基于网络的虚拟调试环境的研究

王位杰,王 铮
(重庆大学 计算机学院,重庆 400044)

摘 要:基于网络的软件通常在现场部署前不能利用现场环境进行充分的调试,并且现场调试具有环境恶劣、周期长、风险大等特点,针对上述情况提出了基于网络的虚拟调试环境。该环境虚拟现场设备,提供各种与现场设备一致的调试数据,用于在现场部署调试之前对软件做充分的调试,验证软件的正确性,提高软件的可靠性,以期缩短现场调试和现场开发时间,降低软件开发和企业风险。文中对基于网络的虚拟调试环境作了阐述和分析,并对框架结构和实现关键技术进行了探讨。

关键词:现场设备;现场调试;虚拟调试环境;虚拟数据源;被调试程序;场景
中图分类号:TP31 **文献标识码**:A **文章编号**:1673-629X(2011)07-0046-04

Research of Network-Based Virtual Debugging Environment

WANG Wei-jie, WANG Zheng
(Institute of Computer, Chongqing University, Chongqing 400044, China)

Abstract: The network-based software can not be fully debugged in the site environment before the on-site deployment. The on-the-spot debugging has the characteristics such as poor environment, long period, risks, etc. For the above, the virtual debugging environment is proposed. By simulating on-site devices, this environment provides a variety of debugging data consistent with on-site devices to fully debug software before the on-site deployment for verifying the correctness of the software and improving software reliability so as to reduce on-site debugging and on-site programming time and reduce the risk of both software development and enterprise. In this paper, the network-based virtual debugging environment is elaborated and analyzed, and the frame structure and the key technologies are discussed.

Key words: on-site devices; on-site debugging; virtual debugging environment; virtual data sources; the debugged program; scenarios

0 引 言

随着信息化的推进,复杂的系统软件在各领域的应用方兴未艾。而软件调试在软件开发过程中占据越来越重要的位置,并逐渐成为软件开发的一个制约瓶颈。在一些特殊应用领域软件开发中,对软件(下文中统称为被调试程序)进行调试需要来自网络中的现场设备输出的数据,而在实际软件开发过程中,这些现场设备处在遥远甚至恶劣的环境中,开发人员既不可能通过远程网络将这些数据传输给被调试的程序或软件,又不可能亲自面临这些设备进行程序调试。如果在开发阶段不进行充分的调试而到现场进行调试必然会产生诸多预想不到的bug,即使在现场调试程序也是一件非常困难的事情。在现场进行调试时若出现bug就需要重新对程序进行修改,然后再在现场进行调试,这样反复的工作必然会造成软件开发延期;并且现场调试具有破坏生产设备甚至引发生命安全的风险。

如果搭建一个虚拟调试环境(Virtual Debugging Environment,VDE),构造虚拟数据源(Virtual Data Source,VDC)模拟现场设备的功能,通过网络为被调试软件提供由虚拟数据源产生的数据,便可以最大程度地解决上述遇到的问题[1~3],提高整个软件开发的灵活性与效率;同时在虚拟调试环境下进行虚拟调试,风险系数可以大大降低,亦不会造成任何生命财产损失。因此开发这样一个虚拟调试环境势在必行,文中就虚拟调试环境在windows平台下的设计需求与实现关键技术进行相关的探讨。

1 虚拟调试环境的需求分析

如上文所述,虚拟调试环境的应用场景是被调试程序对数据的需求来自网络中现场设备。虚拟调试环境由模拟现场设备功能内部的虚拟数据源组成。现场

收稿日期:2010-12-08;**修回日期**:2011-03-17
基金项目:国家自然科学基金项目(60773082)
作者简介:王位杰(1984-),男,硕士研究生,研究方向为分布式计算、软件测试、嵌入式系统;王 铮,副教授,硕士生导师,研究方向为嵌入式系统、分布式系统。

设备的数量决定了虚拟数据源的个数;根据具体的现场情况和计算机的处理能力,虚拟调试环境被部署于不同的计算机上,虚拟调试环境内部通过网络连接进行通信。每个虚拟数据源模拟真实环境中的一个现场设备的功能[4,5]。虚拟数据源产生的数据通过网络传送给被调试程序。

设计和开发虚拟调试环境应当考虑以下几个方面:

(1) 虚拟数据源。由于是非现场环境下对程序进行调试,虚拟数据源应当最大逼真度地模拟现场设备的功能以产生逼真的数据供调试用,每个虚拟数据源模拟一个特定的现场设备。

