现代计算机技术与应用
2022 年 5 月第 4 卷第 2 期

浅析计算机硬件技术研究

陈　瑜

河南科技大学，洛阳

摘　要 | 针对计算机系统中软、硬件可靠性问题的不同特点，讨论容错技术的最新发展现状，分析计算机系统中的各种容错方法，包括传统的冗余设计、错误回卷恢复机制以及当前研究较多的一般化容错设计方法等，研究目前已有的一些容错方法在反应延迟、容错成本、精确量化、异构同步、可靠性建模等方面存在的缺陷以及待解决关键问题，并对如何进一步更好地完善和使用这些容错方法进行总结。

关键词 | 计算机系统；容错；冗余；软件错误；硬件错误

1　概述

容错技术是指当系统在运行时有错误被激活的情况下仍能保证不间断提供服务的方法和技术。在一些特殊应用场合，如航空航天、国防军事、核能电力、工业化工、卫生医疗急救等关键部门，一次计算机系统错误的发生就可能导致不可挽回的巨大损失，在这些关键系统的设计中必须采用大量的容错技术来保证运行中突发的计算机错误不会导致整个系统的失效。

作者简介：陈瑜，河南科技大学，学生。
文章引用：陈瑜．浅析计算机硬件技术研究［J］．现代计算机技术与应用，2022，4（2）：57-68.
https://doi.org/10.35534/mcta.0402009c

计算机系统错误总体可分为硬件错误和软件错误。随着集成电路工艺的进步，处理器系统的瞬态故障率 SER 急剧增加且远远大于其他故障类型数：（1）晶体管工作电压的减小降低了集成电路噪声容限，使芯片更易受瞬态故障影响；（2）处理器主频的提高使其故障率增加；（3）集成度的提高使芯片中晶体管数量呈指数性增长，使整个芯片的故障率增加。文献预测，从 1992 年到 2011 年芯片瞬态故障率将增加 9 个数量级。另一方面软件在其整个生命周期都无法避免地存在设计错误，虽然研究者们已经针对软件设计错误提出了 N-version program 等软件多样化方法以求减小各版本软件在表决点处相关错误的发生概率，但是在目前软件成本某些时候已经大大超过硬件成本的前提下，此策略的可行性还存在很大问题。

综上所述，计算机系统的可靠性研究在提高整个系统可靠性过程中占有举足轻重的地位。本文旨在对目前已存在的各种针对计算机硬件、软件错误的容错方法及技术，恢复策略研究热点的研究现状进行比较全面的综合分析，以期为进一步拓展计算机容错研究范围，深化研究内容提供借鉴。

2　硬件错误的容错方法

计算机硬件错误可分为永久性错误、瞬态错误和间歇性错误。永久性错误一般由硬件老化、电路短路等原因产生，一旦发生则原定功能失效，必须通过替换元器件来完成恢复；间歇性错误处于永久性错误和瞬态错误 2 种情况之间，表现为瞬态错误的发生频率超过系统可靠性允许阈值范围；瞬态错误在目前的硬件环境及未来元器件继续高度集成的发展趋势下正以几何级数的速度增长，其错误数在整个系统错误总数中占最大的比重，对整个系统的可靠性影响度也是最大的。

冗余的系统不一定是容错的，但容错的系统一定是存在冗余的。冗余方法在容错策略中被广泛用来探测、诊断并恢复系统运行时发生的错误。按冗余资源的形式不同可以将冗余方法分为硬件冗余、信息冗余、时间冗余、线程冗余等。

2.1　硬件冗余

硬件冗余按照冗余程度不同可以分为部分冗余（例如 CPU 冗余）和完全系

统冗余。对于完全系统冗余（以双系统为例）按照工作方式不同可分为冷备、温备、热备以及双工 4 种工作方式。（1）双系统冷备：当工作机故障时，未加电的备份机启动并自动接替工作，对工作机进行故障诊断和维修，待故障修复完毕时，工作机去电变成备用机；（2）双系统温备：双机同时加电，一机工作，另一机处于等待。一旦工作机故障时，系统将进行自动切换，由温备机接替工作，并对故障机进行诊断和维修；待故障修复完毕时，工作机变成温备机；（3）双系统热备：双机同时加电，且均处于工作状态，只是热备机的处理结果不输出。一旦值班机出现故障，更改值班机，进行结果切换；待故障修复完毕时，工作机变成热备机；（4）双系统双工：双机同时加电和工作，处理结果同时输出，进行比较一致后输出。

