程序控制流检测算法优化

李剑明[1] 谭庆平[2] 徐建军[2] 尹 胜[3]
(75130部队133分队 贵港 537103)[1] (国防科技大学计算机学院 长沙 410073)[2]
(75130部队政治部 贵港 537103)[3]

摘 要 暴露在太空辐射环境下的星载计算机,其电子元器件可能因受到高能带电粒子的轰击而造成硬件系统的瞬时故障,所以,需要使用容错技术来提高其可靠性。对一种面向硬件瞬时故障的纯软件控制流检测算法RSCFC(Relationship Signatures for Control Flow Checking)进行了有效的改进,通过对标签S进行分段编码,克服了原算法中存在的待加固程序的基本块总数受机器字长限制的问题,并给出了具体的计算证明。计算结果表明,如果机器字长为64位,那么改进后的算法在设定前提下能单层加固程序的最大基本块数可以超过2^{18}。与RSCFC相比,改进后的算法在加固基本块总数比较大的程序时,性能开销明显降低而且检错能力保持不变。

关键词 瞬时故障,数字标签,控制流检测,软件容错
中图法分类号 TP306+.2 **文献标识码** A

Optimization of Control Flow Checking Algorithm

LI Jian-ming[1] TAN Qing-ping[2] XU Jian-jun[2] YIN Sheng[3]
(133 Detachments of 75130 Army, Guigang 537103, China)[1]
(College of Computer, National University of Defense Technology, Changsha 410073, China)[2]
(Special Branch of 75130 Army, Guigang 537103, China)[3]

Abstract In the space environment, the electrical circuses of computer are often subject to hardware transient faults, which are caused by high energy neutrons from cosmic rays. It is necessary to utilize appropriate fault tolerance techniques for improving the reliability of space application. This paper proposed an enhanced algorithm based on RSCFC (Relationship Signatures for Control Flow Checking), which is a control flow checking approach for hardware transient faults. In RSCFC, the sum of basic blocks is limited by the length of machine word. Through the segmented encoding of signatures, the optimized method solves the problem effectively. The analytical results indicate that the maximal number of basic blocks is extended to 2^{18} when the length of machine word is 64 bits. Compared with RSCFC, the overhead of performance and memory is decreased evidently in our algorithm, and the faults detecting capability remains.

Keywords Transient faults, Software signatures, Control flow checking, Software tolerance

计算机系统的硬件并不是完全可靠的,特别是系统暴露在充满宇宙射线和各种高能带电粒子的太空环境中时很可能发生硬件故障。因为在半导体电路中是通过保持或者释放一定的电量来表示逻辑上的1和0,当系统处于这种恶劣环境时,半导体电路中的PN结可能会被各种高能带电粒子轰击而瞬时充放电,导致其中存储的电量发生变化,从而使所存储的信息发生逻辑状态上的翻转,比如原先存储的1翻转成0。瞬时故障通常指的就是处理器以及计算机系统内部一位信息的翻转,被称为单地址翻转(Single Event Upset,SEU)[1]。研究表明,计算机系统中80%~90%的失效都是由瞬时故障引起的[2,3]。这种故障的影响是暂时的,持续时间短暂,且并没有损坏内部硬件电路,但是它却可以通过改变处理器状态或寄存器存储值等方式影响程序的正常运行,甚至有可能导致系统崩溃。

在程序的执行过程中,当SEU故障破坏指令的操作数部分时,就会造成程序的数据流错误;当SEU故障破坏的是指令的操作码部分时,就会造成程序的控制流错误。各种故障注入的结果表明,瞬时故障引起的控制流错误将占错误总数的33%~77%[4]。在航天计算和交通实时控制等安全攸关的情况下,需要用容错技术对硬件系统进行加固以提高其可靠性。目前对控制流错误进行检测的方法主要包括通过硬件实现和通过软件实现两种。在用硬件实现的方法中,典型例子是三部件冗余(TMR)[5,6],这种技术通过比较处理器的3个结果,并选取两个相同的结果作为最终结果。Watchdog[7,8]技术采用辅助的专用处理器来检测总线上数据的正确性。用硬件实现的缺点是需要对原硬件体系结构进行修改,成本和开发周期都相应大大增加。现有的针对控制流的软件容错方法大多是采用数字标签分析(Signature Analysis)的办法,在这些

