基于构件软件的抗衰策略分析

王平涛 许满武 刘志军
(南京大学计算机科学与技术系软件新技术国家重点实验室 南京 210093)

摘 要 软件在长期运行过程中由于资源消耗、数据损坏、数值错误累积而引起性能下降的现象被称为软件老化。在软件失效前选择合适时机通过预反应的技术来保持软件性能的方法称作软件抗衰。由于构件软件常常应用在结构复杂、规模庞大的网络环境中,软件老化现象就更为突出。提出了一种针对构件软件的抗衰策略,通过对构件软件进行可靠性分析,构造了一种结合失效性和泊松失败分布的马尔可夫模型,并根据模型进行软件抗衰来保持构件软件的性能。

关键词 软件老化,软件抗衰,可用性,可靠性,马尔科夫模型,泊松分布
中图法分类号 TP393 **文献标识码** A

Analysis for Policy of Component-based Software Rejuvenation

WANG Ping-tao XU Man-wu LIU Zhi-jun
(Department of Computer Science and Technology, State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China)

Abstract Software aging is the process by which the state of long-running programs degrades over time through exhaustion of system resources, data corruption, or numerical error accumulation. Software rejuvenation is a proactive technique intended to reduce the probability of future unplanned outages due to software aging. The basic idea is to pause or halt the running software, refresh its internal state, and resume or restart it. The aging phenomenon of the component-based software applied to complex large-scale networks is more evident. The paper presented a policy on the component-based software rejuvenation, and designed the model incorporating deterioration and poisson failures by reliability analysis, and rejuvenated the software to maintain performance of system based on the model.

Keywords Software aging, Software rejuvenation, Availability, Reliability, Markov model, Poisson distribution

1 简介

软件开发的构件化和软件应用的网络化使得软件规模和复杂性都大大提高。受时间、开发成本等因素制约,软件一般都存在确定性故障和非确定性故障。确定性故障可以通过软件测试加以排除。因为内存、网络带宽等有限资源消耗而引起软件在运行过程中出现的内存泄露、内存溢出、网络或数据库连接错误等故障称为非确定性故障。非确定性故障与软件的具体运行环境有关,并具有不重演性,不能在软件开发过程中排除。非确定性故障造成软件性能下降的现象,称为软件老化[1]。由于构件软件常常应用在功能复杂、规模庞大的网络环境中,软件老化现象就更为突出。

目前解决软件老化现象主要是通过监测或者预反应的方法来提高软件系统的可用性。软件抗衰是 Huang Y 在 1995 年首先提出的一种预反应方法。它是指在软件失效前选择合适时机通过进行系统级或应用服务级的重启来保持软件的性能,随后许多研究人员相继提出了多种软件抗衰方法。文献[2]中提出了一种采用马尔科夫模型的基于时间的非参统计算法,文献[3]中描述了软件部分重启和全部重启两种抗衰方法。

到目前为止,软件抗衰研究集中于对整个系统的抽象建模上,所采用的策略多为系统级重启。现在的软件如 Web 服务器等一般担负着重要角色,需要有非常高的可用性,采用系统级重启要花费大量的开销,而随着基于构件技术的软件工程逐渐成为主流的软件范程,构件软件得到了广泛的应用,以具有独立功能的构件为重启单位将最大限度地提高软件的可用性,减少系统的开销。本文从提高构件软件整体的可用性出发,把构件软件运行过程划分为多个性能依次随时间下降的阶段,通过在合适的时机重启特定构件来保持整个软件的可用性。

本文提出了一种针对构件软件的抗衰策略,通过对构件软件进行可靠性分析,构造了一种结合可靠性和泊松失败分布的马尔可夫模型,并根据模型来确定进行软件抗衰的时机。

本文第 2 节是马尔科夫模型描述;第 3 节是软件可靠性计算;第 4 节是实现及验证;最后是结论和未来的工作。

到稿日期:2008-06-24 本文受国家自然科学基金(No. 60273035),国家高技术研究发展计划"863"(No. 2001AA113161),江苏省自然科学基金(No. BK2002080)资助。

王平涛(1972-),男,博士生,主要研究领域为软件方法学、新型程序设计,E-mail:wangpt@jsagri.gov.cn;许满武(1944-),男,教授,博士生导师,主要研究方向为软件方法学、新型程序设计;刘志军(1972-),男,博士生,主要研究领域为软件方法学、新型程序设计。