对于故障导向安全性和实时性要求较高的系统，双系统双工是首选的工作方式。一般系统依据系统平均故障诊断时间和切换时间的要求可采用其他 3 种工作模式。此外，双系统中应该规定值班主设备，例如遇到双工结果都正确的前提下，就由值班机负责输出。

除上述双模冗余系统外，目前三模、四模冗余技术也得到了广泛应用。三模冗余 TMR 输入信号由完全相同的 3 个模块分别独立处理，每个模块生成一个运行结果交给决策器，由决策器进行判断并输出结果。但此类方法的缺点是只能发现单个模块错误且没有重构策略来修复错误模块，一个可以允许最大 Byzantine 错误数为 m 的系统的冗余度至少为 3m+1。

在一些对可靠性要求特别高的地方经常采用 NMR（N>3），其基本思路与上述 DMR、TMR 类似，将关键部件由功能上相同的 N 个模块分别独立完成，并经判断输出，由于多个执行单元接收指令以及运行时间上的差异，决策器同步所有的功能单元输出结果的周期变长，相应的延迟也会增大。

目前硬件冗余技术中基于动态可重构现场可编程门阵列 FPGAs 方法已经成为当前研究热点，根据抽象层次的不同可分为 2 层：硬件层 Device-Level（DL）和配置层 Configuration-Level（CL）。其中硬件层的目标是在包含错误资源的单元池中构建一个无错的门阵列，当发现错误单元后对逻辑门做永久性调整，选择冗余的单元来代替错误单元，因为这种改变直接在硬件层执行，所以 DL 方法

https://doi.org/10.35534/mcta.0402009c

一般用在生产制造中，且对于终端用户工具透明；而配置层 CL 方法则不考虑设备实际的物理结构，而将整个 FPGA 抽象成一个可用单元的资源集合，通常以图结构的方式出现，一旦电路确定后，从所有可用单元资源中选出无错的单元使用，由于每次可用单元资源的选择都要进行错误判定，因此 CL 方法可以解决之前未知的新类型单元错误问题，但同时需要付出额外的配置时间代价且 CL 方法对于终端用户工具可见。

硬件冗余方法作为被研究最多、成本最高、提升系统可靠性最有效的方法之一已经有很多研究成果，并已在很多现实场合投入使用。

2.2 信息冗余

所谓信息冗余即在原始数据中附加若干位的冗余信息以达到故障检测或故障恢复等目标的容错技术，包括检错编码与纠错编码 2 种。检错编码可以自动地发现错误，而纠错编码具有自动发现错误和纠正错误的能力。编码技术常用在信息的传输、存储和处理中，具有代表性的信息冗余编码方法不仅有常用的奇偶校验码 Parity、循环冗余校验码 CRC、海明码及它们的扩展改进版本，还有较近期的 RED-FECMechanism、ABFT、check-sumEDAC 等。

信息冗余技术相比其他容错策略有其独特的优势。代价小，仅需要少量额外的存储字节和计算开销或少量的额外编码电路即可，与大规模的系统硬件冗余相比开销要小得多；速度快：冗余信息一般同数据同在一个时间片处理，检错、纠错动作在处理数据时并行完成避免了错误恢复操作带来的延迟。

2.3 时间冗余

在非硬件冗余、非强实时系统中，可使用时间冗余技术来达到容错目的。时间冗余的工作方式有 2 种：（1）RSHW，即在同一硬件上对同一数据在不同时间片执行同一指令集；（2）使用数据延迟设备及表决电路结合，将一次数据处理的输出结果通过设置不同的延迟大小而复制成多个版本并在表决器处进行比较。

