基于耦合映像格子的软件相继故障研究

周宽久 兰文辉 冯金金
(大连理工大学软件学院 大连 116621)

摘 要 软件系统存在故障,无法避免,如何使软件系统在自身发生错误或者受到外界干扰时仍然能够提供连续的无差错的服务,是一个亟待解决的理论问题。大型软件的静态和动态函数调用网络具有小世界效应和无标度特性,基于耦合映像格子的相继故障模型,分析软件系统相继故障的形成机理与传播行为,以提高基于关键节点的软件测试的可信性。

关键词 复杂网络,小世界效应,无标度特性,可信软件,相继故障,耦合映像格子
中图法分类号 TP301 **文献标识码** A

Software Cascading Failures Based on Coupled Map Lattice

ZHOU Kuan-jiu LAN Wen-hui FENG Jin-jin
(Software School, Dalian University of Technology, Dalian 116621, China)

Abstract It is unavoidable that there exist faults in software systems, and how to provide continuing error-free service when software system breaks down or is disturbed by the external perturbation, is an urgent problem of the theory. Networks of static function-call and weighted networks from dynamic software execution are of small-world and free scale. Based on the model of coupled map lattice, the formation mechanism and propagation behavior of cascading failures in the software system were studied to improve the credibility of software testing based on critical node.

Keywords Complex network, Small-world effect, Scale-free property, Trusted software, Cascading failures, Coupled map lattice

1 引言

软件在系统中的作用愈发重要,其规模和复杂性不断增加。相应地,软件可信性问题也日益突出,以高可信软件工程技术为重要内容的可信软件研究成为软件技术研究的热点之一[1]。所谓"可信软件",通常是指在特定环境下运行行为及其结果符合人们预期,并在受到干扰时仍能够提供连续服务的软件[2]。

复杂网络是当前研究的热点。自然界中存在的大量复杂系统如社会关系网[3]、生态网络、电力网络、Internet、交通网等等,它们都可以通过形形色色的网络加以描述,有明显的复杂网络特性[4-7],即小世界效应和无标度特性等。随着复杂网络研究的深入,人们开始用复杂网络理论研究软件系统的特性。最近研究表明,大型软件系统中也存在复杂网络现象[8-11],它们的内部结构并不是随机的,也具有小世界效应和无标度特性。

很多实际网络中的一个或者少数几个节点发生故障(这种故障可能是随机发生的,也可能是由蓄意攻击造成的)会通过节点之间的耦合关系引起其他节点发生故障,产生连锁效应,最终导致相当一部分节点甚至整个网络的崩溃。这种现象称为相继故障,也称雪崩[4,12]。耦合映像格子(coupled map lattice,CML)是一个时间、空间都离散而状态保持连续的非线性动力学模型[13,14],近年来已经被广泛应用于研究复杂系统的时空动力学行为。

与以往研究静态的软件系统不同,本文把软件执行路径看作是加权网络,通过分析加权网络的点强度、聚类系数和节点平均最短距离,得出结论:软件执行路径的加权网络具有复杂网络特性,即小世界效应和无标度特性。基于CML模型,有效地刻画了软件系统在受到干扰时软件故障的规模和传播速度。最后分别模拟了软件执行路径的加权网络在受到随机攻击和蓄意攻击时的不同结果,并提出了基于关键节点的软件测试技术,以用较少的测试用例保证软件的可信性。

2 软件系统的复杂网络特性

文献[15-17]都是研究静态的软件系统。和以往的研究不同,本文研究软件的动态执行过程,把函数看作是节点,函数之间的调用看作是节点之间的有向边,函数的调用次数看作是有向边的权重。

在linux操作系统上,使用Gcc重新编译GNU的开源源码包,通过加入一些调试信息,从而能够自动收集软件执行路径的信息,得到调用关系的拓扑图。本文分别对tar程序、gedit程序和emacs程序进行实验,得出软件执行路径的加权网

