基于数据流分析的软件容错策略*

刘云龙 陈俊亮

(北京邮电大学程控交换技术与通信网国家重点实验室 北京 100088)

摘要 该文就软件容错中备查点与卷回机制展开深入讨论,提出一种基于数据流分析技术的软件容错新方法.首先对软件容错进行简介,指出数据错是一切控制系统软件失效的根源与最终表现以及对数据采取强有力的容错措施的必要性.然后将数据流分析技术应用于软件容错,通过求解程序变量的到达一定值数据流方程来静态地确定任何数据在任何引用点出错时的最小充分卷回,通过求解活跃变量的数据流方程来静态地确定程序在执行各个基本块时需动态保存的变量集合,得出最小充分卷回定理与备查点数据范围定理,从而解决了时间冗余容错途径中必须回答的两个基本问题.此外,还给出了恢复块定义有效的充分条件.最后,以电信系统为应用实例,介绍了该方法的一种具体实施.该方法在简单地扩展后可被广泛应用于各种容错软件的设计中.

关键词 软件容错,数据流分析,最小充分卷回,备查点数据范围.

中图法分类号 TP311

大多数实时控制应用领域对系统的稳定性(Reliability)、可用性(Availability)及安全性(Safety)都有较高的要求,如电信、金融、交通、航空及航天等系统,其中的任何故障都有可能导致重大的经济损失或人员伤亡,所以,在其设计与实现中如何保证系统在出现故障时维持正常的服务,即如何进行系统容错是一个至关重要的问题.时至今日,硬件容错技术已日趋成熟,Tandem,Stratus,Motorola,IBM 等厂商的商用容错(或高可用性)计算机已被广泛应用.相比之下,软件容错虽然在近 20 年得到了广泛的关注与研究,但由于软件自身所具有的应用语义因目标域的不同而不同,使得软件容错的途径方法更具个异性,从而在理论基础与实用程度上均未能达到硬件容错的水平.本文的工作就是基于这个技术背景,针对软件容错展开的.文中论述立足于软件(包括操作系统和应用软件),而忽略下层硬件的特性.

我们认为,软件容错 SFT(software fault tolerance)是在软件出现错误时防止发生系统失效的一种以软件实现的技术.而软件错误按其表现形式,可分为柔性错误(Soft Fault)与固性错误(Solid Fault)两类.柔性错误是那些受各种微妙的环境因素的影响所导致的软件错误,具有瞬态、偶发性、可恢复的特征,也可称为瞬时错误(Transient Fault);而固性错误则是那些由软件内嵌式(Built-in)的缺陷(Bug)所致的软件错误,具有永久性、重复出现、不可恢复的特征,也可称为永久错误(Permanent Fault).文献[1]中的有关研究表明,柔性错误的发生概率要高于固性错误的发生概率,是其 10~30 倍.另据文献[2]中有关 Tandem/GUARDIAN 容错操作系统的测试报告,在成熟软件中,柔性错误约占全部软件错误的 80%以上,由此可见,对柔性错误的防范与容错应被视为软件容错的主要目标.总结现有的研究成果,软件容错的途径可被概括为以下两大类.

(1) 软件冗余(Software Redundancy)[2~5] 这种途径有以下 3 种实施方法:恢复块(Recovery Block)、多版本编程 NVP(N-version programming)和多版本自检编程 NSCP(N-selfchecking programming).这 3 种方法均是基于"对同一功能由不同人以多算法实现,算法之间互补容错"的思想,对柔性和固性错误都有容错效果,但软件开发投入较大,目前的应用例基本在可靠性要求极高而应用目标域又较为复杂,难以通过模拟测试等手段排除软件错误,且绝对不允许系统失效发生的领域,如航天、航空、火控等高危性计算系统(Critical Computation Systems).