时间冗余容错技术的关键问题在于延迟时间 Tdelay 大小的选择，即如果延迟

时间选择过大则造成时间开销增大，从而失去了使用容错方法来避免发生过大延迟的意义；若延迟时间选择过小且小于单粒子翻转 Single Event Upsets（SEUs）的一个脉冲宽度则表决器的输入的大部分可能是错误信息，从而达不到容错的目的。

2.4 硬件线程冗余

硬件实现的线程级容错主要应用于如 SMT（Simultaneous Multi-threading）或 CMP（Chip Multi-processing）等多线程处理器在真正的线程级实现容错，方法的基本思想是在并行多线程平台上将主线程复制成多份并同时运行，最终比较运行结果来屏蔽 SEUs 错误。具有代表性的方法有冗余多线程 RMT（Redundant Multi-threading）用于 SMT 处理器实现容错，基于 CMP 的容错称为芯片级冗余多线程（chiplevel redundant threading，CRT），基于微线程的粗粒度超标量容错结构 MTB 等。这里以并行双线程冗余即 leader/follower 结构为例子来进行讨论。并行线程冗余容错方法最初的工作方式是主线程 leader 与副线程 follower 分别独立执行，在工作时间片内不进行通信，仅比较最终运行结果来实现容错，各个线程之间有很多关键资源存在冲突，如变量存储队列（LVQ）、中间结果存储队列（BOQ）以及存储缓冲区（StB）等，由于线程之间互相独立执行，关键资源的竞争将可能导致死锁或线程不同步而带来的巨大的延迟。

在最近的关于并行多线程冗余容错研究的文献中所体现的共同的主要思想是采用主副线程通信、共享中间结果队列的方式加快线程执行速度，同时完成容错。相比其他容错策略，线程冗余方法所需要的硬件成本较低，主要依靠的运行平台 SMT 或是 CMP 正在趋于市场化，且由于冗余单位小使得发现恢复错误及时、迅速，有很好的实时性。总体来说硬件实现的真正线程级容错是一种有前途的、可行的和高效的硬件容错解决方案。

3 软件错误的容错方法

分析一个计算机系统的可靠性，必须要考虑其软件的可靠性因素，但由于软件可靠性方面的研究比较困难，大大地落后于硬件方面的研究，目前还没有确定并且成熟的一套可供工程使用的方式和方法，所以在进行系统的可靠性预

https://doi.org/10.35534/mcta.0402009c

计时，往往忽视软件的失效率。人为的软件设计错误在软件整个生命周期中都一直存在，这一点已经得到证明。这种设计错误，在一定的输入激励下将产生一定的故障现象，客观上很难使用统一的模型对这种思维的结果进行数学的描述。例如，软件测试可以发现软件设计错误，通过修改软件的可靠性可以得到提高，但也有可能因为修改了已存在的错误带来了新的错误使得可靠性下降，所以软件可靠性在研究方面有很多地方不够成熟。这里主要介绍软件多样性方法，恢复块方法以及防卫式程序设计方法，除此之外提高软件容错能力亦可以从计算机平台环境、软件工程和构造异常处理模块等不同方面达到。此外，利用高级程序设计语言本身的容错能力，采取相应的策略，也是可行的办法，如 C++ 语言中的 try_except 处理法、try_finally 中止法等。

故障的恢复策略一般有 2 种：前向恢复和后向恢复。所谓前向恢复是指使当前的计算继续下去，把系统恢复成连贯的正确状态，弥补当前状态的不连贯情况，所谓后向恢复是指系统恢复到前一个正确状态，继续执行。N-version programming 方法属于前向恢复策略。

3.1 N-version programming 方法

N 版本软件容错方法的基本思想是各个版本软件由不同的团队独立设计，使用不同的方法，不同的设计语言，不同的开发环境和工具来实现。目的是减少各个版本软件在表决点上出现相关错误的概率。这里的各个版本的软件设计过程需要遵循这样几个原则：（1）总体设计相同，避免错误恢复的全局回滚；（2）多样化模块之间统一接口；（3）多样性封装，即模块内部多样性对外不可见；（4）保证各个版本软件设计独立性。每当有方法调用请求到达管理器处，管理模块将此次请求分别发送对应于各个版本软件的相应模块，由于完成同一功能的各个版本软件内部实现机制互相独立可能导致运行时间不一致，则当所有运行结果到达表决点完成错误判定和结果输出。