李剑明(1981-),男,硕士,助理工程师,主要研究方向为计算机系统软件,E-mail:just308@126.com;**谭庆平**(1965-),男,博士,教授,主要研究方向为计算机系统软件;**徐建军**(1978-),男,博士,讲师,主要研究方向为计算机软件工程;**尹 胜**(1980-),男,学士,助理讲师,主要研究方向为计算机软件工程。

方法中首先要将原程序代码分成各个无分支基本块[9]，具有代表性的是 Stanford 大学 CRC 实验室提出的 CFCSS(Control Flow Checking by Software Signatures)[10]，CFCSS 是一个纯软件的指令级方法，具有较好的控制流检错能力，但也存在着匿名问题。Princeton 大学的 Liberty 研究小组提出 SWIFT(Software Implemented Fault Tolerance)[11]算法，在该算法中关于控制流的处理部分与 CFCSS 类似。法国 TIMA 实验室提出的 DSM[12]算法可以检测出由 SEU 引发的所有控制流错误，但是付出的代价是性能下降 3 倍，内存消耗增加 4 倍。

本文对控制流检测算法 RSCFC(Relationship Signatures for Control Flow Checking)[13]的工作原理及其优、缺点进行了深入的分析，并对算法中存在的待加固程序的基本块总数受机器字长限制的问题进行了有效的改进。

本文第 1 节对 RSCFC 技术进行论述；第 2 节对 RSCFC 算法进行改进；最后给出结论。

1 RSCFC 算法

1.1 RSCFC 算法的基本块标签设置

RSCFC 算法为程序的每个基本块设置两个数字标签 S 和 L。假设程序共有 n 个基本块(记为 $V_1,\cdots,V_n$)，那么 S 和 L 均为字长为 $n+1$ 的二进制数。

对任一基本块 V_i：

S 标签(记为 S_i)的构造方法如下：

S_i 的最高位(从右端数起第 $n+1$ 位)恒为 1。

如果 V_j 是 V_i 的合法的后续基本块，即程序存在一条从 V_i 跳转至 V_j 的指令，那么 S_i 从右端数起第 j 位为 1，否则，该位为 0。

L 标签(记为 L_i)的构造方法如下：

L_i 从右端数起第 i 位为 1，其他位均为 0。例如，假设程序共有 10 个基本块(记为 $V_1,\cdots,V_{10}$)，结点 $suc(V_3)=\{V_1,V_6,V_9\}$，那么，$S_3=10100100001$，$L_3=00000000100$。

1.2 RSCFC 算法的检错原理

RSCFC 算法在每个基本块的前端和后端分别插入一条 TEST 断言和 SET 断言。TEST 断言放在基本块的前端用来检测是否发生控制流错误，SET 断言放在基本的后端用来把变量 S 的值更新成当前基本块的 S 标签，如图 1 所示。

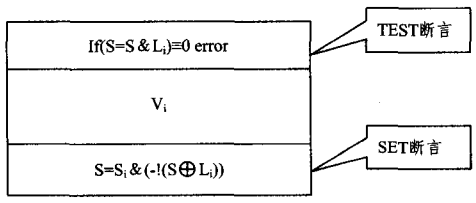

图 1 RSCFC 基本块结构

TEST 断言：if($S=S\&L_i$)≡0 error，假设结点 V_j 是结点 V_i 的前趋结点，则在 TEST 断言中通过用结点 V_j 的 S 标签和当前结点 V_i 的 L 标签进行“按位与”运算，并把运算后的结果赋给变量 S。根据前述的 S 标签构造算法，由于 V_i 是 V_j 合法的后续结点，S_j 从右端数起第 i 位为 1，因此 $S\&L_i\neq 0$；反之，若 V_i 不是 V_j 合法的后续结点，则 S_j 从右端数起的第 i 位必为 0，所以 $S\&L_i=0$，由此即可检测出控制流错误。