(2)网络通信的方式有多种情况,现在较常用的有无线网络传感通信、现场总线技术,串口通信、TCP/IP通信等。被调试程序可能会用到一种或多种通信方式,因此虚拟调试环境要满足被调试软件的这种需求,应当具有多种通信方式。虚拟调试环境内部各个虚拟数据源之间的联系也存在各种通信方式。

(3) 通过虚拟调试环境对程序进行调试,被调试程序在调试未结束之前或新的版本未发布之前必须不能做任何的修改,这样才不会影响其在真实环境下的正常运行。这一原则要求双方通信的协议应当与被调试程序使用的协议一致,这样双方才可能进行有效的通信。

(4) 当存在虚拟调试环境中的多个虚拟数据源部署于不同的计算机上时,对虚拟调试环境的管理配置就变得很繁琐,特别是调试活动频繁,调试环境复杂多样时。因此必须整体考虑对虚拟调试环境的配置。

通过上面的分析,开发这样一个虚拟调试环境需要解决以下关键问题:

(1) 虚拟数据源的建立与实现;

(2) 虚拟数据源与被调试程序之间的数据流模型;

(3) 虚拟调试环境与被调试程序之间的网络通信;

(4) 虚拟调试环境的配置与管理 。

以下内容将对这几个问题进行探讨。

2 虚拟调试环境的技术研究与实现

2.1 虚拟数据源的建立与实现

虚拟调试环境由至少一个虚拟数据源组成,数据源的个数由现场环境的设备个数和被调试程序对现场设备的需求共同决定。一个虚拟数据源模拟现场环境一个特定的设备。每个虚拟数据源均是一个自治单元,内部实现一个指定现场设备的需求功能,根据被调试程序需求生成并输出相应的数据然后通过网络传送给被调试程序。

虚拟数据源采用面向对象技术实现。近些年来的软件开发过程中,面向对象技术得到了广泛的应用,它采用数据抽象和信息隐藏技术,将数据和对数据的处理封装在一起,作为一个相互依存、不可分割的整体来处理。面向对象技术将对象及对象的操作抽象为一个新的数据类型:类,并且考虑不同对象之间的联系和对象类的重用性[6]。虚拟数据源对外以对象的方法呈现,内部的功能函数来模拟现场设备的功能。下面是虚拟数据源的定义:

```
Class  CVde{
Private:
…
//虚拟数据源内部变量
….
Public:
CVde();
CVde( parameter,…);
//虚拟数据源内部函数;
Return_value Function_Name( parameter,…);
…
};
```

根据虚拟数据源产生的数据的逼真度可将虚拟数据源类里的函数分为两类,一类是模型化的功能函数,它们通过专业领域的知识建立模型实现,根据输入信息(初始化条件)输出高度逼真的数据;另一类是对数据逼真度不高的函数,这类函数随着时间和空间的变化产生具有一定趋势的数据。使用构件技术通过提供丰富的数据源功能函数库可以完全满足这些设备的功能需求,构件技术对外提供统一的接口,内部的功能扩充不会影响外部的结构变化,再使用虚拟系统集成技术可以构造不同条件下的虚拟数据源[7,8];构件技术的使用不需改变整个虚拟调试环境结构并使其具备了可扩展性,进而满足不断增加和变化的调试需要。

虚拟数据源的初始化信息来源于被调试程序的需求,以相应的配置文件为载体传递给虚拟数据源,虚拟数据源读取配置文件的配置信息后传输给相应的模拟功能函数由其产生被调试程序需求的数据。windows平台下的.ini文件是windows的系统配置文件,统管windows的各项配置,它可以用来存放应用软件信息、注册表信息等,windows平台提供对.ini配置文件读取和写入的API函数。在windows平台下,文中虚拟数据源首先使用.ini文件来存储和配置初始化信息的配置文件,另一方面也可以从人机界面来传达被调试程序对虚拟数据源的需求,良好的人机界面可以使开发人员与虚拟调试环境进行有效的交互,这方面在下文进行讨论,这里对.ini配置文件使用作说明。