3.2 恢复块方法

恢复块方法采用后向恢复策略。它提供具有相同功能的主块和几个后备块，

主块首先投入运行，结束后进行验收测试，如果没有通过验收测试，系统经现场恢复后由一后备块运行。这一过程可以重复到耗尽所有的后备块，或者某个程序故障行为超出了预料，从而导致不可恢复的后果。设计时应保证实现主块和后备块之间的独立性，避免相关错误的产生，使主块和后备块之间的共性错误降到最低限度。验收测试程序完成故障检测功能，它本身的故障对恢复块方法而言是共性，因此，必须保证它的正确性。

3.3 防卫式程序设计方法

防卫式程序设计是一种不采用任何一种传统的容错技术就能实现软件容错的方法，对于程序中存在的错误和不一致性，防卫式程序设计的基本思想是通过在程序中包含错误检查代码和错误恢复代码，使得一旦错误发生，程序能撤消错误状态，恢复到一个已知的正确状态中去。其实现策略包括错误检测、破坏估计和错误恢复 3 个方面。

4 错误回卷恢复机制

前文介绍的各种针对硬件错误、软件错误的容错方法其目标均为在系统运行发生错误时，第一时间检测到错误信息、定位错误位置，并最大可能屏蔽错误信息输出，在不打断系统正常运行情况下或以最小延迟为代价保证系统依然正确输出，但无论是何种容错方法，他们的屏蔽错误能力都是有限的，这就需要另外的引入错误回卷恢复机制来保证系统在发生无法屏蔽的错误后在最小的延迟时间内回归到正常工作状态。

讨论较多的错误回卷恢复方法按照其基于的对象不同可以分为两大类：基于检查点的错误回卷恢复，以及基于日志的错误回卷恢复。所谓 PWD 假设是指，假设所有由进程产生的非决定性事件都可以由一些决定性信息完全演绎，且这些决定性信息可以被记录在日志中。

所谓全局一致检查点是指：在全局检查点中，若某进程的检查点文件显示其已接收到一条消息，则必有相应进程的检查点文件显示对该消息的发送，故障恢复必须从一个全局一致检查点开始。任何系统恢复策略的基本原则之一就

https://doi.org/10.35534/mcta.0402009c

是将系统由非一致性的状态恢复到错误发生前的一致性状态。检查点又分为协同检查点和非协同检查点两类。非协同检查点允许每个进程自己有最大的自由度来决定何时来设置检查点，这样做的好处是每个进程都能在最方便的时候设置检查点，从而减小全局阻塞设置检查点模式带来的开销；但另一方面这样做的缺点是：有可能引发多米诺效应，导致大量有用计算丢失而使计算回到最初阶段（这是最坏的情况）；非协同检查点模式可能会生成无用的检查点（即永远不属于全局一致性状态的一部分），并导致过期检查点资源回收机制出现异常。

基于日志的错误回卷恢复在判断错误发生后使用此错误之前最近的检查点和日志信息完全重新演绎指令执行过程。基于日志的错误回卷恢复方法特别适用于频繁与外界交互且外界操作不可撤销的应用场合（比如自动提款机已经支付出的钱或者打印机已经打印出的字符等均为不可逆操作），其原因在于信息在交付外界设备执行之前可以由日志记录的决定性信息将处理过程重复演绎并核查结果，保证了交付信息的一致性。

5　一般化容错方法

虽然不同的应用背景对于可靠性的要求不同（可靠性、成本、反应时间等诸多因素）使得各种容错策略的设计迥异，研究者们还是在试图寻找各种容错设计的最大共同点和更具有一般性、广泛性的方法，目标是实现可靠性设计的非定制 COTS 与可靠性的可裁剪，下面分别介绍两个从硬件和软件不同角度出发的以容错设计一般化为目标的例子。