SET 断言：$S=S_i\&(-!(S\oplus L_i))$，用来对变量 S 进行更新。当 TEST 断言中没检测出错误时，$S=S_j\&L_i=L_i$；在执行 SET 断言时 $S\oplus L_i=0$，对 0 取逻辑反，则变成 1，再对 1 取负，则成为 -1，-1 用二进制补码表示为全 1，所以 -1 和 S_i 进行“按位与”运算后变量 S 的值就是 S_i。用这个公式设置变量 S 的目的是可以检测出非法分支跳转到基本块内部的控制流错误。例如，当从 V_j 跳转到 V_i 的基本块内部时，位于 V_i 前面的 TEST 断言被忽略，变量 S 的值仍然是 S_j，因为 S_j 的最高位是 1，而 L_i 的最高位为 0，所以这时无论 V_i 是否是 V_j 的后续结点，变量 S 经过 $S\oplus L_i$ 运算后一定不会为 0，再对其取逻辑反，变量 S 就变成 0，对 0 取负还是为 0，而 S_i 与 0 进行“与”运算后还是为 0，接着无论从 V_i 跳转到任何结点，都会在结点的 TEST 断言中检测出控制流错误。

在 RSCFC 算法中，如果 TEST 断言和 SET 断言都是原子执行，那么它就可以检测出所有的控制流错误。但是算法中标签 S、L 的长度受到机器字长的限制。例如，在 32 位机器中，因为标签 S 最高位总是置成 1，算法最多只能检测含有 31 个基本块的程序。RSCFC 算法通过分组基本块来解决这个问题。在每个基本块组的内部，可独立应用 RSCFC 方法，而它们对于其它基本块组而言，可被看作一个“基本块”来参与外部大范围的 RSCFC 的应用。即基本块组内部各结点形成内层检测，而以各个基本块组为独立单位形成外层检测。但是这种内外层嵌套的方法在程序基本块总数比较大的情况下具体实施起来比较复杂，需要解决很多细节上的问题，比如如何合理地进行基本块分组，如何实现从外层检测到内层检测的转换，这些问题的解决将付出很大的内存开销和时间开销。为此本文将在下一节对 RSCFC 算法进行改进，以解决其中存在的待测程序的基本块总数受机器字长限制的问题。

2 RSCFC 改进

我们知道在 RSCFC 算法中，对于一个 32 位机器，若要单层实现算法，则待加固程序的基本块数最多不能超过 32−1=31 个，通过改进 RSCFC 算法中基本块标签设置方法可以显著地提高算法可以处理的待加固程序的基本块数。

我们可以把 RSCFC 算法中的 S 标签采用分段的方式进行设置，如图 2 所示，可以把 V_0 的 S 标签中有效的 31 位分成 4 个段，第一段 8 位、第二段 8 位、第三段 8 位、第四段 7 位，每个段对应着 V_0 的一个后续结点，第一段对应着 V_0 后续结点中的 V_1，第二段、第三段和第四段分别表示后续结点 V_2、后续结点 V_3 和后续结点 V_4。

在程序编译阶段，我们要为程序控制流图中的每个结点设置一个唯一的编号 i($i=0,1,2\cdots,SUM-1$)，SUM 为程序控制流图中的结点总数。我们把程序控制流图中不拥有任何前趋结点的结点称为程序的始结点，不拥有任何后续结点的结点称为程序的终结点，把程序的始结点编号为 0。

然后要为程序控制流图中的每个结点(始结点除外)设置一个位置信息值，用来表示每个结点应该用标签 S 的哪一段表示，如结点 V_i 的位置信息值为 1，说明编号为 i 的结点应该用前趋结点的 S 标签的第一段表示。由于在结点总数为 SUM 的程序控制流图中，只有当其中的一个结点是其余 SUM−1 个结点的前趋结点时，控制流图中所有结点的最大位置信息值才会达到 SUM−1，所以，结点位置信息值的所有可能取值为 $1,2,\cdots,SUM-1$。我们通过以下的方法设置程