通过 windows 系统提供的.ini 文件读写 API 函数可以对.ini 文件中的各种配置信息进行操作。.ini 配置文件即配置有虚拟调试环境内部的配置信息,控制着虚拟调试环境内部的组建信息,又包含虚拟数据源的相关配置信息,传递虚拟数据源的初始条件,同时可以通过写入增加配置信息。下面的.ini 配置文件中显示了基于 TCP/IP 通信的一个大致内容:

```
[HostInfo1]
    Hostip = 172.16.48.150
    Hostport = 59526
[HostInfo 2]
    Hostip = 172.16.48.151
    Hostport = 59526
...
[HostInfo N]
    Hostip = 172.16.48.152
    Hostport = 59526
[DataInfo]
  IsTimed = 1/0          ; 1 为定时发送,0 为连续发送
  TimeInterval = 30      ; 时间单位为秒,此为定时的时间间隔,只有当 IsTimed 为 1 时此值才会有效
        DataType = " unsignedint/int/char/long/double/Boolean/struct/Frame"
  DataLen = 0            ; 这个可以采用默认的或是指定的
  DataStrategy = 1/0     ; 1 为长时间传输直到进程结束,0 为某个时间段内传输
  beginTime =            ; 开始时间
  endTime =              ; 结束时间
```

上面的 HostInfo 1 到 HostInfo N 定义了一个虚拟数据源本身的配置信息、虚拟调试环境内部的组织信息和被调试程序所在主机的 IP 地址信息,这些信息体现整个的网络组织关系。后面的 Datainfo 是对虚拟数据源的初始条件要求。

2.2 虚拟调试环境与被调试程序之间的数据流模型

虚拟调试环境由多个虚拟数据源组成,每个虚拟数据源又可以是一个单独的虚拟调试环境,递归定义为集合 $\{D \mid D_1, D_2, \cdots, D_i, 1 \leqslant i\}$,每个子集又可定义为 $\{D \mid D_1, D_2, \cdots, D_{kj}, 1 \leqslant k \leqslant i, 1 \leqslant j\}$,它们通过网络连接。被调试程序可以是一个单独的应用程序,也存在位于多台计算机上同时被联合调试的情况,可定义为集合 $\{S \mid S_1, S_2, \cdots, S_j, 1 \leqslant j\}$。虚拟调试环境与被调试程序存在以下几种应用场景:

(1) D_1 对应 S_1,这种情况对一个单独的软件开发人员常用,虚拟调试环境只有一个单虚拟数据源。双方的网络通信可以看作是点对点模式,采用阻塞方式可忽略虚拟调试环境节点与被调试程序的启动顺序。虚拟数据源只需要根据被调试程序需求产生相应的数据流,按照约定好的用户通信协议的组织方式与被调试程序进行通信。

(2) D_1 对应 $S_1, S_{2j}, \cdots, S_k$,其中 $2 \leqslant k \leqslant j$。此时 D_1 与 $S_1, S_2, \cdots, S_k$ 之间存在多路数据流通信。D_1 通过多线程转发技术实现虚拟调试环境的基本功能,实现与多个被调试程序的数据流交互,通过设置通信协议消息头中相应的信号位可以实现与多个被调试程序节点进行消息通信[9]。D_1 将本身产生的数据封装后按线程的优先级高低分路转发给 $S_1, S_2, \cdots, S_k$;D_1 也按照线程的优先级高低接收来自 $S_1, S_2, \cdots, S_k$ 反馈的信息。

(3) $D_1, D_2, \cdots, D_l$ 对应 S_1,其中 $2 \leqslant l \leqslant i$。与情况(2)类似,此时 $D_1, D_2, \cdots, D_l$ 与 S_1 之间存在多路数据流通信。D_k 若用多线程技术实现多种功能,使用多路复用技术与 S_1 的通信可显著提高通信速率[10];若 D_k 只是以单进程单个功能与被调试程序交互,此时被调试程序需要处理多路数据流,如作为服务器处理大量的数据计算。

(4) $D_1, D_2, \cdots, D_l$ 对应 $S_1, S_2, \cdots, S_k$,其中 $2 \leqslant l \leqslant i, 2 \leqslant k \leqslant j$。这时的网络比较复杂,双方既可能都是单进程情形,也有均为多线程进行通信的情况;要保证各个虚拟调试环境节点内部及与被调试程序之间传输就绪的协调一致,应使用时间戳或使用固定时间槽模式进行同步的通信,为每个被调试程序和每个虚拟数据源提供固定时间槽[11]。

2.3 虚拟调试环境与被调试程序之间的通信

在虚拟调试环境之上对被调试程序进行调试,虚拟调试环境应对被调试程序件具备透明性并提供无缝连接,以消除虚拟调试环境与现场环境之间的差异,同时屏蔽被调试程序与虚拟调试环境之间的虚实,使被调试程序如同身临其境。实际的应用中存在不同的通信技术,如 TCP/IP 下的 Socket 通信,现场总线技术、串口通信,telnet 通信等,根据具体的应用需求选择不同的通信技术,文中以 windows 平台上 TCP/IP Socket 通信作为实现研究对象。