(2) 时间冗余(Time Redundancy)[2,3,6,7] 这种途径主要是基于"失败后重做(Retry-on-Failure)"的思想.它采用备查点与卷回(Checkpointing & Rollback)机制,若与 Watchdog,Selfchecking 等技术结合起来使用,能很好地对柔性错误起到检错与容错的作用,但对固性错误,则只有检错能力而无容错作用.随着软件避错(Software Error

---

* 本文研究得到国家教委博士学科点专项科研项目基金资助.作者**刘云龙**,1972 年生,博士,主要研究领域为智能网,软件容错,形式化方法等.**陈俊亮**,1933 年生,教授,博士导师,中国科学院院士,主要研究领域为智能网,通信软件.

本文通讯联系人:陈俊亮,北京 100088,北京邮电大学程控交换技术与通信网国家重点实验室

本文 1997-04-07 收到原稿,1997-06-19 收到修改稿

Avoidance)技术与软件测试(Software Testing)技术的逐渐成熟,近年来,人们的研究兴趣越来越多地集中到对柔性错误有良好容错作用的时间冗余途径上来.此途径具有软件容错的附加投入小、易于软件开发与维护的显著优点,目前已广泛被高危性计算系统之外的其他高可靠性应用领域所采用.

一个控制系统出现故障后对环境产生的不利影响主要来自于它将产生错误的输出(数据),从而导致它所控制的目标域进入异常的状态.* 在以有限自动机FSM(finit state machine)为模型的软件系统(如电信系统软件)中,事件(数据)驱动是自动机运作的唯一有效途径,所以,在这样的系统中,数据直接影响着程序的控制流与运算逻辑,而针对其数据采取强有力的容错措施便显得极为重要.本文首次将数据流分析技术引入到软件容错的领域,通过求解程序变量的到达一定值数据流方程来静态地确定任何数据在任何引用点出错时的最小充分卷回(定义见下节),通过求解活跃变量的数据流方程来静态地确定程序在执行各个基本块时需动态保存的变量集合,这就为软件容错中恢复块的定义、备查点的插入及需检测的数据范围等关键问题的解决提供了理论依据.

1 数据流分析技术在软件容错中的应用

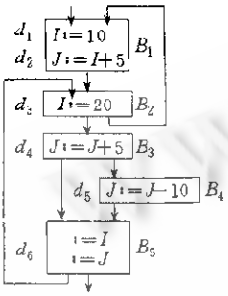

图1 例程流图

分析程序中所有变量的定值(指对变量赋值或输入值)和引用之间的关系的过程叫作数据流分析DFA(data flow analysis).[8]下面,我们将结合文献[8]中给出的一个简单例子来阐述数据流分析技术在软件容错中的应用.考察图1的程序流图,列出此流图的到达一定值数据流方程[8]

$$\begin{cases} OUT[B]=(IN[B]-KILL[B])\cup GEN[B]; \\ IN[B]=\cup OUT[p], P\in P[B]. \end{cases}$$

其中$OUT[B]$代表到达基本块B出口之后的各个变量的所有定值点集合;$IN[B]$代表到达基本块B入口之前的各个变量的所有定值点集合;$GEN[B]$代表基本块B所"生成"的定值点集合;$KILL[B]$代表基本块B所"注销"的定值点集合;$P[B]$代表基本块B的所有前驱基本块的集合.

$GEN[B]$和$KILL[B]$均可由流图直接求出,然后用迭代法求解此方程组,按各结点的深度为主次序(由前向后),依次可计算此例程各基本块的$IN[B]$和$OUT[B]$,计算结果见表1.

表1 例程$IN[B]$与$OUT[B]$的计算结果

基本块	$IN[B]$	$OUT[B]$
B_1	$\{d_2,d_3,d_4,d_5\}$	$\{d_1,d_2\}$
B_2	$\{d_1,d_2,d_3,d_4,d_5\}$	$\{d_2,d_3,d_4,d_5\}$
B_3	$\{d_2,d_3,d_4,d_5\}$	$\{d_3,d_4\}$
B_4	$\{d_3,d_4\}$	$\{d_3,d_5\}$
B_5	$\{d_3,d_4,d_5\}$	$\{d_3,d_4,d_5\}$

表2 ud链计算结果

基本块	ud链
B_1	$ud(I,d_2)=\{d_1\}$
B_2	无
B_3	$ud(J,d_4)=\{d_2,d_4,d_5\}$
B_4	$ud(J,d_5)=\{d_4\}$
B_5	$ud(I,d_6)=\{d_3\}$;$ud(J,d_7)=\{d_4,d_5\}$

利用表1中的结果,我们可以根据引用一定值链(简称ud链,对于某变量A的某个引用点u,称能到达u的A的所有定值点的集合为A在引用点u的ud链[8],记为$ud(A,u)$)的计算规则求出变量I和J在各个引用点的ud链,结果见表2.