序控制流图中各个结点的位置信息值。所有结点的位置信息值都初始化成为 0。

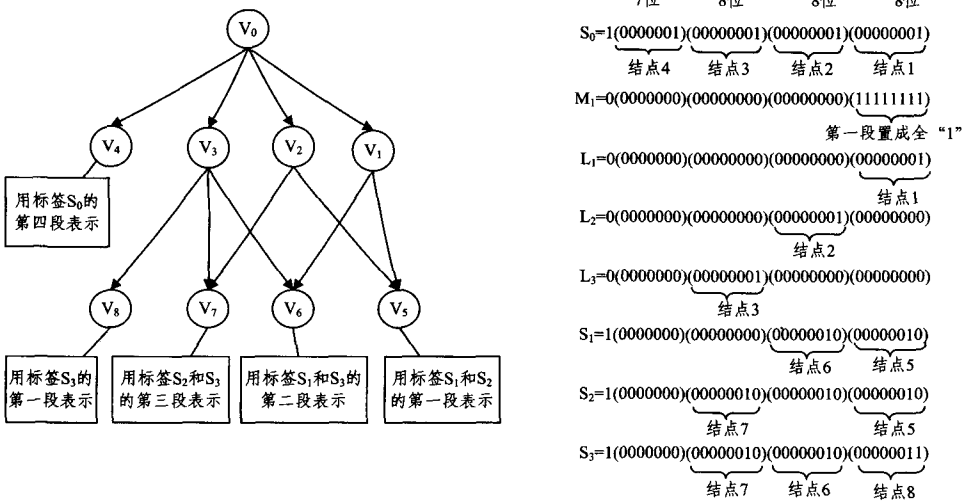

图 2 改进后标签的设置方法

根据结点编号按从小到大的顺序把控制流图中的结点 $V_i(i=1,2\cdots,SUM-1)$ 的位置信息值设置为：结点位置信息的所有可能取值(1,2,…,SUM−1)中还未被与结点 V_i 拥有相同前趋结点的所有结点的位置信息值占用的最小值。

通过以上方法对所有结点设置位置信息值后，我们把所有位置信息值中最大的那个位置信息值记为 MAX，并把标签 S(最高位除外)相应地分成 MAX 段。将程序控制流图中最大的后续结点个数记为 max，由于每个后续结点都必须用标签 S 中的一段表示，因此就有 MAX≥max。

如图 2 所示，程序控制流图中结点总数为 9，所以结点位置信息值的所有可能取值为：1,2,…,8。由于 V_0 是始结点，因此从 V_1 开始为控制流图中的每个结点设置位置信息值。设置 V_1 的位置信息值时，与 V_1 拥有相同前趋结点的结点有 V_2、V_3 和 V_4，此时 V_2、V_3 和 V_4 的位置信息值都是为 0，所以，结点位置信息的所有取值(1,2,…,8)中还未被与结点 V_1 拥有相同前趋结点的所有结点的位置信息值占用的最小值为 1，那么将 V_1 位置信息值设置为 1。按相同的方法，V_2、V_3、V_4、V_5、V_6、V_7 和 V_8 的位置信息值分别设置为 2、3、4、1、2、3 和 1。所有位置信息值中最大的那个值 MAX 为 4，所以，把标签 S(最高位除外)相应地分成 4 段。

根据 RSCFC 算法的检错原理，对需要用标签 S 第一段进行编码的所有结点，可以从编号最小的结点开始，依次从 1 重新编号。对需要用标签 S 第二段、第三段、…、第 MAX 段编码的结点，采用同样的方法重新进行编号。

例如，根据上面对图 2 中所有结点设置位置信息值后，需要用标签 S 第一段表示的结点有 V_1、V_5 和 V_8，其重新编号后的新编号分别为 1、2 和 3。需要用标签 S 第二段表示的结点有 V_2 和 V_6，其重新编号后的新编号分别为 1 和 2。需要用标签 S 第三段表示的结点有 V_3 和 V_7，其重新编号后的新编号分别为 1 和 2。需要用标签 S 第四段表示的结点有 V_4，其重新编号后的新编号为 1。

