面向通信协议故障的分布式软件可靠性测试技术研究

房友园 齐 璇 战 茅
(北京系统工程研究所 北京 100101)

摘 要 分析了分布式软件系统中的典型通信协议故障,研究了基于API Hook的分布式软件可靠性测试方法,提出了基于通信协议故障注入的应用软件可靠性测试方法,并设计、实现了工具原型,为基于故障注入的分布式软件可靠性测试提供了技术手段。

关键词 通信协议故障,可靠性测试,故障注入
中图法分类号 TP311.5 **文献标识码** A

Research on Communication Protocol Fault-oriented Reliability Testing of Distributed Software

FANG You-yuan QI Xuan ZHAN Mao
(Beijing Institute of System Engineering, Beijing 100101, China)

Abstract We analyzed the classic communication protocol faults, researched on the reliability testing methodology of large distributed software, and designed a reliability testing tools based on communication protocol fault injection.

Keywords Communication protocol fault, Reliability testing, Fault injection

1 引言

故障注入(Fault Injection)技术作为一种非传统的软件测试技术,是指按照特定的故障模型,用人为的、有意识的方式产生故障,并施加特定故障于待测系统中,以加速该系统故障和失效的发生。故障注入技术最早应用于20世纪70年代的硬件测试,后来根据不同的测试对象发展出基于硬件的技术、基于软件的技术、基于模拟的技术和离子辐射的技术。软件故障注入测试(Software Fault Injection Testing, SFIT)是指采用软件的方法对软件系统进行测试的技术,始于1978年De Millo提出的程序变异测试。SFIT可以提高软件质量、评估软件等级,在一些高可靠性软件及安全关键软件等领域,如航空、航天软件的测试等应用广泛。

软件故障注入的测试方法适用于大型分布式软件可靠性的测试。基于协议的消息是分布式软件通信的基础,大型分布式软件实现了各个独立软件系统的协同工作,基于网络的进程间通信在分布式软件系统中处于核心地位。其本质是基于不同层次通信协议的消息传递机制,在不同的系统、操作环境、程序语言间进行信息交换,进而实现各应用程序之间的相互协作。由于各种基于标准的通讯消息具有良好的可读性和易分析性,因此对于分布式软件来说,通过分析协议交互内容来完成分布式软件交互测试是可行的测试方法,其有助于分析和发现影响系统可靠性的内部和外部原因,并对协议设计和实现的改善、软件可靠性的提高和脆弱性的降低有着重要意义。因此,本文重点针对基于协议消息实施故障注入测试,验证分布式软件的可靠性。

2 通信协议故障

在大型分布式软件系统中,网络管理系统、域名服务器、目录管理服务等应用的可靠性直接关系到分布式软件系统的质量,因此可以通过注入构造的HTTP、SNMP、LDAP、DNS等网络协议故障数据,测试分布式软件应用。其特定故障类型包括基于溢出的故障、基于URL异常的故障、基于bit异常的故障、基于BER编码的故障、基于格式化字符串异常的故障、基于整型数值异常的故障、基于符号缺失的故障、基于UTF-8编码异常的故障、基于二进制标记模式的故障、基于SQL命令的故障和基于编码异常的故障。

表1 通信故障模型

故障类型	故障描述
故障描述	通过捕获复制请求/响应/错误消息,测试分布式软件的反应
消息丢失	截获请求/响应/错误消息或将消息地址改为一个错误的链接地址,使其无法到达服务器端
消息内容错误	捕获修改请求/响应/错误消息内容,包括对数据类型、数据格式等内容的修改,再向调用方发送,测试分布式软件的反应
消息延迟	在请求/响应/错误消息中注入一个延迟,使其小于消息超时允许的最大时间
	在请求/响应/错误消息中注入一个延迟,使其大于消息超时允许的最大时间
消息乱序	应答消息先于请求消息到达

同时,由于大型分布式软件大多采用消息作为基本的数据通信方式,因此通信故障主要集中在消息的传输上,包括消息的复制、丢失、延迟、乱序、附加消息等,进而出现非法数据、

到稿日期:2011-08-01 返修日期:2011-11-27 本文受国家863项目(2008AA01A204)资助。
房友园(1981—),男,硕士,助理研究员,主要研究方向为软件工程、软件测试等,E-mail:fyy_bise@163.com;齐 璇(1970—),女,博士,副研究员,主要研究方向为软件工程、软件测试等;战 茅(1971—),女,硕士,副研究员,主要研究方向为软件工程、软件测试等。