我们认为,计算变量的ud链对于软件容错意义重大.ud链不仅记录了数据在引用点的所有定值点的集合,同时它也揭示了数据在该引用点出错时的最小充分卷回.下面给出相关的定义与定理.

定义1. 最小充分卷回MSR(minimal sufficient rollback).在程序一次执行的迹中,对于数据A的某个引用点u,我们称点u之前最近一个A的定值点v到点u之间的程序语句(点)序列为数据A在引用点u的最小充分卷回,记为$MSR(A,u)$.

定义2. 最小充分卷回点MSRP(minimal sufficient rollback point).在程序1次执行的迹中,对于数据A的某个

---

* 一般来说,控制系统对环境的不利影响来自两方面:输出的正确性与时效性.输出时效性有两方面含义:①系统在某时段本应该产生一个输出,但实际上没有;②系统在某时段本不应该有输出,但实际却产生了某个输出.我们认为,输出的时效性是正确性的一种特例,时效性含义一方面可以解释为:正确输出O发生畸变而成为空输出∅;另一方面可以解释为:正确输出∅发生畸变而成为错误输出O.基于以上论述,本文将控制系统出现故障后对环境产生的不利影响归结为一个根源:错误的输出.

引用点 u，我们称点 u 之前最近一个 A 的定值点 v 为数据 A 在引用点 u 的最小充分卷回点，记为 $MSRP(A,u)$.

定理 1. 在时间冗余的软件容错途径中，若程序的一次执行中，数据 A 在其某个引用点 u 被发现出错，则恢复此错误的最弱充分条件为程序卷回到数据 A 在 u 的最小充分卷回点 $MSRP[A,u]$.

$$MSRP(A,u)=V|_{(v\in ud(A,u))\wedge[\forall w,wd(A,u):|v,u|=\mathrm{MIN}(|w,u|))]}. \tag{1}$$

证明：由 ud 链的定义及反证法易证(略). □

式(1)中，$|v,u|$ 叫作点 v,u 的模，指本次执行的迹中点 v 到点 u 的距离. 上式说明的是某变量 A 在某引用点 u 的最小充分卷回点为其 ud 链中对于本次执行与出错点 u 距离最近的那个定义点；如果 u 不是 A 的引用点，则 $MRP(A,u)=\varepsilon$，ε 为空点.

请看图 2 所示的例程一次执行的迹 T，若程序执行到点 d_6 时对数据 I 的引用出错，则程序卷回到变量 I 在引用点 d_6 的最小充分卷回点 $MSRP(I,d_6)$ 就是以恢复此数据错，因为 $ud(I,d_6)$ 中只有一个元素 d_3，所以，$MSRP(I,d_6)$ 就是 d_3；若程序执行到点 d_7 时对数据 J 的引用出错，而 $ud(J,d_7)=\{d_4,d_5\}$，则由定理 1 可求出 $MSRP(J,d_7)=d_5$，进而可得此时程序卷回到数据 J 的定义点 d_5，便足以容错.

由此可见，ud 链的计算对于软件容错中卷回范围的静态确定具有极其重要的指导意义. 此外，对于采用恢复块的容错机制，我们设应用目标域的关键数据(即关系到本恢复块 R 所要完成的事物处理的安全性、正确性及一致性的数据)集合为 Ω，则我们可进一步给出定理 1 的引理.