控制流图中的每个结点重新编号后，就可以为结点 V_i 设置 S、L 和 M 标签。程序的始结点没有 L 和 M 标签，程序的终结点没有 S 标签。

V_i 的 S 标签(记为 S_i)的构造方法如下：

S_i 的最高位恒为 1。

对于标签 S_i 的第 $m(m=1,2,\cdots,MAX)$ 段，如果结点 V_i 中有需要用该段进行编码的后续结点，那么 S_i 从右端数起第 m 段内的数值为该后续结点的新编号的二进制值，否则该段数值为 0。

V_i 的 L 标签(记为 L_i)的构造方法如下：

如果 V_i 需要用前趋结点的 S 标签的第 $m(m=1,2,\cdots,MAX)$ 段进行编码，那么 L_i 从右端数起第 m 段内的数值为结点 V_i 的新编号的二进制值，其它位全部为 0。

数字标签 S 和 L 经过这样改进后，RSCFC1 算法的 TEST 断言要相应地修改为：if$(S=S\&M_i)\neq L_i$ error。

其中标签 M_i 长度和 L_i 相同，其值的设置是把 V_i 在标签 L_i 中所对应的那一段全置 1，而在其它位全部置 0。

例如，在如图 2 中 M_1 = 00000000000000000000000011111111，这样在 TEST 断言中，变量 S 在与 M_i 进行"按位与"运算后，只把与 V_i 对应的那段信息保留下来，而其它段的信息全部变成了 0，再通过比较更新后的变量 S 是否与标签 L_i 相等来判断是否发生了控制流错误，如果相等则说明没有发生错误；反之，则说明发生了控制流错误。算法的 SET 断言保持和原算法一样。

通过这样改进后结点 V_i 的标签 $S_i(i=0,1,2,3)$ 如图 2 所示。

对于机器字长为 32 位的计算机，如果算法把标签 S(最高位除外)分成 4 段，理论上，改进后的算法在单层上能够加固程序的最大基本块数为：

$2^7+2^8+2^8+2^8-4$(4 个编码为全 0 的情况)$=892$，远大于 RSCFC 中的 31。

从以上的计算可以知道程序的最大后续结点个数 max 越大，相应地标签 S 的段数 MAX(MAX≥max)也会越多，算法所能够加固程序的最大基本块数就会越少。

对算法进行改进后，可以明显看出单层上加固程序的最大基本块数有了很大的提高，如果待加固程序的基本块总数比较大而使得改进后的算法无法在单层上对其进行加固，那么也可以采用对程序进行分层处理的方法，即通过分组基本

块来解决这个问题，基本块组内部各结点形成内层检测，而以各个基本块组为独立单位形成外层检测。与原算法的 TEST 断言进行比较可知，调整后的 TEST 断言并不增加检查指令的数量，所以改进后的算法在内存和时间开销上并没增加，在检错能力上也没有下降。

在前面的改进中，虽然算法可加固程序的最大基本块数有了很大的增加，但是并没有达到最好的效果，因为改进后数字标签 S 各个段的位数并没有考虑到待加固程序的特点，而是平均地分配位数，这就有可能存在有些段的位数不够用的情况。所以经过改进后的算法能加固程序的最大基本块数量并没有达到最大，进一步改进的目的就是要在前面改进的基础上合理分配标签 S 各段的二进制位数。

正如上面所分析，要使算法能加固程序的最大基本块数量达到最大，就要合理分配标签 S 各个段的位数，而 S 各个段的位数是由需要用该段进行编码的结点数量决定的。现假设机器字长为 W，程序所有结点中的最大位置信息值为 MAX，所以把标签 S(最高位除外)分成 MAX 段，而且对各个结点设置位置信息后具有以下性质：

用第 $i(i=1,2,\cdots,\text{MAX}-1)$ 段进行编码的结点数量为：NUM_i

标签 S 第 i 段的位数为：$[\log_2(NUM_i+1)]$

标签 S 各段的位数如图 3 所示。标签 S 的第 MAX 段的位数是由机器字长减去前面各段之和后，再减去最高位恒为 1 的那位后的结果，这样设置的目的是把可能没用到的标签 S 的位数都纳入到第 MAX 段中。每段的结点数量都加上 1 的目的是保证需要用各个段表示的每个结点在段内的二进制值不会为 0。其中"$[x]$"表示大于等于 x 的最小整数，如$[1.2]=2$。