乱序数据、上下文不一致数据、非法分隔、不一致数据等故障数据。测试该类故障时，可以采用故障注入技术对消息进行译码，并将有意义的故障注入消息中。其主要内容见表1。

3 基于 API Hook 的分布式软件可靠性测试方法

基于 API Hook 的故障注入方法能自动在消息中执行通信消息译码和状态扰动，通过配置消息发送方操作系统的协议堆栈来注入故障。消息中的信息以 API 脚本的形式呈现，使译码更易于实现。该方法使用一个包含两段 Hook 代码的消息处理 API。一个 Hook 截断输入的请求消息，通过一个消息连接将消息传送到故障注入器并接收来自故障注入器的经过修改的消息。然后这条消息被正常传送到目的地。另一个 Hook 截断应答消息，并进行类似的处理，最后输出。

故障注入器作为一个单独的过程来处理。这样构造的原因是：(1)简化注入器的设计；(2)便于把这个处理过程安排在一个单独的机器上运行，再通过 TCP/IP socket 等通信协议与消息处理 API 装置连接；(3)这使得多个节点可以使用相同的测试用例，实现测试自动化。

基于 API Hook 的故障注入方法使用一个解析模块和一个两阶段的解析过程来处理消息。第一阶段确定消息是否必须注入故障；第二阶段仅对那些在第一阶段表明需在第二阶段注入故障的消息进行有意义的参数扰动，并提供一个产生并执行测试用例的框架。利用已知通信协议故障模型和扩展的故障模型可以得到具体的测试脚本，从而生成测试用例。同时，为确保测试过程实时可见，方法允许交互消息和参数的实时可见，最终形成一个帮助确定分布式系统中哪些方法是测试关注的重点概要。

基于 API Hook 技术的故障注入方法拦截分布式软件服务应用在发送和接收消息时对操作系统网络套接字服务 API 的调用，转而执行测试程序的相应函数。测试程序识别出传输消息所使用的具体承载协议，提取消息的内容，并测试整个分布式软件服务调用过程。该方法具有客观化、自动化和轻量化的特点。

整个方法的总体结构如图1所示。

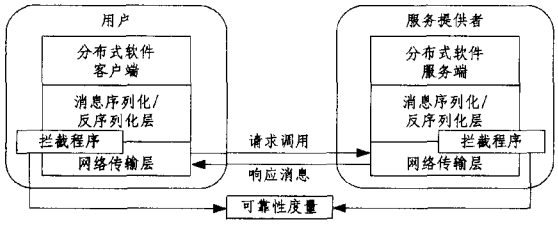

图1 基于 API HOOK 的可靠性测试方法

拦截程序所处的位置位于下面的网络传输层和上面的消息序列化/反序列化层之间，这样既可以很方便地截取消息，又可以获知与具体的网络环境相关的一些信息。整个测试过程完全自动进行，用户和服务提供者本身并不直接参与其中，这就保证了测试结果的客观性。

4 设计与实现

通过研究分布式软件可靠性测试技术，设计、实现了基于通信协议故障注入的可靠性测试工具。针对不同的分布式应用，通过产生各类故障消息单元，观察分布式应用对各种故障消息单元的响应行为，分析和发现影响分布式系统可靠性的内部和外部原因，进而改善分布式系统设计和实现缺陷，提高分布式软件的可靠性，降低系统脆弱性，为分布式软件的可靠性这一关键质量特性的测评提供有效技术手段。

4.1 体系结构设计

故障注入可靠性测试工具基于故障注入的分布式软件可靠性测试技术，通过对典型通信协议注入 Fuzz 测试数据植入故障，加速系统失效，观察系统在被注入故障情况下的行为，分析系统对故障的反应，实现大型分布式软件故障处理能力的测试。故障注入可靠性测试工具主要由控制器模块、Fuzz 测试数据模块、故障注入模块和数据收集与分析模块组成。故障注入可靠性测试工具的体系结构如图2所示。