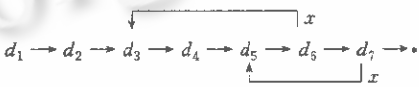

图 2　例程的一个迹 T

引理. 恢复块的定义有效的充分条件为：对于目标容错数据集 Ω，其中的所有数据的所有引用点被包含在所设计的恢复块 R 中，且这些数据在这些引用点的最小充分卷回点亦被包含在此恢复块 R 中，即

$$\forall\, X;\Omega,\forall\, u;USE(X).\,[(u\in R)\wedge(MSRP(X,u)\in R)^{-}. \tag{2}$$

式(2)中 $USE(X)$ 是数据 X 的所有引用点集合.

证明：由定理 1 易证(略). □

上面考察了例程流图的到达一定值数据流方程，下面再考察一下此流图的活跃变量数据流方程.

$$\begin{cases} L\_IN[B]=(L\_OUT[B]-L\_DEF[B])\cup L\_USE[B];\\ L\_OUT[B]=\cup L\_IN[s],s\in S[B]; \end{cases}$$

相关符号定义如下：$L\_IN[B]$ 代表基本块 B 入口之前的活跃变量集合；$L\_OUT[B]$ 代表基本块 B 出口之后的活跃变量集合；$L\_DEF[B]$ 代表基本块 B 中定义的，但定值前未曾在 B 中引用过的变量集合；$L\_USE[B]$ 代表基本块 B 中引用的，但引用前未曾在 B 中定值的变量集合；$S^{-}[B]$ 代表基本块 B 的所有后继基本块的集合；$DEF[B]$ 代表基本块 B 中定义的全部变量集合.

从流图可直接求出 $L\_DEF[B]$，$L\_USE[B]$ 及 $DEF[B]$，然后用迭代法求解此方程组，按各结点的深度为主次序的逆序(由后向前)，依次可计算此例程各基本块的 $L\_IN[B]$ 和 $L\_OUT[B]$，结果见表 3.

表 3　$L\_IN[B]$ 与 $L\_OUT[B]$ 的计算结果

基本块	$L\_DEF[B]$	$L\_USE[B]$	$L\_IN[B]$	$L\_OUT[B]$	$DEF[B]$	$L\_OUT[B]\cap DEF[B]$
B_1	{ }	{I,J}	{I,J}	{J}	{I,J}	{J}
B_2	{ }	{J}	{I,J}	{I,J}	{I}	{I}
B_3	{ }	{J}	{I,J}	{I,J}	{J}	{J}
B_4	{I}	{ }	{J}	{I,J}	{J}	{J}
B_5	{I,J}	{ }	{ }	{J}	{ }	{ }

表 3 中的 $L\_OUT[B]$ 指示的是离开基本块 B 后，哪些变量的值在 B 的后继中还会被引用；而 $DEF[B]$ 指示的是基本块 B 中对哪些变量做了定值，所以，$L\_OUT^{-}[B]\cap DEF[B]$ 就指示出由基本块 B 定值，且将被其后继所引用的变量集合，这也正揭示了基本块 B 为了后继的引用点出错时能成功地进行恢复而必须做备份的数据集合. 由以上论述可得定理 2.

定理 2. 为了在引用点出错时成功地对目标数据集 Ω 中的数据进行卷回与恢复，则每个基本块 B 必须对 $L\_OUT[B]\cap DEF[B]\cap\Omega$ 中的所有数据进行合法性检测与备份保存.

证明:(反证法).

假设1. 基本块B不必对某一数据$\alpha\in L_OUT[B]\cap DEF[B]\cap\Omega$进行备份.因为$\alpha\in L_OUT[B]\cap DEF[B]$,所以,数据$\alpha$在$B$内被定值,且此定值在$B$之后必存在某个引用点$\mu$,这时,如果数据$\alpha$在$B$之后且$\mu$之前发生畸变,则在$\mu$发现错误后的卷回操作无法恢复数据$\alpha$在$B$内的定值,进而程序恢复后,在引用点$\mu$将对数据$\alpha$发生错误引用,故卷回失败,所以假设1不成立;