2 马尔科夫模型描述

2.1 条件和假设

(1) 假设构件软件由于软件老化而造成性能不断下降的过程是一个可划分为 $k+1$ 个阶段的离散的马尔可夫过程，每个阶段的持续时间呈指数分布，分布率为 δ_k，其中 $1,2,\cdots,k$ 表示构件软件处于各个不同稳定性能状态的阶段，f 表示构件软件处于失败阶段。

(2) 构件软件的故障类型包括泊松故障和性能下降故障两种，这两种故障类型都可通过维护来修复。假设构件软件有固定的泊松故障分布率 λ_0，就可以用 $1/\lambda_0$ 表示发生泊松故障的平均时间，用 $1/\mu_0$ 表示构件软件经过维护后返回到泊松故障前状态的平均时间。性能下降故障是由于构件软件中的部分构件性能下降或失效引起的，我们用 $h(t)$ 表示构件软件在时刻 t 时的故障率，用 $R(t)$ 表示整个构件软件的可靠性[4]。$h(t)=f(t)/R(t)$，其中 $f(t)$ 表示构件软件生命期的密度函数。当构件软件处于第 k 个稳定阶段时，随着构件系统性能的不断下降，构件系统的可靠性也逐渐下降，系统的故障率就逐渐增加，当系统的性能及可靠性下降到某个程度时，需要采取措施以保证构件软件能继续处于第 k 个稳定阶段。我们通过进行构件级的软件抗衰来使系统恢复到第 k 个稳定阶段的初始状态，即对性能下降或失效的构件及与其相关联的构件进行重启。用 μ_r 表示进行构件重启的平均时间。

(3) 用 $P(i,n)$ 表示构件软件在状态 (i,n) 时的稳定概率，i 表示构件软件所处的阶段，$i=1,2\cdots,k,f;n=0,1,2$ 分别表示软件处在正常工作状态、进行抗衰的状态或从泊松故障恢复的状态。

(4) 软件在进行构件重启或泊松故障维护时不能转变为失败状态。除非进行构件重启、泊松故障维护，否则软件一直处于运行状态。

2.2 马尔科夫模型

根据以上假设，我们建立了一个能够反映构件软件运行状态的马尔科夫模型。图 1 显示了构件软件性能下降及进行抗衰的整个过程，其中 $P(k,0)$ 表示软件在性能下降过程中的各个阶段的稳定状态，$P(k,1)$ 表示构件软件在遇到泊松故障后的状态，$P(k,2)$ 表示构件软件进行构件重启的状态。为清楚起见，图中没有列出构件软件性能下降的各个阶段持续时间 δ_k。

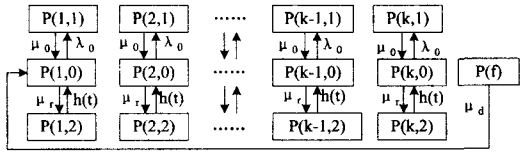

图 1 马尔科夫模型

2.3 模型分析

根据图 1 模型和排队论理论[4]，可得出以下等式：

构件软件初始运行阶段：

$$[\lambda_0+h(t)+\delta_1]P(1,0)=\mu_0P(1,1)+\mu_rP(1,2)+\mu_dP(f) \tag{1}$$

软件性能下降阶段：

$$[\lambda_0+h(t)+\delta_i]P(i,0)=\mu_0P(i,1)+\mu_rP(i,2)+\delta_{i-1}P(i-1,2),\text{其中}1<i\leqslant k \tag{2}$$

当软件处于失败阶段 f 时：

$$\mu_dP(f)=\delta_kP(k,2) \tag{3}$$

同时，也可得出以下公式：

$$\lambda_0P(i,0)=\mu_0P(i,1) \tag{4}$$

$$h(t)P(i,0)=\mu_rP(i,2),\text{其中}\ 1\leqslant i\leqslant k \tag{5}$$

根据以上等式可得到：

$$P(i,0)=\frac{\delta_i}{\delta_1}P(f) \tag{6}$$

$$P(f)=\sum_{i=1}^{k}\sum_{n=0}^{2}P(i,n)=1 \tag{7}$$

$$P(f)=\frac{1}{1+\frac{\mu_d}{\delta_1^2}(1+\frac{\lambda_0}{\mu_0}+\frac{h(t)}{\mu_r})\sum_{i=1}^{k}\delta_i} \tag{8}$$