到稿日期:2010-06-08 返修日期:2010-09-01
周宽久(1966—),男,博士生,教授,CCF会员,主要研究方向为系统工程和软件工程,E-mail:zhoukj@dlut.edu.cn;兰文辉(1983—),男,研究生,主要研究方向为系统工程和软件工程;冯金金(1986—),女,研究生,主要研究方向为系统工程和软件工程。

络具有明显的无标度特性。

对于每一个程序，都用 200 个测试例子来测试，每一个测试例子生成一个执行路径。文献[15-17]从软件的静态结构来分析，把软件看作是无权网络，指出了软件具有小世界效应和无标度特性。在这里，软件执行路径看作是加权网络。图 1 显示了 tar 程序执行路径的加权拓扑图。

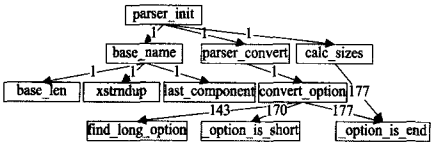

图 1　tar 程序执行路径的加权拓扑图

2.1　小世界效应

把软件执行路径看作是无权网络时，它具有明显的小世界效应。表 1 显示了再把软件执行路径看作是加权网络时的实验数据。作为对比，L_{rand} 表示随机网络的平均路径长度，C_{rand} 表示随机网络的聚类系数。

表 1　软件执行路径的加权网络统计信息

程序	Node	Edge	S	L	L_{rand}	C	C_{rand}
tar	327	699	90.7	0.429	12.5	0.003	0.00015
gedit	623	1435	25.3	0.592	10.6	0.004	0.00036
emacs	783	1627	28.2	0.613	12.6	0.003	0.00032

从表中可以看出，软件执行路径所构成的加权网络比随机网络具有更小的平均最短距离和更大的聚类系数，从而说明了软件执行路径所构成的加权网络同样具有小世界效应。

2.2　无标度特性

在软件执行路径的无权网络中，度分布具有无标度特性；在加权复杂网络中，度自然推广为点强度。图 2 显示了入点强度的分布。

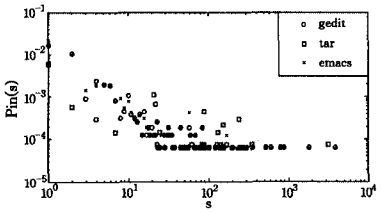

图 2　软件执行路径构成的加权网络的入点强度的分布对数图

从图中可以看出，软件执行路径的加权网络的入点强度具有明显的无标度特性。更精确地说，对于 gedit 程序，入点强度分布为：

$$Pin(s)=0.0168*s^{-1.376} \tag{1}$$

对于 tar 程序，入点强度分布为：

$$Pin(s)=0.006*s^{-2.372} \tag{2}$$

对于 emacs 程序，入点强度分布为：

$$Pin(s)=0.0217*s^{-1.575} \tag{3}$$

通过大量实验可知，软件执行的加权网络具有复杂网络特性，即小世界效应和无标度特性。这就为深入研究软件系统的各种特性及动力学过程与拓扑结构间的相互作用奠定了理论基础。近年来，研究复杂网络上相继故障的动态模型主要有负荷-容量模型、二值影响模型、OPA 模型、CASCADE 模型和耦合映像格子模型。考虑到软件系统的复杂网络特性，CML 模型被广泛应用到复杂网络研究中。因此为了更好地分析不同节点对于软件系统的影响，本文同样采用耦合映像格子模型来研究软件系统的相继故障。

3　CML 相继故障模型

大多数时空系统的时间、空间和状态数量是连续的，适宜用偏微分方程来描述。无论是进行理论分析还是数值分析，都比较复杂，而且运算量大。一种有效的方法是将时间和空间离散化，但状态变量仍保持连续，既能克服上述缺点，又能从本质上显示出系统的复杂时空特性，这就是耦合映像格子。CML 最初由 Kaneko 于 1984 年提出[13]，它是一个时间、空间都离散而状态保持连续的非线性动力学模型，近年来已经被广泛应用于研究复杂系统的时空动力学行为，在神经网络[18]、城市交通[19]等很多方面都取得了广泛的推广和应用。

