第 20 卷 第 4 期 福建师范大学学报（自然科学版） Vol120 No14
2004 年 12 月 Journal of Fujian Normal University (Natural Science Edition) Dec12004

文章编号: 100025277(2004)0420028204

软件故障注入方法及其仿真应用[X]

谷振宇[1]，王胜文[2]，洪炳熔[2]，乔永强[3]
(11 福建省高等学校招生委员会办公室计算机管理科，福建 福州 350003; 21 哈尔滨工业大学计算机科学与技术学院，黑龙江 哈尔滨 150001; 31 航天时代电子公司研究院，北京 100081)

摘要: 软件故障注入是一种评测系统可靠性的有效方法，但是对目标系统的开放性要求比较高，这给不能开放源码的待测系统的可靠性测试带来困难 给出一种引入智能控制策略的软件故障注入方法，在不开放系统源码的情况下，可以对系统的可靠性进行评测，通过仿真实验证明了该方法的有效性

关键词: 可靠性系统; 评测; 智能控制; 软件故障注入
中图分类号: TP301 **文献标识码**: A
XX

Simulation Application of Software Implemented Fault Injection

GU Zhen-yu[1], WANG Sheng-wen[2], HONG Bing-rong[2], QIAO Yong-qiang[3]
(11*General Office of Fujian Higher Educational Colleges and Universities Admission Committee, Fuzhou* 350003, *China*; 21*Department of Computer Science and Engineering, Harbin Institute of Technology, Harbin* 150001, *China*; 31*Areospace Institute of Epoch Electronic, Beijing* 100081, *China*)

Abstract: Software fault injection (SFI) is one of the most important methods in dependability evaluation of reliability system. there are diffcults to some disable get source2code system which needs reliability testing. A kind of SFI with intelligent control strategy is proposed, this methed be used to reliability evulation without source2codes, Its efficiency is shown by a simulated example.

Key words: reliability system; evaluation; intelligent control; software fault injection

软件故障注入作为一种比较成熟的评测系统可靠性的关键技术，以其对硬件无损伤、易实现、易移植等特点，引起众多工程设计者和研究人员的重视 软件故障注入是目前对可靠性系统中可靠性机制评测的有效方法[1]，但是 前提是被测系统必须对实验者开放[2]. 在某些保密领域，开放系统是不允许的，这使系统测试变得十分困难 本文提出一种基于智能控制的故障注入实验方法，它通过软件方法，在系统中引入智能控制策略，把故障注入系统内核嵌入到目标系统，进行故障注入实验时由故障注入系统内核代替目标系统内核运行，在不开放系统的情况下，对目标系统的输出状态进行跟踪控制，获得比较好的实验效果

1 软件故障注入

故障注入实验的目的是为了对容错系统所采用的容错机制进行测试和评估 作为评测容错机制的有效方法，故障注入技术通过向目标系统中引入故障，加速目标系统发生故障和失效的过程，然后通过对注入故障后的目标系统的监测，获得目标系统存在故障时的运行情况，与系统的声明进行比较，从而对目标系统容错机制的正确性和效率进行评估

X 收稿日期: 2004206202
XX 作者简介: 谷振宇 (1973—)，男，辽宁沈阳人，工程师

2　1　控制策略

设S 是一个可靠性系统, S_s 为无故障系统输出状态集合, $S_s = \{S_{s1}, S_{s2}, \cdots, S_{si}, \cdots\}$, $i = 1, \cdots, n$; S_f 为注入故障后系统输出状态集合, $S_f = \{S_{f1}, S_{f2}, \cdots, S_{fi}, \cdots\}$, $i = 1, \cdots, n$. 当选择的一种故障注入到系统中并被激活时, 系统当前状态和没有注入故障系统当前输出状态就会产生不一致, 这种不一致就是故障系统行为表现, 这种变换称为系统输出状态集. 如果对系统输出状态做到实时跟踪, 便可以了解故障注入到目标系统后目标系统的行为反映, 从而达到故障注入实验的目的. 无故障系统状态转换过程如图 1.

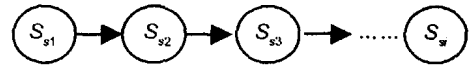

图 1　无故障系统状态转换过程

以一个故障注入系统为例, 故障注入后系统状态转换过程如图 2.

根据系统这种变化规律, 笔者在故障注入系统中引入控制策略. 控制策略的实现通过在目标系统中嵌入微内核, 微内核具有目标系统内核的基本功能, 当故障注入实验开始, 注入的故障被触发后, 激活嵌入到目标系统的微内核, 由控制策略控制故障注入实验, 直到一次完整的故障注入实验结束, 控制权转回到目标系统.