系统的可用性为所有稳定状态 $(D(i,0),1\leqslant i\leqslant k)$ 概率的和，即：

$$A=\frac{\frac{\mu_d}{\delta_1^2}\sum_{i=1}^{k}\delta_i}{1+\frac{\mu_d}{\delta_1^2}(1+\frac{\lambda_0}{\mu_0}+\frac{h(t)}{\mu_r})\sum_{i=1}^{k}\delta_i} \tag{9}$$

系统的不可用性为：

$$U=\frac{\frac{\mu_d}{\delta_1^2}(\frac{\lambda_0}{\mu_0}+\frac{h(t)}{\mu_r})\sum_{i=1}^{k}\delta_i}{1+\frac{\mu_d}{\delta_1^2}(1+\frac{\lambda_0}{\mu_0}+\frac{h(t)}{\mu_r})\sum_{i=1}^{k}\delta_i} \tag{10}$$

要使系统的不可用性最小，则可使 $\frac{dU}{dt}=0$，设 $S=\frac{\mu_d}{\delta_1^2}(\frac{\lambda_0}{\mu_0}+\frac{h(t)}{\mu_r})\sum_{i=1}^{k}\delta_i$，$V=\frac{\mu_d}{\delta_1^2}\sum_{i=1}^{k}\delta_i$

$$\frac{dU}{dt}=\Big(\frac{S}{1+S+V}\Big)'=0 \tag{11}$$

式(11)化简后可得：

$S'+S'V-SV'=0$

也就是：

$$h(t)=\mu_r(1+\frac{\mu_d}{\delta_1^2}\sum_{i=1}^{k}\delta_i-\frac{\lambda_0}{\mu_0}) \tag{12}$$

式(12)表示构件软件处在各个稳定阶段时，当故障率 $h(t)$ 达到 $\mu_r(1+\frac{\mu_d}{\delta_1^2}\sum_{i=1}^{k}\delta_i-\frac{\lambda_0}{\mu_0})$ 时进行软件抗衰操作可使软件的可用性最大。根据构件系统故障率与可靠性的关系，我们通过计算构件软件的可靠性来确定软件抗衰的合适时机。

3 软件可靠性计算

目前，基于构件的软件系统的可靠性评估方法有 3 类[8]：基于状态的模型评估方法、基于路径的模型评估方法和基于操作剖面的模型评估方法。由于构件系统多是由第 3 方开发的现成构件通过胶合逻辑聚集而成的，构件开发者一般只是对构件本身进行可靠性度量，对构件在预期应用的操作剖面的可靠性给出描述。构件软件的可靠性不仅与每个构件的可靠性有关，也和构件运行的上下文环境、构件交互过程中接口的可靠性、信息传输的可靠性有关。我们采用的是一种基于路径的评估方法，用软件所有运行路径出现频度与该路径可靠性乘积的累加 $(R=\sum_{P_i\in\wp}(R_{P_i}\times F_{P_i}))$ 来评估软件的可靠性[5]，其中 R 为可靠性，$\wp$ 为所有运行路径的集合，R_{P_i} 和 F_{P_i} 分别为运行路径的可靠性和出现频率。对于基于构件的软件系统的可靠性计算模型可看作是表示构件依赖关系的有向图，图的节点表示构件，有向边表示构件之间的依赖关系。整

个软件可靠性计算过程可分为建立构件依赖图模型、参数分析、可靠性分析程序3部分。

3.1 构件依赖图

我们用称为构件依赖图(CDG)的有向图来表示构件之间的依赖关系和可能的执行路径[7]。构件依赖图定义：

$CDG=\langle N,E,s,t\rangle$，其中 n 代表构件，N 是构件集合，$N=\{n\}$；

e 表示构件之间的依赖关系，E 构件依赖关系的集合，$E=\{e\}$；

s 和 t 分别表示开始和结束构件集合；

$n=\langle C_i,R_{C_i},M_{C_i}\rangle$，其中 C_i 表示第 i 个构件的名称，R_{C_i} 表示第 i 个构件的可靠性，M_{C_i} 表示第 i 个构件的最大执行时间；