根据第 2 节对软件复杂网络特性的描述，为考察函数的变化对整个软件系统整体的影响，将函数映射成网络节点。考虑包含 N 个节点的 CML 模型：

$$x_i(t+1)=|(1-\varepsilon)f(x_i(t))+\varepsilon\sum_{j=1,j\neq i}^{N}w_{i,j}f(x_j(t)/s(i))| \quad i=1,2,\cdots,N \tag{4}$$

式中，$x_i(t)$ 表示第 i 个节点在 t 时刻的状态，N 个节点的连接信息用连接矩阵 $A=(w_{ij})N\times N$ 表示。w_{ij} 表示节点 j 到节点 i 的入边权重，ε 表示耦合强度，$s(i)$ 表示节点 i 的入点强度。函数 f 表示节点自身的动态行为，这里选择混沌 Logistic 映射：$f(x)=4x(1-x)$。绝对值符号保证各节点的状态非负。

如果节点 i 的状态在 m 个时序内始终在(0,1)范围内，即 $0\leqslant x_i(t)\leqslant 1, t\leqslant m$，那么称节点 i 处于正常状态。如果在 m 时刻，节点 i 的状态 $x_i(m)\geqslant 1$，那么称节点 i 在此时刻发生故障。这种情况下，节点在以后的任意时刻状态都恒等于零。如果所有 N 个节点的初始状态都在(0,1)范围内，且没有外部骚动(即没有函数节点出现错误)，那么所有节点都将永远保持正常状态。

为了研究单个节点受到冲击导致的相继故障，在 m 时刻给节点 b 施加一个外部扰动 $R\geqslant 1$：

$$x_b(t+1)=|(1-\varepsilon)f(x_b(t))+\varepsilon\sum_{j=1,j\neq b}^{N}w_{bj}f(x_j(t))/s(b)|+R \tag{5}$$

这种情况下，节点 b 在 m 时刻发生故障。在 $m+1$ 时刻，所有与节点 b 相连的节点都将受到 m 时刻 b 节点的状态 $x_b(m)$ 的影响，并且这些节点的状态按照式(5)计算得出。此时计算的节点状态值也可能大于 1，从而引发新一轮的节点故障。这个过程反复进行，节点故障就可能扩散。

图 3 显示了软件中相继故障的传播过程。当节点 i 发生故障后，它就通过节点之间的调用关系以及调用的次数，根据式(4)把故障传播到它相邻的节点上。

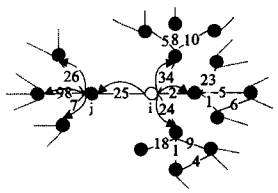

图 3　软件相继故障过程

4　数值模拟

4.1　算法分析

基于 CML 的软件相继故障的模拟算法：

初始化所有节点的状态值、ε的值，系统有 N 个节点。

$t=0$;//从0时刻开始模拟软件系统各个节点的状态。

$allNum=0$;//记录发生故障的节点数目。

while(网络中不再有新节点发生故障)

for(在 t 时刻，对网络中的每一个没有发生节点 i)

$F_i(t)=(1-\varepsilon)f(x_i(t-1))$//根据ε的取值，计算上一时刻自身节点对 t 时刻 i 节点的影响值 $F_i(t)$。

for(对节点 i 相连的每一个节点 j，总共有 A 个节点)

$F_j(t)=(W_{ij}/S(i))*f(x_j(t-1))$//计算上一时刻节点 j 的状态对 t 时刻节点 i 的影响 $F_j(t)$。

$x_i(t)=F_i(t)+\varepsilon\sum_{j=0}^{A}F_j(t)$//计算 t 时刻节点 i 的状态值。

if($t==m$)