设故障注入实验的故障集为 $F\{f_1, f_2, f_3, \cdots, f_n, \cdots\}$, control- fault ( ) 为状态控制函数, fault- example- end ( ) 为实验结束标志函数, fault- example- end ( ) = 1 为实验结束, corrupt- fault ( ) 为注入函数, proce- obj ( ) 为目标控制函数, s- trans- cond ( ) 为输出状态转换函数, record- s ( ) 为输出状态记录函数. 以一次注入实验为例, 控制策略如下:

```
Intialize fault jnjection
While fault- example- end ( ) < > 1 do
      corrupt- fault (f1)
        if s- trans- cond ( ) = 1 then
          call control- fault ( )
        else call proce- obj ( )
          control- fault ( ) suspended
            record- s ( ) = record- s+ 1
              end
```

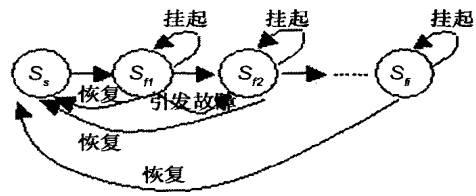

图 2　注入故障后系统状态转换过程

目标系统是星载计算机系统, 在其上运行工作负载, 即故障注入的目标, 可以是标准程序(Benchmark) 或模拟系统操作环境的合成负载. 基于软件实现的故障注入器对目标系统注入故障, 通常是把故障引入目标系统上运行的应用程序中来仿真目标系统中的故障. 虽然故障可能发生于硬件, 也可能发生于软件, 但无论发生在哪里, 最终都要表现在软件执行上. 因此从结果出发, 都可以通过最终的软件执行来仿真目标系统中发生的故障.

软件实现的故障注入大多数是用来仿真硬件故障. 故障可能出现在CPU、内存、总线或网络上, 这些故障导致软件执行错误, 如执行不正确的指令或访问不正确的数据, 因此可以通过软件执行来仿真系统发生的故障.

212　目标系统故障的仿真原理

如图 3 所示, 假设某硬件功能模块在 t_i 时刻发生了故障(t_i为注入时刻), 这个模块在 t_j 时刻(t_j为注入故障触发时刻) 第一次被软件访问. 如果该硬件模块所发生的故障已消失或者只是引起一个无效错误, 则程序继续正常的执行, 仿佛故障没有发生过一样; 如果该故障影响到程序的正常执行(例如引发指令错或计算数据错), 则系统在未来某个 t_3 时刻检测到错误. 如果做一个映射 f_a, 将 t_i 时刻发生的硬件故障 a 映射到 t_i 或 t_j 时刻程序映像 C 上, 则为了模拟 $t1$ 时刻的硬件故障 a, 只需在 t_i 或 t_j 时刻对程序映像 C 做一个改动, 令 $C = f_a(C)$, 使程序映像 C 在 t_i 或 t_j 时刻继续执行, 则程序在运行中的表现与 t_1 时刻发生故障的表现相同或相似.

对于软件故障, 由于其本身就是由软件发生, 所以更容易用软件实现. 基于软件实现的故障注入方法中执行故障注入的通用过程如下: 第一步, 以某种方式中断目标系统上运行的应用程序 (例如在

应用程序中加入中断指令或者在跟踪模式下运行应用程序);第二步,执行故障注入程序,向系统中不同位置注入错误仿真故障(例如处理器的寄存器,存储器);第三步,恢复被中断的程序,使之继续执行;第四步,收集故障在不同层次上的表现(系统层,应用程序层,容错机制层等等).

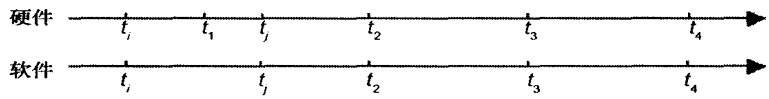

图3 故障注入时间

3 软件故障注入系统的设计

311 系统结构及各模块的功能

系统结构如图4所示,系统由控制器、故障库、故障注入器、数据收集器和分析器等模块组成.各个模块完成的功能不同,通过它们之间相互协调配合工作,完成对目标系统的故障注入实验,然后给出对目标系统的容错性能的评价.

故障注入系统需要完成下面的功能:根据一定的故障模型生成故障;对目标系统注入故障;收集有关故障对目标系统影响的信息;对结果进行分析.上述功能是由故障注入系统中各个模块相互配合共同实现的.下面介绍各个模块的功能

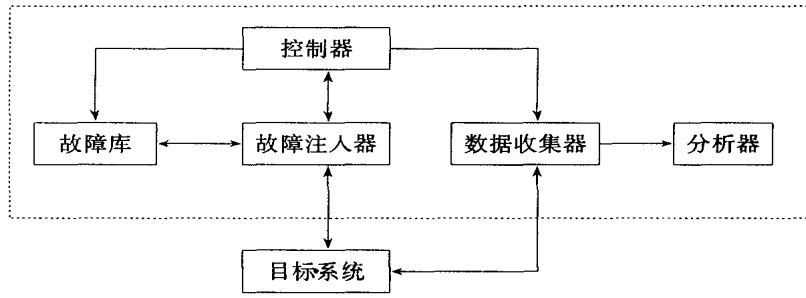

图4 软件故障注入系统

31111 控制器

控制器是整个系统的管理程序,用于控制整个故障注入实验进行的软件,可运行于目标系统或另一台计算机上.它提供用户与故障注入系统的接口,使用户可以选择故障类型、注入方式、注入地址、故障的持续时间等,也可以由机器随机生成这些故障属性.控制器还要控制故障注入器从故障库中选出故障并注入到目标系统上运行的应用程序中.另外,数据采集器对无故障运行和带故障运行的目标系统采集相关的数据也是在控制器的控制下完成的.