$e=\langle T_{ij},R_{T_{ij}},P_{T_{ij}}\rangle$，其中 T_{ij} 表示第 i 个构件和第 j 个构件之间依赖关系的名称，$R_{T_{ij}}$ 表示依赖关系 T_{ij} 的可靠性，$P_{T_{ij}}$ 表示依赖关系 T_{ij} 发生的概率；

$T_{ij}=\langle n_i,n_j\rangle$，表示从第 i 个构件转移到第 j 个构件的依赖关系。

在构件依赖图 $CDG=\langle N,E,s,t\rangle$ 中，若路径 $C_0T_{01}C_1T_{12}C_2T_{23}\cdots C_{n-1}T_{(n-1)n}C_n$ 满足：

1) $\forall 0\leqslant i\leqslant n, C_i\in N$；

2) $\forall 0\leqslant i\leqslant n, 0\leqslant j\leqslant n, T_{ij}\in E$；

3) $C_0\in s, C_n\in t$；

则称该路径为构件依赖图的一个运行路径。若运行路径上任一节点或转移失效，则此路径失效。

根据软件的体系结构和不同的输入所调用的不同构件及构件的依赖关系，就可构建软件的构件依赖图。

3.2 参数分析

构件依赖图建成后，就需要获得有关参数的具体数据。R_{C_i} 和 M_{C_i} 可从构件的详细配置说明中获得。构件转移依赖关系的可靠性 $R_{T_{ij}}$ 不仅与构件接口的兼容性有关，也和构件之间通信链路的稳定性有关。一个构件的接口描述与其他构件之间的交互关系，输入接口表示构件需要得到其它构件提供的系列服务，输出接口表示此构件所提供的系列服务。数据结构不兼容、数据传递的序列有误、数据格式或类型不匹配、请求的时间严格性不一致都可能导致进行数据交互的构件之间的接口不完全匹配，从而影响构件转移依赖关系的可靠性。由于构件的应用环境多与网络有关，构件之间的调用可能是跨平台和异构网络的，因此操作系统、网络的连通性等因素也影响构件转移依赖关系的可靠性。构件转移依赖关系的可靠性超出了本文的研究范围，我们采用文献[6]的构件转移依赖关系的可靠性计算方法来分析其可靠性。$P_{T_{ij}}$ 可以采用下式计算：

$$P_{T_{ij}}=\sum_{P_i\subset\wp}F_{P_i}\times\frac{\sum\langle C_i,C_j\rangle}{\sum_{h=1}^{n}\langle C_i,C_h\rangle}$$

其中 $\sum\langle C_i,C_j\rangle$ 表示在路径 P_i 中 T_{ij} 出现的次数，$\sum_{h=1}^{n}\langle C_i,C_h\rangle$ 表示在路径 P_i 中构件 i 作为源构件的所有构件转移依赖关系出现次数的和。我们通过路径中每个构件的最大执行时间来计算路径 P_i 的最大执行时间 M_{P_i}。

3.3 可靠性分析程序

利用构件依赖图和上述的参数分析，我们可以利用下面的算法来获得软件系统的可靠性：

算法：

输入：M_{C_i}，CDG

输出：R

初始化：$R=0$，$Time=0$，$R_{temp}=1$

While ($P_i\in\wp$) do

  push$\langle C_i,R_{C_i},M_{C_i}\rangle$，$Time=0$，$R_{temp}=1$

While Stack not EMPTY do

pop$\langle C_i,R_{C_i},M_{C_i}\rangle$，Time，$R_{temp}$

If $Time>M_{P_i}$ or $C_i=t$

$R_{P_i}+=R_{temp}$

Else

push$\langle C_j,R_{C_j}$，$Time+=M_{C_i}$，$R_{temp}=R_{temp}\times R_{T_{ij}}\times P_{T_{ij}}$

End if

End while

  $R+=R_{P_i}\times F_{P_i}$

End while

通过计算每一路径的可靠性和发生的概率来确定整个系统的可靠性，为了防止出现无穷长路径，当路径中的下一个节点是结束节点或路径的执行时间大于路径的最大执行时间时，路径深度扩展停止，这样保证系统不会进入死循环。