作为当前 TCP/IP 网间编程最为流行的 Socket 编程最早由 4BSD Unix 首先提出用于解决网络间进程通信问题。在 windows 平台上,由 Winsock.dll 等动态链接库提供网络编程接口。在网络中,Socket 用一个三元组描述:协议、本地地址、本地端口;一个完整的 Socket 连接用一个五元组描述:协议、本地地址、本地端口、远程地址、远程端口。每一个 Socket 由操作系统分配一个本地唯一的 Socket 号[12]。Socket 有面向连接与面向无连接两种主要的操作方式。面向连接方式即流方式(Stream Socket),使用 TCP 协议,这种方式下进行通信的双方必须先建立一种虚拟的连接关系。每一次完整的数据传输都要三个过程:建立连接、使用连

接和终止连接。这种方式可靠,连接持续,数据收发一致且内容相同。面向无连接方式即数据报方式(Datagram Socket)使用UDP协议。每个分组都携带完整的目的地址,各分组在系统中独立传输[13]。

文中以TCP/IP的C/S模型和面向连接的流式套接字接口为例实现虚拟调试环境与被调试程序通信。虚拟调试环境充当服务器的角色,被调试程序作为客户机角色。通信原理为:服务器端和客户端都必须建立通信套接字连接,首先服务器端套接字进入监听状态,然后客户端套接字发出连接请求,服务器端收到请求后,建立另一个套接字进行通信。原来负责监听的套接字仍进行监听,如果有其它客户发来请求,则再建立一个套接字[14]。默认状态下连接请求的最大客户个数为5个。整个通信过程的基本流程见图1。图1中即显示了C/S模型下的基本流程,也提供了相应的API函数。在开发过程虚拟调试环境中调用这些API函数可以实现网络通信功能。与被调试程序进行交互通信,虚拟调试环境使用与被调试程序一致的用户协议。虚拟数据源输出的调试数据经过用户协议的封装,然后通过Socket通信传输给被调试程序。

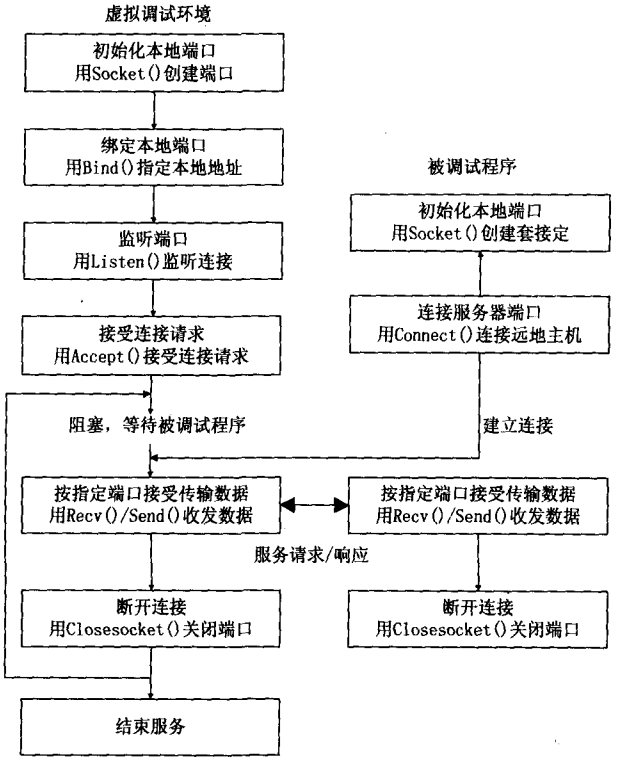

图1　基于Socket的面向连接的C/S基本结构图

2.4　虚拟调试环境的配置与管理

提供友好和可使用性的人机交互界面,将虚拟数据源参数配置以图形界面的形式展现给调试人员,方便调试人员对虚拟调试环境的理解;针对虚拟调试环境占用计算机比较多时,通过人机界面可以实现远程自动部署,统一管理虚拟调试环境[15]。人机调试界面通过网络连接远程的虚拟调试环境节点对其进行管理,各个节点以独立的调试界面显示,拥有独立的操作窗口,互不影响。这些窗口中的初始参数可以从.ini配置文件读出,也可以重新配置并存储到.ini文件中。通过界面配置可以灵活地配置各种类型调试数据(如极限数据)进而达到对被调试程序的全方位的充分的调试。

