件	1,2,3,4,5,6,8,9	8	是	简单	无
完整	1,2,5,6,8,9	6	是	较简单	无
可计算	1,2,5,6,8,9	6	是	一般	有
有限	1,2,8,9	4	否	一般	无
	1,2,5,6,8,9	6	是		有
最小	2,5,6,8,9	5	是	复杂	有

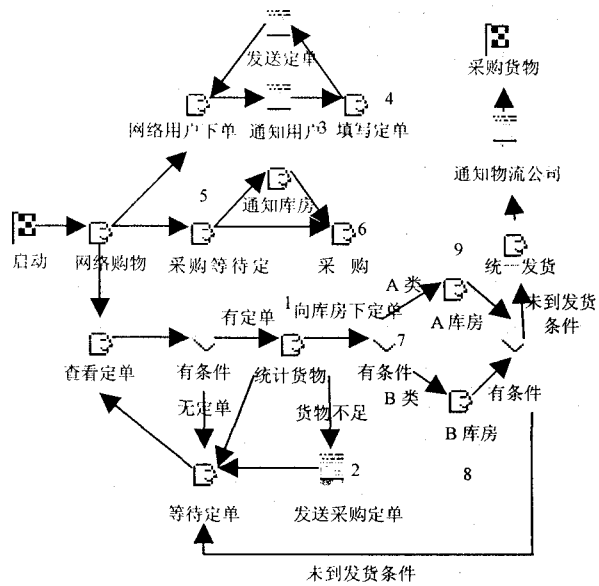

图 4 网上购物工作流

所以要根据工作流的复杂度，补偿节点的数量的多少，具体实现的难易程度，在补偿性能与算法实现的简易性上进行折中。

3 结束语

文中所设计的几种补偿范围已经在启明 ERP 工作流系统进行了尝试，并取得了良好的效果。接下来的研究重点是如何设计一个补偿引擎，动态计算各种补偿策略在补偿时刻所将要消耗的时间和资源，以便调用最合适的补偿策略。

参考文献：

[1] 崔永花，谭庆平，杨艳萍. 工作流中的长事务模型[J]. 计算机工程与科学，2006，28(4)：87－89.

[2] 王 睿，李从心. 事务性工作流中的长事务处理方法研究[J]. 计算机工程，2007，33(4)：29－31.

(下转第 187 页)

处理建议;下方为该图像的基本信息内容,是采集图像在远程提交时上传的相关信息,该部分内容保存在信息数据库的图像信息表中。并且,诊断结果还可以以报表形式导出。

针对电力系统实际运营中的设备故障检测问题,本系统可以起到很好的辅助作用,最大限度地帮助操作人员完成设备检修。此外,本系统还具有以下优点:

(1)运用数字图像处理中的特征提取和经典分割算法获取图像中的目标区域,得到目标的边缘信息以及目标区域信息,并把基于内容的图像检索技术应用于共享数据库的检索中。

(2)红外热像仪输出结果的解读与控制。红外热像仪大都输出标准格式的数字图像(如 jpg 格式),通过数字图像处理中图像格式的理论,对红外热像仪输出的标准格式的图像进行分析处理,解读出图像中每一个像素的温度数据以及相关参数信息。

(3)采用 B/S 和 C/S 结合的多层结构,实现最大限度的发挥不同构架的不同优势。

(4)系统采用模块化开发,具有良好的可靠性和扩展性,便于系统的维护和二次开发。

5　结束语

所开发的带电设备红外辅助诊断系统集现场预检、远程提交、故障诊断、图像库管理为一体,不仅可以使采集到的红外图像信息及时传送,实现对设备的及时诊断,而且可以使诊断结果再次利用。该系统能及时检测出带电设备的事故隐患,能有效地降低事故发生率,从而提高设备运营的安全性和可靠性。可见,该系统可以广泛应用于在电力、冶金、石化等工业领域。但是由于系统缺乏完备的典型故障图像库,在故障诊断时,缺少相应的典型故障图像作参考,所以不利于为操作人员提供直观的对比分析,这是进一步要解决的问题。

参考文献:

[1] 田裕鹏.红外检测与诊断技术[M].北京:化学工业出版社,2006.

[2] 叶　风,张晓宁.红外技术在电力设备外部故障检测中的应用[J].辽宁工学院学报,2002,22(5):13-15.

[3] 于　勇,孟广军.带电设备红外线检测及诊断[J].青海电力,2007,26(1):16-19.

[4] 胡世征,程玉兰,廖福旺,等.DL/T 664—1999 带电设备红外诊断技术应用导则[S].北京:中国电力出版社,2005.

[5] 付小宁,殷世民,吴志鹏,等.红外图像的动态阈值分割[J].光电工程,2002,29(6):57-60.

[6] 蒋锡健,吴功平,肖晓晖,等.高压输电线路巡检数据库及其管理系统[J].电力建设,2006,27(8):65-68.

(上接第 27 页)

通过对分布式倒排索引进行改进,引入二次的 VSM 排序,避免了传统算法中传输中间结果消耗的网络流量,并根据文档与特征词的关联度,对搜索结果进行了有效的排序。同时提出了一种通用的资源发布算法支持 TFIDF-VSM 中相应数据的维护,有较高的可行性。

参考文献:

[1] Stoica I, Morris R, Karger D, et al. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications[M]. USA: ACM Press, 2001:149-160.

[2] Rowstron A, Druschel P. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems [M]. Germany: Springer verlag, 2001.

[3] Reynolds P, Vahdat A. Efficient Peer-to-Peer Keyword Searching[M]. Rio de Janeiro, Brazil: [s.n.], 2003:21-40.

[4] 郑仲伟,郑有才.一个 P2P 搜索引擎的架构和实现[J].电子科技,2007(6):39-42.

[5] 王志晓,张大陆,刘　雷,等.基于本体的 P2P 复杂搜索[J].计算机应用,2007,27(4):780-783.

(上接第 68 页)

[3] CHEN Hong-na, ZU Xu, ZHOU Feng. On the Developing Situation, Research Content and Trend of Workflow Technology[J]. Journal of Chongqing Institute of Technology, 2006, 20(2):65-67.

[4] 郝丽波,李建华,夏明伟.工作流事务性研究综述[J].计算机工程与设计 2007,28(13):3209-3212.

[5] 郭科翔.工作流异常和常见的处理办法[J].闽江学院学报,2007,28(5):79-82.

[6] Shukla D, Schmidt B. Essential Windows Workflow Foundation[M]. [s.l.]: Addison-Wesley, 2006.

[7] Allen K S. Programming Windows Workflow Foundation: Practical W F Techniques and Examples using XAML and C#[M]. Birmingham: PACKT publishing, 2006.

[8] 吕成戍.工作流系统事务处理的研究与应用[D].大连:大连理工大学,2005.

[9] WANG Ni-hong, YU Hai-hao. Research on workflow technology and its developing trend[J]. information technology, 2007(6):67-69.

[10] 林春莺.工作流事务处理的应用解决方案[J].集美大学学报:自然科学版,2006,11(1):58-60.