31112 故障库

故障库集成了各种目标系统的容错机制所针对的故障类型,它们都是依据目标系统设计的.容错计算机系统的故障可能是内部的、外部的;硬件的、软件的等等.故障库的建立需要得到目标系统所发生故障的详细情况.

31113 故障注入器

故障注入器是整个系统的核心,所谓注入,就是把故障引入到原本无故障运行的系统中.故障注入器完成的功能就是向目标系统中注入故障,要求能够保持注入故障的类型和位置等信息,并包含适当的软件或硬件逻辑以确保故障能在正确的时刻注入到正确的位置.

31114 数据采集器

数据采集器跟踪工作负载的执行,并在适当的时刻进行在线的数据采集,采集的数据将来要用于评测系统的容错性能.数据采集器是在控制器的控制下在线采集数据的,不但要采集目标系统无故障运行使得相关数据作为标准数据,还要在目标系统带故障运行时采集其带故障运行数据.

31115 分析器

分析器用来完成对数据采集器采集的数据进行处理和分析,分析器的工作通常是脱机的离线分析.通过分析计算,可以评价目标系统的容错能力及各种容错机制的性能,为进一步改进和完善目标系统

提供依据

312　目标系统仿真

目标系统以普通的 PC 机来代替星载计算机作为硬件平台, 虽然无法模拟真实的硬件环境, 但是在设计和开发故障注入系统时, 还是要考虑星载计算机的特点及所采用的容错机制　星载计算机的中央处理器只是 8086CPU, 但是该系统采用了以下一些特殊的软硬件容错机制: 采用三模冗余ö分布式处理体系结构; 存储器板带有 EDAC (Error Detect And Correct) 电路, 以信息冗余的方法来提供可靠性; 单机具有时间监视定时器 (Watchdog).

4　仿真实验

故障注入实验的目标系统是具有错误检测机制的实时操作系统 rtos, pc 工作站　错误检测机制有: 非法指令、计时器超时、越段保护和程序异常终止　实验选用的目标程序是矩阵乘 Matrix ( ), 故障类型为总线故障和寄存器故障　故障列表见表 1. 实验使用本文给出的引入智能控制的软件故障注入方法 (方法 1) 和传统的 FINE 软件故障注入方法 (方法 2) 对同一目标系统在相同条件下分别进行了故障注入实验　实验选择的故障是在表 1 所列故障类中随机选择 2000 次故障进行一次注入实验, 随机选择的故障次数见表 2　两种实验方法比较结果在表 3 中给出, 通过结果可以看出, 本文提出的故障注入方法在不开放源码的条件下, 同样可以达到对目标系统可靠性的测试, 而且十分有效

表 1　故障列表

功能模块	故障类型	故障说明
寄存器	文本段错	内容更改
	数据段错	内容更改
总线	地址线	执行错误指令
	数据线	操作数错
	控制线	执行错误控制指令

表 2　随机选择故障次数

故障类型	次数	成功注入次数
文本段错	472	431
数据段错	381	369
地址线	573	512
数据线	217	143
控制线	357	309

表 3　两种不同软件故障注入方法实验结果比较

实验方法	非法指令	越段保护	程序异常终止
11 智能控制的软件故障注入方法	0128	0171	0111
21FINE 方法	0129	0163	0114

参考文献:

[1] Stott D T, Ries G, Hsueh M C, et al11 Dependability analysis of a high2speed network using software2implemented fault injection and simulated fault [J] 1 IEEE Trans on Computers, 1998, 47 (1): 108- 119.

[2] Velazco R, Rezgui S, Ecoffet R1 Predicting error rate for microprocessor2based digital architectures through C1E1U 1 (Code Emulation Upset) Injection [J] 1 IEEE Transactions on Nuclear Science1 2000, 47 (12): 2405- 24111

[3] Benso A, Prinetto P, Rebaudengo M, et al1 A Fault Injection Environment for Microprocessor2based Boards [J] 1 IEEE International Test Conference1 1998, 31 (1): 768- 773.

[4] Triebel W A1 硬件、软件及接口技术教程 [M]1 王克义, 王钧, 方晖, 等译1 北京: 清华大学出版社, 1998.

(责任编辑: 林　敏)