得到软件的可靠性后，与通过马尔科夫模型获得的可靠性进行对比，如果软件的可靠性大于模型获得的值，则说明软件还不需要执行抗衰操作，反之，则说明软件已经变得不可靠，需要进行抗衰操作。

4 实现及验证

在分布式多媒体应用中可靠性受到越来越多的关注，其中对服务质量有着严格要求的视频点播(VoD)服务就需要有更高的可靠性。

我们开发的基于MPEG-7的视频点播系统(VoD-7)在设计时采用了本文提出的软件抗衰策略。VoD-7由视频管理、音频管理、视频播放、音频播放、路由、交换、网络传输、管理器等组件构建而成[9]。管理器的主要功能就是记录组件的相互调用关系、监控系统资源消耗情况、根据马尔科夫模型和可靠性计算来确定重启的具体构件和时间。我们采用主频2.8 GHz双处理器的微机服务器作为视频点播服务器，用户要点播的多媒体信息存放在磁盘阵列中，视频点播服务器通过光纤接入南京大学校园网供校内用户访问。根据不同的用户请求类型和系统已有的构件相互关系来计算整个系统在某一运行时刻的可靠性，再与通过马尔科夫模型计算出的系统可靠性进行比较来确定是否需要进行软件抗衰操作，若需要则根据管理器记录的构件运行时产生的动态依赖关系来确定需要重启的相关构件。在系统运行过程中，我们观察到系统的内存、cpu等有限资源能够被反复利用和释放，系统性能能够在相当长时期内保持稳定。当构件重启时会造成服务的暂时中断，为了避免用户请求的丢失，我们的网络传输组件在构件重启时缓存用户的请求，在构件重启完成后重试用户请求，对于正在接受服务的用户，我们采取返回提示的方式请用户等待，等重启完成后继续为用户服务。相比较整个机器重启要花费大量的时间而言，构件重启所需要的时间能够被用户接受。

结束语 目前软件抗衰研究多集中于传统软件的建模和

(下转第180页)

C-value	16.13%	23.95%	37.47%
MI	6.09%	12.56%	31.62%

5.3 结果分析

表1将3种方法实验中前2000个结果的正确率进行了比较，很明显可以看出，DV的正确率比其他两个方法要高出很多。在抽出的前1000个术语中，DV方法的正确率比C-value方法要高出24个百分点，比MI方法高出34个百分点。对3个实验抽取出的前1000个术语中错误的结果进行分析后发现，结构不完整的词串占了比较大的部分，如"中随机抽取"、"数据集上"，这类错误的词串一般以介词"中"、"上"、"基于"或动词"使用"、"进行"等开头或结尾。出现这种错误主要是由于本文使用的语言规则比较宽松，并没有像文献[9]中限制于名词短语。虽然这种错误在3个实验结果中都出现了，但具体所占的比例却有很大的差距，如表3所列。

表3 DV、C-value和MI方法实验抽取出的前1000个术语中错误词串结构完整和不完整的数目

	DV	C-value	MI
结构不完整	69	275	100
结构完整	39	74	346

由表3可以发现，DV和C-value结果的错误词串中，结构不完整因素占了绝大多数，DV为63.89%，C-value方法为79.80%。而MI方法则完全相反，结构不完整的词串只占了22.42%。虽然DV方法的错误词串中结构不完整因素占了比较大的比例，但是总的数量却远比另外两种方法少，由此可见，DV方法具有一定的能力能够排除结构不完整的词串。

对结构完整的错误词串进行分析后发现，3种方法的错误原因各不相同。在DV的实验结果中，那些词频较高，但文档频率非常低的普通词串会被误识为术语，如"网络聊天"和"新闻报道"，前者只在1篇文档中出现过，但词频高达73；后者在4篇文档中共出现了32次，这主要是由于作者采用了网络聊天或新闻报道的内容作为语料，这和本文使用的测试语料规模较小有关。而C-value方法中，普通词串的识别能力很差，错误中甚至包括最常用的"表1"、"图2"等，排名分别为48和93，但在其他两种方法中，这两个词的排名均在3000之后。而且C-value方法对低频术语的识别能力较弱，如："贝叶斯算法"，在语料中只出现了9次，在C-value方法的结果中排名2100，但是在DV的实验结果中排名675。MI方法则对那些低频、结构稳定的普通词串鉴别能力较弱，如变化较少的人名、地名和机构名等。