文献中给出了一种容错软件设计的一般化结构，称作 Chameleon，根据可靠性需求可进行不同选择来动态地达到可靠性可裁剪的目的。整个软件体系都由对象 ARMORs 来组成，根据功能不同可以将 ARMORs 分为三类：（1）管理模块 Managers：负责管理其他 ARMORs 对象根据用户可靠性要求完成容错策略的选择和组织；（2）后台通信模块 Daemons：允许 Chameleon 访问网络中其他节点，提供 ARMOR 的错误探测并给出 ARMOR 之间通信的渠道；（3）普通 ARMORs：根据应用的可靠性需求提供具体实施方法，比如指令重执、表决、检查点、心跳检测等具体解决方法。

整个 Chameleon 软件结构类似于操作系统的结构组成，内核由执行硬件平台构成，外围是商业非定制的一般化操作系统（例如 Linux 等），操作系统之上依次分别为可重构的 ARMORs 框架以及具体 ARMORs 对象和管理执行层，最后是具体的应用层。

软件 COTS 原则，给出了一种容错计算机系统设计的一般化体系结构，在最大程度保证容错策略普适性的同时最小化具体应用的功能特点。根据具体应用的可靠性不同可以对各个可配置层进行不同的配置方案，比如冗余信道、冗余链路的冗余度大小、输入输出数据整合层的数据核查方案的选择等。通过配置的改变，系统可以达到可靠性由低到高很大范围内的自由裁剪。一般化硬件结构 GUARDS 基于所有组件（包括硬件和软件）COTS 原则，给出了一种容错计算机系统设计的一般化体系结构，在最大程度保证容错策略普适性的同时最小化具体应用的功能特点。根据具体应用的可靠性不同可以对各个可配置层进行不同的配置方案，比如冗余信道、冗余链路的冗余度大小、输入输出数据整合层的数据核查方案的选择等。通过配置的改变，系统可以达到可靠性由低到高很大范围内的自由裁剪。

6　有待进一步解决的问题

尽管目前计算机系统容错研究已经取得了一些成果但仍然存在一些薄弱的研究点以及一些关键问题没有得到很好的解决，具体可以概括为以下几点。

（1）目前的硬件冗余容错方法研究，虽然可以根据系统结构从总体上使用概率统计方法分析系统的可靠性，但还缺乏广泛适用的有效方法实现对复杂冗余系统的有效建模、管理和精确量化分析，特别是在各部件可靠度不一致的情况下，建模分析的理论问题尚未解决。另外，硬件冗余实现成本大、功耗大、占用空间大等依然是需要解决的大难题；

（2）对于目前的信息冗余容错方法来说，绝大部分 ECC 算法对于处理连续位出错的能力非常有限，而占系统总错误比例非常大的瞬态错误所导致的内存错误通常就是多个连续位出错的情况。这方面还需要继续研究发展；

（3）对于时间冗余容错方法，除了由时间换空间而带来的较大的延迟外，

https://doi.org/10.35534/mcta.0402009c

此类方法还存在对永久性错误不敏感的问题；

（4）对于并行多线程冗余容错方法，冗余线程之间的同步、通信等问题，线程之间的系统资源合理分配避免死锁问题等目前都没有得到很好的解决，而且对于在单线程的处理器上引入真正的线程级机制实现容错人们研究较少；

（5）在软件冗余方面，对于规模较小、数据处理过程较简单的小型软件很难体现软件多样性的优势，而另一方面，如果程序规模过大、过于复杂，一般来说 N 版本软件的费用也会不菲。目前软件冗余技术的研究尤其滞后；

（6）对于恢复块方法来说，用来判断程序是否安全运行的测试模块的无错运行假设是整个系统设计的前提，这个假设本身就存在一定的问题；

（7）防卫式程序设计方法缺乏系统的理论依据，主要依靠程序员自身的经验是无法进行系统可靠性分析的。

总体来说当前硬件错误的容错研究相对比较全面和深入，不论是理论研究还是实际应用都有了很多成果，但是软件方面不论是应用软件还是操作系统方面的容错研究都明显不足,事实证明软件可靠性是一个非常重要的问题，也是一个具有巨大探索价值的研究领域。针对以上所述本文提出了下一步的工作重点。