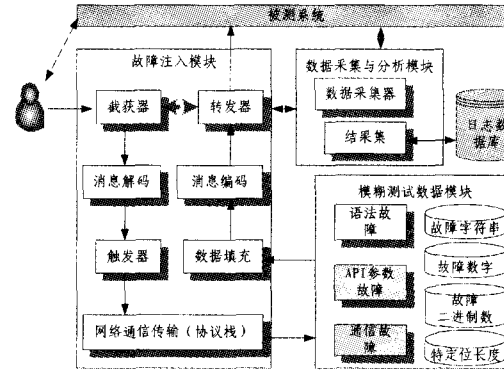

图2 软件体系结构图

其中，用户通过模拟测试数据模块针对特定数据包中的特定字段生成测试数据，并将其提交给控制模块，从而调度故障注入模块，基于模拟测试数据通过修改网络层消息中的参数来进行状态值扰动。测试中，数据收集与分析模块统计应答消息，对照服务故障模型，给出可靠性测试结果。

在设计基于通信协议故障注入的可靠性测试工具时，遵循以下设计原则：

- 选择面向对象的软件开发方法，以减少软件错误；
- 严格按照接口格式对有关数据实施检查，使输出数据尽量符合要求；
- 对从人机界面输入的有关操作实施严格的检查，增加边界值和无效值的判断，以防止输入错误，引起系统异常；
- 对所有可能引起系统异常行为(如溢出等)的软件模块，设置异常处理程序进行处理。

4.2 管理配置模块设计

管理配置模块是整个系统的管理程序，用于控制整个故障注入试验的软件。它提供用户与故障注入系统的接口，使用户可以选择故障数据。管理配置类还控制故障注入器将故障数据注入到目标系统中。另外，对无故障运行和带故障运行的目标系统的数据采集和分析工作，也是在管理配置类的控制下完成的。

管理配置类通过 Run 方法控制整个故障注入的生命周期。基于用户定义，对指定位置数据包注入指定长度和类型的故障，然后调用 reconnect 方法连接远程服务器。如果是 UDP，则直接连接；如果是 TCP，则使用 best-effort 服务尽力发送数据，并通过 Log 类记录日志；然后利用 send 方法，将 Fuzz 数据生成请求数据包，并通过选定的通信协议将数据包

发送到指定端口，从而最终完成基于 Fuzz 数据的故障注入。

4.3 故障注入模块设计

故障注入模块作为整个系统的核心部分，通过分析协议交互内容来完成分布式软件的交互测试。其主要功能是向目标系统中注入故障，要求其能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，以确保故障在正确的时刻注入到正确的位置。

故障注入模块基于协议扁平化思想，将协议解析为字符长串，自底向上从不同层次支持包括 TCP/IP、HTTP、SOAP 及私有协议在内的各类通信协议，通过网络客户与服务端的通信监听，实现客户端通信报文的动态捕获。故障注入类解析通信过程中所捕获的消息数据包，获取消息格式，结合协议规范中所描述的消息单元语法，构造完整的消息单元。对于所构造的消息单元，依据语法和语义信息选取特定字段作为故障注入点，结合 Fuzz 测试数据生成的故障数据消息，并通过绑定服务端通信进程，利用故障消息模拟客户与服务器通信。

同时，故障注入模块还负责 Fuzz 测试数据的生成。它根据目标程序中可能存在的特定故障而产生数据，这些故障数据很有可能引发软件故障。故障注入模块使用 Fuzz 测试验证应用的可靠性，其基本思想是故意将随机的坏数据插入应用程序，然后等待并观察哪里遭到了破坏。Fuzz 测试的技巧在于，它是不符合逻辑的：不去猜测哪个数据会导致破坏，而是将尽可能多的杂乱数据投入程序中。

Fuzz 测试的输入数据主要强调的是数据的随机性，包括消息传输过程的通信故障：延时、丢包、乱序；接口调用过程的数据包语法故障和 API 参数故障，可概括为：在命令行模式下随机输入的随机 ASCII 字符流（如非法数据、乱序数据、上下文不一致数据、非法分隔符、不一致数据、空数据等），以及在视窗模式下随机输入的有效键盘和鼠标输入序列。这些输入完全不考虑系统逻辑，从某种程度上说是一种破坏性的测试。具体实现界面如图 3 所示。