通过表2可以发现3种方法的召回率普遍都比较低，相差并不大。导致召回率偏低的主要因素是测试语料的规模很小，很多术语在整个测试语料中只出现了1，2次，在抽取候选术语模块中，就已经被过滤掉。除此以外，还有词性标注错误、语法规则覆盖面不够广等原因。

结束语 本文提出了一种利用术语在语料中的词频分布变化来衡量其termhood的计算方法，并结合一系列的语言知识，构成一个完整的中文术语抽取系统。从实验可以看出，在小规模的语料中，该方法要优于目前常用的C-value方法和互信息方法，尤其是具有较强的低频术语和高频普通词串识别能力。

在下一步的工作中，我们将针对实验中出现的错误，加入词串结构稳定性的检验，进一步优化并完善现有的术语抽取系统。

参 考 文 献

[1] Bourigault D. Surface Grammatical Analysis for the Extraction of Terminological Noun Phrases[C]//Proceedings of COLING'92. 1992:977-981

[2] Pantel P, Lin D. A Statistical Corpora-based Term Extractor[C]//Lecture Notes in Artificial Intelligence. Springer, Verlag, 2001:34-46

[3] Frantzi K T, Ananiadou S, Mima H. Automatic Recognition of Multi-word terms: the C-value/NC-value Method[J]. International Journal on Digital Libraries, 2000, 3(2):115-130

[4] Kageura K, Umino B. Methods of Automatic Term Recognition: A Review[J]. Terminology, 1996, 3(2):259-289

[5] 刘桐菊，于浩，杨沐昀. 基于TFIDF的专业领域词汇获取的研究[C]//第一届学生计算语言学研讨会论文集. 2002

[6] 李勇. 基于聚类方法对特定领域术语的自动筛选[J]. 计算机工程与科学，2008，30(2)：64-66

[7] 张普. 信息领域汉语术语的特征及其在语料中的分布规律[J]. 语言教学与研究，2001

[8] 张榕. 术语定义抽取、聚类与术语识别研究[D]. 北京：北京语言大学，2006

(上接第123页)

分析，而随着构件技术的快速发展，利用构件设计大型复杂软件系统的软件开发方法日趋成熟，如何评估构件软件的可靠性，研究构件软件的老化原因，设计针对构件软件的抗衰策略，保持构件软件的性能，都需要新的适合构件软件的研究方法。

本文利用马尔科夫模型，通过分析构件软件的可靠性，提出了一种能够评估构件软件的可靠性并通过软件抗衰来保持软件性能的软件分析方法，今后我们将结合实例来进一步研究构件软件抗衰问题。

参 考 文 献

[1] Avritzer A, Weyuker E. Monitoring smoothly degrading systems for increased dependability[J]. Empirical software Engng, 1997(2):55-77

[2] Dohi T, Goseva-Popstojanova K, Trivedi K S. Estimating software rejuvenation schedules in high-assurance systems[J]. Compute, 2001, 44:473-82

[3] Wei X, Yiguang H, Trivedi K S. Analysis of a two-level software rejuvenation policy[J]. Reliability engineering & system safety, 2005, 87(1):13-22

[4] Cross D, Harris C N. Fundamentals of Queuing Theory. John Wiley and Sons, New York

[5] Goševa-Popstojanova K, Trivedi K S. Architecture-based approach to reliability assessment of software systems[J]. Performance Evaluation, 2001, 45(7):179-204

[6] Barlow R E, Proschan F. Statistical theory of reliability and life testing: probability models. New York: Holt, Rinehart and Winston, 1975

[7] Yacoub S, Cukic B, Ammar H. A Scenario-based Reliability Analysis Approach for Component-based Software [J]. IEEE Transactions on Reliability, 2004, 53(4):465-480

[8] 毛晓光，邓勇进. 基于构件软件的可靠性通用模型[J]. 软件学报，2004，15(1)：27-32

[9] 单锦来，陈博，杨献春，等. MPEG-7和MPEG-7实验模型参考软件[J]. 计算机科学，2003，30(6)：31-37