假设2. 基本块B对$L_OUT[B]\cap DEF[B]\cap\Omega$中的所有数据进行备份,但备份前不必对某一数据$\alpha\in L_OUT[B]\cap DEF[B]\cap\Omega$进行合法性检查.因为$\alpha\in DEF[B]$,所以数据$\alpha$在$B$内必存在定值点$v$,这时,如果程序在$v$之后且$B$结束之前的某一时刻出错,而导致数据$\alpha$的值发生畸变,则基本块$B$保存的$\alpha$值将为畸变值,又因为$\alpha\in L_OUT[B]$,所以,数据$\alpha$在$B$之后必存在引用点$\mu$,则引用点$\mu$将对数据$\alpha$发生错误引用,此时即使卷回,恢复后,数据$\alpha$的值依然为畸变值,故卷回失败,所以,假设2亦不成立.

由以上两个假设不成立,可总结出定理2正确. □

定理1和定理2分别从两个不同的侧面对数据的容错作了约束.定理1规定了一个数据在其某个引用点出错时的最小充分卷回范围(某个备查点).定理2规定了为成功地实施定理1规定的卷回所要预先在备查点检测与备份的数据集合.可以说,这两个定理回答了软件容错中的两个最基本的关键问题,即备查点的插入及需要检测的数据范围问题.最后值得说明的是,在本方法的具体应用中,基本块的概念可以被实施于任何满足"单入口单出口"条件的程序体,可以是一个顺序程序段,也可以是一个过程或函数以及进程的一个状态等等.

2 在电信系统中的应用

电信系统可以被抽象为若干电信业务Service的一个闭包:$TeleSys::=Service^*$;而每个电信业务的行为又可以用扩展的有限状态自动机(EFSM)来描述.

$$Service::=\langle D,S,I,O,s_0,\delta\rangle$$

其中D为业务的全局数据;S为自动机的状态集合;I为自动机的输入事件集合;O为自动机的输出事件集合;s_0为自动机的初始状态;δ为映射函数:$S\times I\to S\times O^*$.

对于一个状态$s\in S$,它可被形式化地描述为$s::=(D',I',O',S',\delta')$,其中$D'\subseteq D$为在状态$s$中引用的全局数据集合,映射$\delta':s\times I'\to S'\times O'$.

基于以上描述,我们可以对目标电信系统的每个电信业务进行数据流分析,分析中以有限自动机的状态做基本块进行处理,同时把输入事件均列入本状态的定值数据集合$DEF[s]$,把输出事件均列入本状态的后继引用数据集合$L_OUT[s]$,这样,就可以按照第1节给出的计算方法,得到一个电信业务在其所有状态下的$L_OUT[s]\cap DEF[s]\cap\Omega\cap D'$.这里值得注意的是,在像电信系统这样的分布式系统中,由于每个实体与环境的实时交互普遍存在,致使孤立实体软件容错的卷回策略往往无法实施,而全局卷回策略又易于引发多米诺效应而导致更严重的系统失效.但在我们的方法中不会产生这个问题,因为我们实施的卷回,并非程序真正地回退到前面的备查点,而只是把出错的数据恢复到前面的备查点处的定值,这就避免了分布式系统中由于卷回而可能导致的各种隐患(多米诺效应、各分布实体间语义不一致等等).具体作法如下:对业务进行数据流分析,求出每个状态s的$L_OUT[s]\cap DEF[s]\cap\Omega\cap D'$,这样,在业务执行中,当进入状态$s$时,首先要对$D'[s]\cup I'[s]$中的数据进行合法性检测(这是激活状态$s$的充分条件).若正确,就把$D'[s]\cup I'[s]$备份到$CP_s$.否则,若出错数据属于$D'[s]$,则从备查点$CP_D$(见下文)中恢复该数据的正确定值后继续,若出错数据属于$I'[s]$,则业务执行失败(此时为外来的错误传播,亦可由业务逻辑要求错误消息源进行重发);在离开状态s前,要对$L_OUT[s]\cap DEF[s]\cap\Omega\cap D'$中的数据进行合法性检测,若正确就做备份到$CP_D$,否则,需对$DEF[s]$进行重新定值(卷回到$CP_s$由备用恢复块重做,因$CP_s$中已包含了激活状态$s$的充分条件$D'[s]\cup I'[s]$,故此卷回对自动机的环境透明).这样,整个系统在任何时刻只需动态维护两个备查点CP_D与CP_s,就能够有效地实施出错后的卷回恢复.