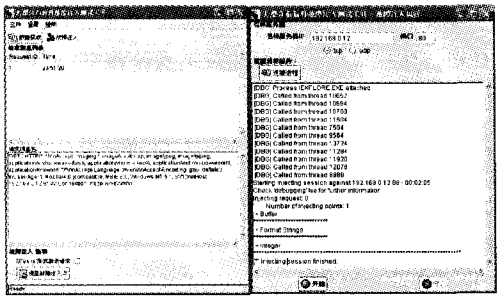

图 3 故障注入模块解界面图

4.4 XML 数据处理模块设计

XML 数据处理类主要针对 XML 文件的处理，提供了诸如读写 XML 元素、生成 XML 文档对象模式（DOM）、处理 XSL 文件等功能。

4.5 数据收集与分析模块设计

数据收集与分析包负责跟踪工作的任务，并在适当的时刻采集数据，同时对采集到的数据进行分析和处理。数据分析工作采用离线分析模式，通过分析计算可以判定目标系统中是否存在特定故障，为进一步进行故障诊断、隔离和获取提供数据基础。数据收集与分析包还通过日志记录器将工具运行期间的信息保存至本地硬盘。同时，日志记录器还保存所有接收和发送的数据包。

结束语 可靠性是衡量大型分布式软件质量的关键属性。但长期以来，在分布式软件可靠性测评领域缺乏易操作、高效的测试方法。本文分析了分布式软件系统中的典型通信协议故障，研究了基于 API Hook 的分布式软件可靠性测试方法，其包括分布式软件故障模型构建、故障数据构造等；提出了基于通信协议故障注入的应用软件可靠性测试方法，并实现工具原型，为基于故障注入的分布式软件可靠性测试提供了技术手段。

参 考 文 献

[1] Andrews J H, Brand L C, Labiche Y. Is mutation an appropriate tool for testing experiments[C]// Proceedings of the 27th International Conference on Software Engineering (ICSE'2005). St. Louis, MO, USA, 2005: 402-411

[2] Offutt A J, Untch R. Uniting the orthogonal[C]// Proceedings of the Mutation 2000: Mutation Testing in the Twentieth and the Twenty First Centuries. San Jose, CA, USA, 2000: 45-55

[3] Delamaro M E, Maidonado J C, Mathur A P. Interface mutation: An approach for integration testing[J]. IEEE Transactions on Software Engineering, 2001, 27(3): 228-247

[4] Ma Y-S, Kwon Y-R, Offutt J. Inter-class mutation operators for Java[C]// Proceedings of the 13th International Symposium on Software Reliability Engineering (ISSRE'2002). Annapolis, MA, USA, 2002: 352-363

[5] Lee H-J, Ma Y-S, Kwon Y-R. Empirical evaluation of orthogonally of class mutation operators[C]// Proceedings of the 11th Asia-Pacific Software Engineering Conference. Busan, Korea, 2004: 512-518

[6] Aichernig B K. Mutation testing in the refinement calculus[J]. Formal Aspects of Computing, 2003, 15(2/3): 280-295

[7] Jiang Y, Hou S-S, Shan J-H, et al. Contract-based mutation for testing components[C]// Proceedings of the 21st International Conference on Software Maintenance (ICSM 2005). Budapest, Hungary, 2005: 483-492

(上接第 103 页)

[5] 马知恩. 传染病动力学的建模和研究[M]. 北京：科学出版社，2004

[6] Heffernan J M, Smith R J, Wahl L M. Perspectives on the basic reproductive ratio [J]. Journal of the Royal Society Interface, 2005, 2(4): 281-293

[7] Anderson R M, May R M. Infectious Diseases of Humans [M]. Oxford: Oxford University Press, 1991

[8] Dickmann O, Heesterbeck J A P. Mathematical epidemiology of infectious diseases [M]. New York: John Wiley & Sons, 2000

[9] Van den Driessche P, Watmough J. Reproduction numbers and sub-threshold endemic equilibria for compartmental models of disease transmission [J]. Math Biosci, 2002, 180: 29-48

[10] Baroyan O V, Rvachev L A, Basilevsky U V, et al. Computer modeling of influenza epidemics for the whole country (USSR) [J]. Advances in Applied Probability, 1971, 3: 224-226