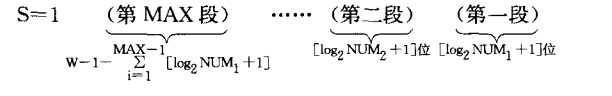

图 3　进一步改进后标签 S 各段位数

现在估计当机器字长为 32，MAX 为 4 时，用进一步改进后的算法能够加固程序的最大基本块数。由于算法对结点设置位置信息值时，总是设置为结点位置信息的所有可能取值中还未被与它拥有相同前趋结点的所有结点的位置信息值占用的最小值，因此假设对程序控制流图中的各个结点设置位置信息后还具有以下性质：

用第一段、第二段、第三段和第四段进行编码的结点数量分别占结点总数的 50%、25%、15%和 10%。

算法能够加固程序的最大基本块数记为 G，能够用第 i 段进行编码的结点数量记为 NUM_i，则有：

$NUM_1=0.50G$，$NUM_2=0.25G$，

$NUM_3=0.15G$，$NUM_4=0.10G$

由于是要计算改进后算法能加固程序的最大基本块数，因此标签 S 的各个段中的每一个编码(除了各段中为全 0 的编码)都会用到，所以有：

$$[\log_2(NUM_1+1)]+[\log_2(NUM_2+1)]+[\log_2(NUM_3+1)]+[\log_2(NUM_4+1)]=31 \quad (1)$$

式(1)近似为：

$$\log_2 NUM_1+\log_2 NUM_2+\log_2 NUM_3+\log_2 NUM_4\approx 31$$

所以，

$$NUM_1\times NUM_2\times NUM_3\times NUM_4\approx 2^{31} \quad (2)$$

把 NUM_1，NUM_2，NUM_3，NUM_4 的值代入式(2)得：

$0.5G\times 0.25G\times 0.15G\times 0.1G\approx 2^{31}$

得：$G^4\approx 2^{31}\times 533$

最后得出：$G\approx 1034$，远大于 31。

从以上的计算结果可以看出，改进后算法可单层加固程序的最大基本块数可以达到 1034 个。从计算过程可以看出，G 的值取决于机器字长、程序所有结点的最多后续结点数和程序内部的结构特征，所以取不同的假设前提得到的 G 也将不同，上面例子的前提假设是比较一般化的。

现在计算如果机器字长为 64 位，而其它假设与上面相同的情况下 G 的值。可得：

$G^4\approx 2^{63}\times 533$

得：$G\approx 2^{18}$

从计算结果可以知道，当机器字长为 64 位时，待加固程序的最大基本块数可以达到 2^{18}。

结束语　本文通过对 RSCFC 算法中标签 S 和 L 设置方式的改进及对 TEST 断言的调整，使得改进后的算法在不增加检查指令数量和降低检错能力的前提下，单层加固程序的最大基本块数比原算法提高了 40 倍之多，极大减少了 RSCFC 算法在检测比较大的程序时采用分层嵌套的方式处理所付出的内存开销和时间开销。特别是在设定前提下，如果机器字长为 64 位，程序的最大基本块数将超过 2^{18}。那么改进后的算法基本上可以在单层上实现对大多数程序的加固，而 RSCFC 算法在 64 位机器上通过单层对程序进行加固，程序的最大基本块数只有 63。

参 考 文 献

[1] Shivakumar P, Kistler M, Keckler S W, et al. Modeling the effect of technology trends on the soft error rate of combinational logic [C]// Bethesdaed. Proceedings of the 2002 International Conference on Dependable Systems and Networks (DSN 2002). Washington: D. C, 2002: 389-399

[2] Siewiorek D P, Swarz R S. The Theory and Practice of Reliable System Design[M]. Digital Press, 1982