$x_b(m)=x_b(m-1)+R$;//随机选择在 m 时刻，对某节点 b 施加一个外部干扰 $R\geqslant 1$;

for(对于上一时刻被记录发生故障的节点 i)

$allNum+=1$;//发生故障节点总数加1

$x_i(t)\equiv 0$;//在 t 时刻以后，其状态值恒等于0

$L[i]=0$;//清零

for(对网络中的每一个节点 i)

if(节点状态值>1)//发生故障的节点

记录节点的位置，保存在 $L[i]$;//如果 $L[i]$ 值为1，说明 i 节点是故障节点

统计 t 时刻网络中发生节点故障的总数;//便于统计故障规模的速度

$t+=1$;//时钟加1

统计故障规模、故障发生速度等;

4.2 随机故障和蓄意攻击的模拟结果分析

考虑两种引发相继故障的策略：随机故障和蓄意攻击。也就是说，随机选择一个节点，或者选择点强度最大的节点，对它的状态施加扰动 $R\geqslant 1$，判断网络中最终有多少节点发生故障。

根据第2节对gedit，emacs和tar程序所做的实验，得到每一个程序的节点总数和节点的连接矩阵。表2显示了实验程序的节点总数和入点强度分布。

表2 实验程序的信息统计

程序	节点总数	入点强度分布
tar	327	$Pin(s)=0.006*s^{-2.372}$
gedit	623	$Pin(s)=0.0168*s^{-1.376}$
emacs	783	$Pin(s)=0.0217*s^{-1.575}$

由于本文实验的对象是面向过程的软件系统，而面向过程的软件系统中，函数之间的耦合强度都比较大，因此取ε为0.6，根据软件相继故障的模拟算法，得到故障规模相对值 $S=I/N$。其中 I 表示故障规模，N 表示节点总数。

图4显示了蓄意攻击和随机故障情况下，故障规模的大小。图中数据分别是200次试验的平均值。从图4(a)可以看到，曲线上都有一个临界点 R_c，当 $R\leqslant R_c$ 时，网络中最终只有4%左右的节点发生故障。当 R 由 R_c 增加时，故障规模也迅速上升。因此当 $R\leqslant R_c$ 时，就应该对软件系统进行及时控制，如果不进行控制，软件系统故障会迅速增加，从而导致整个软件系统崩溃。从图4(b)可以看出，在随机故障中，故障规模没有随着 R 的增大而增大，而是始终保持在一个很低的水平上。在这里始终控制在4%以内。对比图4(a)和图4(b)，在给定 R 的情况下，蓄意攻击要比随机故障所造成的破坏严重得多。

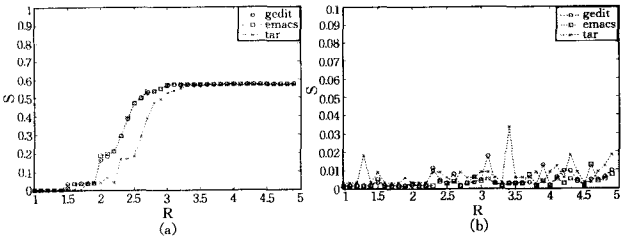

图4 故障规模的大小

此外，在给定 R 的情况下蓄意攻击所造成的破坏扩散速度也比随机的快。图5是对gedit软件执行路径生成的加权网络做的模拟，ε为0.6，其中 $R=4$(如图4(a)所示，这样可以保证最终的故障规模最大)，4条曲线分别对应点强度不同的节点。从中可以看出，节点的点强度越大，其破坏的速度也越快。

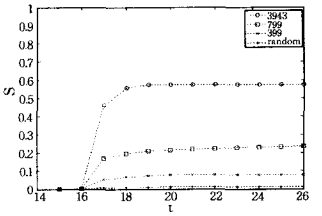

图5 软件相继故障的扩散速度

4.3 基于关键节点的软件测试

由4.2节的模拟结果可知，对于一个软件系统，施加一个外部扰动 R，当作用到点强度很大的节点上时，所造成的故障规模和故障规模的传播速度都明显大于作用到点强度小的节点上。因此，为了使软件系统可信，即在受到外界干扰时仍然能够提供连续服务，我们定义软件系统中点强度大的节点为关键节点。对系统中的关键节点进行测试，保证能够在任何环境下都能正常工作，为其它节点提供一个可信的基础。