（1）ISO9126 将软件的可靠性定义为“软件在特定的条件下，在特定的时间内，与维持其性能水平的能力有关的一组属性，这组属性包括控制流结构、数据流结构、数据结构、内部重用度、耦合形式、内聚等影响软件可靠性的诸多因素。当前所要做的是不仅可对每个属性单独进行分析，还应给出一个综合度量作为诸如可理解性、正确性、可维护性、可靠性、可测试性和实现容易度之类的各种概念的一个指示器；

（2）针对软件可靠性的建模研究自从 J-M 模型以来，已经有了数百种可供参考的模型，有基于故障间隔的 S-W 模型、L-V 模型，以及基于缺陷计数的 Shooman 模型、Musa 模型、G-O 模型、M-O 模型等，但是至今仍没有一个普遍适用于所有潜在用户的有效方法，各种模型所表现出的可靠性度量准确性差异很大，没有哪几个模型可以被认为在所有情况下都是可靠的。更困难的是到目前为止还没有什么办法先验地判断出哪些数据集适合于一个具体模型。广泛开

展软件可靠性模型的分析、评价工作并找出失效数据集合的特征与可靠性模型之间的关联是解决此难题的关键点；

（3）使用冗余软件模块技术作为软件残留故障的保护是从硬件中得到继承的，几乎所有的软件故障都和设计实现相关联，而且也可以被复制，复制的故障将导致所有版本的仿真失效，这就是陷阱的影响。尽管已经提出的 N-version programming 方法在一定程度上解决了这个问题，但是如果软件失效是一致的、关联的或相似的，问题就变得更复杂了，解决问题的关键是构建专门针对冗余软件的可靠性测试模型，并可以对其可靠性和失效关联度进行定量分析；

（4）同构软件冗余与异构软件冗余的同步等关键技术；

（5）作为处理机、存储器、设备、文件和作业等诸多计算机系统关键资源的管理者，操作系统的安全性是整个计算机系统软件安全的必要条件，但是迄今为止，国内外在安全操作系统的实际应用并不成功。如何构建可信操作系统的体系结构，解决可信操作系统的自身完整性、环境适应性、主体客体行为可信问题以及冗余操作系统的关键实现技术也是下一步研究的重点。

7 结束语

随着计算机系统规模的不断扩大，对系统可靠性要求的不断提高，容错技术将成为保证计算机系统稳定运行的关键技术。理想的容错方案应该具有高可靠性，错误检测、错误恢复的低延迟性、性能的低损耗性以及成本低廉性的特点。针对具体应用的不同可靠性要求来设计构建最大程度贴近理想状态的容错方案远比看上去要困难。本文总结分析了近年来计算机系统在容错方面的技术发展情况、存在的问题和未来仍需要解决的一些重要问题，为进一步研究构建更加可靠的计算机系统提供了有益的借鉴。

参考文献

[1] 聂林波，刘孟仁．软件缺陷分类的研究 [J]．计算机应用研究，2004，21（6）：84-86.

[2] 陆阳，张本宏，魏臻，等．“二乘二取二”和“双模冗余—比较”结构对

https://doi.org/10.35534/mcta.0402009c

比研究［J］. 电子测量与仪器学报，2009，23（3）：15–22.
［3］张本宏，陆阳，韩江洪，等. “二乘二取二”冗余系统的可靠性和安全性分析［J］. 系统仿真学报，2009，21（1）：256–261.
［4］朱明程，温粤. FPGA 动态可重构数字电路容错系统的研究［J］. 东南大学学报（自然科学版），2000，30（4）：138–142.

Analysis of Computer Hardware Technology Research

Chen Yu

Henan University of Science and Technology, Luoyang

Abstract: In view of the different characteristics of software and hardware reliability problems in computer systems, the latest development status of fault tolerance technology is discussed, and various fault tolerance methods in computer systems are analyzed, including traditional redundancy design, error rollback recovery mechanism and general fault tolerance design methods which are studied more at present. This paper studies the defects of some existing fault tolerance methods in response delay, fault tolerance cost, accurate quantization, heterogeneous synchronization, reliability modeling and other aspects as well as the key problems to be solved, and summarizes how to further improve and use these fault tolerance methods.

Key words: Computer system; Fault tolerance; Redundancy; Software error; Hardware error