[3] Iyer R K, Rossetti D J. A measurement—based model for workload dependence of CPU errors[J]. IEEE Transactions on Computer, 1986, 35(6): 511-519

[4] Bagchi S, Kalbarczyk Z, Iyer R, et al. Hierarchical error detection in a SIFT environment[D]. U of nlinois, 2001

[5] Lyons R E, Vanderkulk W. The Use of Triple-Modular Redundancy to Improve Computer Reliability[J]. IBM Journal of Research and Development, 1962, 6(2): 200-209

[6] Pradhan D K. Fault-Tolerant Computer System Design [M]. Prentice Hall, 1996

[7] Lu D J. Watchdog Processor and Structural Integrity Checking [J]. IEEE Transactions on Computers, 1982, 31(7): 681-685

[8] Mahmood A, McCluskey E J. Concurrent error detection using watchdog processors-a survey[J]. IEEE Transactions on Computers, 1988, 37(2): 160-174

[9] Aho A, Sethi R, Ullman J. Compilers: Principles, Techniques and Tools(2nd)[M]. Reading MA: Addison-Wesley, 1986

[10] Oh N, Shirvani P P, McCluskey E J. Control-Flow Checking by Software Signatures [J]. IEEE Transactions on Reliability, 2002, 51: 111-122

[11] Reis G A, Chang J, Vachharajani N, et al. SWIFT: Software implemented fault tolerance[OL]. http://liberty.princeton.edu/publications.cg03-swift.pdf

[12] Nicolescu B, Savaria Y, Velazco R. Software Detection Mechanisms Providing Full Coverage Against Single Bit-Flip Faults [J]. IEEE Transactions on Nuclear Science, 2004, 51(6): 3510-3518

[13] Li Ai-guo, Hong Bing-rong. Software implemented transient fault detection in space computer[J]. Aerospace Science and Technology, 2007, 11(2/3): 245-252

(上接第 350 页)

结束语 数据流的时变性决定了分析数据流中概念变化的重要性。目前概念变化的研究非常活跃，一些解决数据流中概念变化的方法和机制已经被提出。然而，总体说来，数据流中的概念变化处理还处于理论和方法的探讨阶段，还没有一种方法可以完全地检测到数据流中的所有概念的变化。为了简化概念变化的处理，通常采取的做法有：假设数据中只存在某种类型的概念变化；数据流的数据分布恒定；同时限定概念变化的速率和范围等。而真实的数据流中往往不符合以上 3 个条件，还需要进一步研究其概念变化本质。

目前数据流研究逐渐面向不确定数据流(Uncertain Data streams)[30]，如移动位置流(Location Streams)、文本数据流(Text Streams)、传感数据流(Sensor streams)，这些数据流具有不确定性(Uncertainty)，概念变化研究仍有着重要的应用价值，可以将概念变化与异常检测相关结合，以及时有效地发现数据流的异常现象。不确定数据流中的概念变化研究仍然有着重要的意义。

参 考 文 献

[1] 王涛，李舟军，颜跃进，等. 数据流挖掘分类技术综述[J]. 计算机研究与发展，2007，44(11)：1809-1815

[2] Tsymbal A. The problem of concept drift: definitions and related work. TCD-CS-2004-15 [R]. Dublin, Ireland: Department of Computer Science Trinity College, Trinity College, 2004

[3] Gama J. A survey on learning from data streams: current and future trends[J]. Progress in Artificial Intelligence, 2012, 1(1): 45-55

[4] 曾华军. 机器学习[M]. 张银奎，等译. 北京：机械工业出版社，2003

[5] 王永利. 基于概要的数据流挖掘若干研究[D]. 南京：东南大学，2006

[6] 张鹏. 挖掘概念漂移的数据流[D]. 北京：中国科学院，2009

[7] Widmer G, Kubat M. Effective learning in dynamic environments by explicit context tracking[C]//Proceedings of the Sixth European Conference on Machine Learning. Vienna, Austria, 1993: 69-101

[8] Schlimmer J, Granger R. Incremental learning from noisy data [J]. Machine Learning, 1986, 1(3): 317-354