本文以gedit软件系统为例，分析基于关键节点的软件测试，并模拟其效果。gedit软件执行路径生成的加权网络共有节点623个，入点强度总和为15773。通过计算可以得到gedit中各个节点(函数)的点强度。表3显示了点强度最大的10个节点的点强度大小、每个节点的点强度占总点强度的比例，并且把这10个节点定义为关键节点。

表3 入点强度最大的10个节点

名字	点强度	百分比
g_once_init_enter	3943	25.0%
gedit_document_get_type	1832	11.6%
gedit_debug	893	5.6%
gedit_window_get_type	799	5.1%
gedit_tab_get_type	611	3.9%
gedit_debug_message	525	3.3%
gedit_view_get_type	359	2.3%
gedit_tab_get_view	302	1.9%
language_compare	275	1.7%
gedit_window_get_active_view	268	1.7%
总计	9807	62.2%

通过关键节点的定义，可以采取各种测试方法来保证关键节点的可靠性，使其能够在各种环境下保证运行的正确性。这样就为关键节点建立了一层保护，使其对其它节点可信，并使这个软件系统可信。图6显示了保护前和保护后蓄意攻击所造成的故障规模。

(下转第174页)

cal Word Reordering in English to Turkish Phrase-based Statistical Machine Translation[C]//IEEE Early Access. 2009:1

[21] Hettige B, Karunananda A S. A Morphological Analyzer to Enable English to Sinhala Machine Translation[C]//IEEE Conferences. 2006:21-26

[22] Duwairi R, Al-Refai M, Khasawneh N. Stemming Versus Light Stemming as Feature Selection Techniques for Arabic Text Categorization[C]//IEEE Conferences. 2007:446-450

[23] Karageorgakis P, Potamianos A, Klasinas I. Towards incorporating language morphology into statistical machine translation systems[C]//IEEE Conferences. 2005:80-85

[24] Robertson S E, et al. Okapi at TREC-8, In The Enghth Text Retrieval Conference (TREC 8) [J]. Gaithersburg, MD: NIST, 2000

[25] 刘海峰,王元元,丘国防. 密度聚类模式下一种基于层次的自动文摘方法研究[J]. 情报杂志,2007(03)

[26] 刘金红,陆余良. 基于 Ontology 改进的 N-Gram 文本分类模型研究[J]. 计算机工程与设计,2007(13)

[27] 于津凯,王映雪,陈怀楚. 一种基于 N-Gram 改进文本特征提取算法[J]. 图书情报工作,2004,48(8):48-51

[28] 朱志国,邓贵仕,孔立平. 基于 N-gram 的 Web 用户浏览模式分类算法研究[J]. 情报学报,2009(6):389-394

[29] 刘鹏远,赵铁军. 基于 Web 的无指导译文消歧词模型与 N-gram 模型及对比研究[J]. 电子与信息学报,2009,31(12)

[30] 秦健. N-gram 技术在中文词法分析中的应用研究[D]. 青岛:中国海洋大学,2009:1-63

(上接第 131 页)

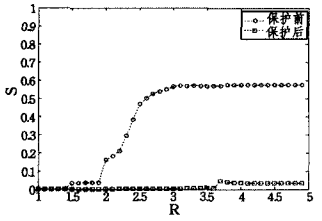

图 6 保护前后软件相继故障规模

由图 6 可知,在对关键节点提供保护之后,蓄意攻击所能造成的故障规模明显减小,控制在 5%左右。点强度的分布具有明显的无标度特性,也就是说点强度较大的节点只占节点总数的很少一部分。通过表 3 也可以看出,关键节点只定义了 10 个,而总的节点有 623 个,关键节点占总节点的 1.6%,这样就可以用较少的测试用例来保证软件的可信性。