3 结束语

软件的失效从根本上讲都源于数据的畸变,最终的表现形式亦将是数据的畸变,所以,科学地对数据进行研究,并找出合理的容错方案是实施软件容错的一个重要思路.本文首次将数据流分析的方法应用于软件容错,得出了最小充分卷回定理(定理1)与备查点数据范围定理(定理2),并作为最小充分卷回定理的一个引理,给出了恢复块定义有效的充分条件,这些对时间冗余途径中备查点的插入与软件冗余途径中恢复块的设计都具有指导意义.

此外，在本文提出的数据容错策略中，理论上，所有数据错误的潜伏期(Error Lantency)都不会超过其定值后的第1个引用点，但这是依赖于理想的数据合法性检测机制的．一般说来，数据的语法检测(数据越界、无效指针、被除数为零等)比较容易，而语义检测则困难得多．如何高速、有效地进行数据合法性检测是今后需要解决的问题．实际上，在数据合法性检测机制不完善的前提下，分布式系统中结点间的信息交互将很可能导致数据错误的传播与扩散．此时，为了恢复某一个结点内的错误，就可能需要其他若干结点(直至整个系统)随其一起进行卷回，才能恢复系统数据的一致性，而在这种情况下，分布式系统的全局卷回策略及可能造成的多米诺效应的解决则不可避免地成为必须进一步研究的难题．

参考文献

1 Siewiorek D F, Swarz R S. The theory and practice of reliable system design. Bedford: Digital Press, 1982

2 Lyn M R. Software fault tolerance. New York: Wiley & Sons Ltd., 1995

3 Johnson B W. Design and analysis of fault tolerance digital systems. Menlo Park, California: Addison-Wesley Publishing Company, 1989

4 Avizienis A, Chen L. On the implementation of N-version programming for software fault tolerance during execution. In: Proceedings of the International Conference on Computer Software and Applications. New York: IEEE Press, 1977. 149～155

5 Avizienis A, Kelly J P. Fault tolerance by design diversity: concepts and experiments. IEEE Computer, 1984,17(8):67～80

6 Huang Y, Jalote P, Kintala C M R. Two techniques for transient software error recovery. Hardware and Software Architectures for Fault Tolerance: Experience and Perspective, Lecture Notes in Computer Science, Springer Verlag, 1994. 159～170

7 Chandy K M, Ramamoorthy C V. Rollback and recovery strategies for computer programs. IEEE Transactions on Computer, 1972,C-21(2),137～146

8 陈火旺，钱家骅，孙永强．编译原理．北京：国防工业出版社，1984．253～256
(Chen Huo-wang, Qian Jia-hua, Sun Yong-qiang. Compiling Principles. Beijing: National Defense Publishing House, 1984. 253～256)

A DFA-based Approach for Software Fault Tolerance

LIU Yun-long　CHEN Jun-liang

(National Laboratory of Switching Technology and Telecommunication Networks
Beijing University of Posts and Telecommunications　Beijing　100088)

Abstract　In this paper, the checkpointing & rollback mechanism is studied deeply, and a new approach for SFT (software fault tolerance) is presented, which is based on the DFA (data flow analysis). The authors introduce the SFT technology at first, and point out that the peculiarity of data is the ultimate reason and also the final result of the software faults in control systems, so it is very necessary to adopt a powerful measure for data fault tolerance. Then, they discuss the applications of the DFA technique in the SFT in details, and give two theorems, one is for the minimal sufficient rollback point and another is for the checkpoint data set. Besides, they give the sufficient condition for the validation of the definition of a rollback block. They also use the telecommunication system as an example to illustrate the usage of this method. It is shown that only two checkpoints are needed to be maintained dynamically in their solution. The method presented in this paper can be used widely by extending simply.

Key words　Software fault tolerance, data flow analysis, minimal sufficient rollback, checkpointing data set.