[9] Li P-P, Wu X-D, Gao Y-J. A Random Decision Tree Ensemble for Mining Concept Drifts from Noisy Data Streams[J]. Applied Artificial Intelligence, 2010, 24(7): 680-710

[10] Kubat G, Gama J, Utgoff P. Special issue on incremental learning systems capable of dealing with concept drift[Z]. Amsterdam, Netherlands: Intelligent Data Analysis, 2004

[11] 尹志武，黄上腾. 一种自适应局部概念漂移的数据流分类算法[J]. 计算机科学，2008，35(2)：138-139

[12] 孙岳，毛国君，刘旭，等. 基于多分类器的数据流中的概念漂移挖掘[J]. 自动化学报，2008，34(1)：93-97

[13] 刘耀宗，张宏，王永利. 一种自适应概念变化的数据流分类器[J]. 计算机研究与发展，2007，44(Suppl)：63-68

[14] Klinkenberg R. Using labeled and unlabeled data to learning drifting concepts[C]//Proceedings of the Workshop notes of the 1 JCAI-01 Workshop on Learning from Temporal and Spatial Data. Menlo Park, CA, USA, 2001: 16-24

[15] Zhang P, Zhu X, Guo L. Mining data streams with labeled and unlabeled training examples[C]//Proceedings of the ninth IEEE International Conference on Data Mining. Miami, FL, USA, 2009: 627-636

[16] Widmer G, Kubat M. Effective learning in dynamic environments by explicit context tracking[C]//Proceedings of the Sixth European Conference on Machine Learning. Vienna, Austria, 1993: 69-101

[17] Kholghi M, Keyvanpour M R. Active learning framework combining semi-supervised approach for data stream mining[J]. Intelligent Computing and Information Science, 2011, 135: 238-243

[18] Hulten G, Spencer L, Domingos P. Mining time-changing data streams[C]//Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. San Francisco, California, USA, 2001: 97-106

[19] Peter V, Abranham B. Entropy-based concept drift detection[C]// Proceedings of the 6th International Conference on Data Mining. Hong Kong, China, 2006: 1113-1118

[20] 张育培，柴玉梅，王黎明. 基于鞅的数据流概念漂移检测方法[J]. 小型微型计算机系统，2013，34(8)：1787-1792

[21] Verpeck J T, Meehl G A, BonyS, et al. Climate data challenges in the 21st century[J]. Science, 2011, 331(6018): 700-702

[22] 罗秀，王大玲，冯时，等. 一种面向周期性概念漂移的数据流分类算法[J]. 计算机研究与发展，2009，46(Suppl)：400-405

[23] Li P P, Wu X D, Hu X G. Mining recurring concept drifts with limited labeled streaming data[C]//Proceedings of the 2th Asian Conference on Machine Learning. Tokyo, Japan, 2010: 241-252

[24] Yang Ying, Wu Xin-dong, Zhu Xing-quan. Combining proactive and reactive predictions for data streams[C]// Proceedings of the eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Chicago, Illinois, USA, 2005: 710-715

[25] Chen S, He H, Li K, et al. MuSeRA: multiple selectively recursive approach towards nonstationary imbalanced stream data mining[C]//Proceedings of the 2010 International Joint Conference on Neural Networks. Barcelona, Spain, 2010: 1-8

[26] Chen S, He H. SERA: selectively recursive approach towards nonstationary imbalanced stream data mining[C]//Proceedings of the 2009 International Joint Conference on Neural Networks. Atlanta, Georgian, USA, 2009: 522-529

[27] 辛铁，郭躬德，陈黎飞，等. IKnnM-DHecoc：一种决概念漂移问题的算法[J]. 计算机研究与发展，2011，48(4)：592-601

[28] 朱群，张玉红，胡学钢，等. 一种基于双层窗口的概念漂移数据流分类算法[J]. 自动化学报，2011，37(9)：1078-1083

[29] 杨春宇. 数据流上聚类与分类算法[D]. 北京：清华大学，2009

[30] 周傲英，金澈清，王国仁，等. 不确定性数据管理技术综述[J]. 计算机学报，2009，32(1)：1-16