结束语 本文把软件执行路径看作是加权网络来研究,利用复杂网络的理论来分析软件执行路径。在进行大量实验的基础上,发现软件执行路径的加权网络具有小世界效应和无标度特性,不仅仅点强度分布具有无标度特性,权重的分布同样具有无标度特性。

基于 CML 相继故障模型,研究软件系统中各个节点(函数)的非线性动力学特征、软件系统大规模故障的形成机理与传播行为。根据模拟结果可以看出,点强度越大,它发生故障所能造成的最终故障规模越大。因此本文提出了基于关键节点的软件测试技术,定义点强度大的节点为关键节点。基于 gedit 软件,模拟了基于关键节点的软件测试技术的效果。定义 1.6%的关键节点,就可以保证故障的最终规模在 5%以内,即使用较少的测试用例就能保证软件的可信性。

参 考 文 献

[1] 陈火旺,王戟,董威. 高可信软件工程技术[J]. 电子学报,2004,31(12):1934-1938

[2] 刘克,单志广,王戟,等. "可信软件基础研究"重大研究计划综述[J]. 中国科学基金-科学进展与展望,2008,3:145-151

[3] Watts J S, Strogatz S H. Collective dynamic of small-world networks [J]. Nature, 1998, 393(6684):440-442

[4] Strogatz S H. Exploring complex networks [J]. Nature, 2001, 410(6825):268-276

[5] Albert R, Barabási A-L. Statistical mechanics of complex networks [J]. Rev. Mod. Phys, 2002, 74(1):47-97

[6] Dorogovtsev S N, Mendes J F F. Evolution of Networks: From Biological Nets to the Internet and WWW [M]. London: Oxford University Press, 2003

[7] Newman M E J. The structure and function of complex networks [J]. SIAM Review, 2003, 45(2):167-256

[8] Potanin A, Noble J, Frean M, et al. Scale-free geometry in object-oriented programs [J]. Communications of the ACM, 2005, 48(5):99-103

[9] Valverde S, Ferrer-Cancho R, Sole R V. Scale-free Networks from Optimal Design [J]. Europhysics Letters, 2002, 60(4): 512-517

[10] Myers C R. Software systems as complex networks: structure, function, and evolvability of software collaboration graphs [J]. Phys. Rev. E, 2003, 68(4):046116

[11] Wheeldon R, Counsell S. Power Law Distributions in Class Relationships [C]//Proc Third IEEE Int'l Workshop Source Code Analysis and Manipulation. 2003

[12] Crucitti P, Latora V, Marchiori M. Model for cascading failures in complex networks[J]. Phys. Rev. E, 2004, 69:045104

[13] Kaneko K. Period-doubling of kink-antikink patterns, quasiperiodicity in antiferro-like structures and spatial intermittency in coupled map lattices [J]. Prog. Theor. Phys, 1984, 72(3): 480-486

[14] Kaneko K. Coupled map lattices[Z]. Singapore: World Scientific, 1992

[15] Concas G, Marchesi M, Pinna S, at al. Power-laws in a large object-oriented software system [J]. IEEE Transactions on Software Engineering, 2007, 33(10):687-707

[16] Hyland-Wood D, Carrington D, Kaplan S. Scale-free Nature of Java Software Package, Class and Method Collaboration Graphs [R]. MIND Laboratory, University of Maryland Collage Park, 2006

[17] Zheng X L, Zeng D, Li H Q, et al. Analyzing open-source software systems as complex network [J]. Physica A, 2008, 387(24):6190-6200

[18] 王宏霞,何晨. 细胞神经网络的动力学行为[J]. 物理学报,2003,52(10):2409-2414

[19] 陈星光,周晶,朱振涛. 基于耦合映像格子的城市交通系统相继故障研究[J]. 数学的实践与认识,2009,39(7):79-84