重现软件调试或测试发现问题时的场景是一件令人头痛的事情,特别是对于概率事件小的bug的场景的重现更是棘手,因此保存调试过程中出现bug时的调试环境非常重要,有必要存储虚拟调试环境产生的数据以便于重现这个场景从而更有利于定位问题和解决问题。解决的方法是:将虚拟调试环境生成的大量数据流发送给被调试程序的同时存储于本地或远程的数据库中;将虚拟调试环境的各种配置参数如网络组成信息、环境配置信息、虚拟数据源信息、异常信息等存储于与存储数据的同一个数据库中。以上两种动作都以时间戳为标记;当它们存储于非本地的数据库中时需要标记上相信的虚拟数据源的IP地址。当需重现某个场景时,调用此时间点的数据信息即可重现当时的场景,进而得到可重复的执行过程。

3　结束语

文中提出了针对基于网络进行调试的虚拟调试环境,该环境虚拟现场设备,提供各种与现场设备一致的虚拟数据源,可以满足在现场调试环境下的各种需求,使开发人员在去现场调试部署之前就能够对被调试程序进行充分的调试,进而缩短现场调试和软件开发周期,降低现场调试风险,最终节省开发资金,使企业生产早日投入运营,达到软件开发公司与生产公司的双赢。

参考文献:

[1] Albertsson L, Magnusson P S. Using complete system simulation for temporal debugging of general purpose operating systems and workloads[C]//Proceedings of MASCOTS 2000. [s.l.]:IEEE Computer Society Press, 2000:191-198.

[2] Magnusson P S, Christensson M, Eskilson J, et al. Simics: A Full System Simulation Platform[J]. IEEE Computer, 2002, 35(2):50-58.

[3] Balzer R M. EXDAMS-Extendable Debugging and Monitoring System[J]. AFIPS Proc, 1969, 34:567-580.

[4] Albertsson L, Magnusson P S. Simulation-based Temporal Debugging of Linux[C]//Proceedings of the Second Real-Time Linux Workshop. [s.l.]:[s.n.], 2000.

[5] You L L, Rajgopal N, Wimble M D. Debugging system with

(下转第53页)

应该达到的目标,根据图 5 的软件质量综合特性和定量指标度量模型[12]将质量特性细化,该定量指标度量模型是在 ISO/ IEC 9126 软件内部和外部质量特性模型的基础上改进得到的。

该模型分为三层:

第一层是面向管理的质量特性,每一个质量特性是用以描述和评价软件质量的一组属性,代表软件质量的一个方面;

第二层质量子特性是上层质量特性的细化,一个特定的质量特性可以对应若干个质量子特性;

第三层是度量特性,对于度量特性只要按照相应的标准进行度量,就可为质量特性和质量子特性提供定量值。

其中各定量指标是由软件工程工作人员在长期的工作经验中得出的,直观明了,具有可操作性、实用性、可行性和有效性。根据每个软件产品构件的具体情况,可以选取其中的部分特性进行度量,也就只需要确定部分质量特性的质量目标。

4　结束语

软件配置管理与软件质量控制有许多的交叉点,文中将它们的理论技术相结合,使得在软件配置管理的过程中加强对质量控制的支持,通过对软件过程构件和软件产品构件的控制,实现对软件开发过程与软件产品构件的质量控制。

参考文献:

[1] 徐晓春,李高健. 软件配置管理[M]. 北京:清华大学出版社,2002.

[2] 范文峰. 软件过程度量的研究与应用[D]. 上海:上海大学, 2004.

[3] 徐如志. 基于复用的软件过程改进方法与优化控制技术研究[D]. 上海:复旦大学,2004.

[4] 杨芙清,梅　宏,李克勤. 软件复用与软件构件技术[J]. 电子学报,1999, 27(2):68-75.

[5] 马　亮. 基于规约匹配的构件检索[J]. 小型微型计算机系统,2002,23(10):1153-1157.

[6] Elrac W A, Carleton A D. 度量软件过程[M]. 任爱华,刘又诚,译. 北京: 北京航空航天大学出版社,2002.

[7] 李　健,金茂忠. 软件过程度量技术的研究[J]. 计算机工程与应用,2001(5):86-90.

[8] 马　亮. 基于规约匹配的构件检索[J]. 小型微型计算机系统,2002,23(10):1153-1157.

[9] GB/T 16260.3-2006/ISO/IEC TR 9126-3: 软件工程产品质量. 第 3 部分:内部量[S]. 2003.

[10] Damiani E, Fugini M G, Bellettini C. A hierarchy-aware approach to faceted classification of objected-oriented components[J]. ACM Transactions on Software Engineering and Methodology,1999,8(3):215-262.

[11] ISO/IEC 9126. Software Engineering - Product Quality - Part 3: Internal Metrics, ISO[S]. 2003.

[12] 程　诺,万　琳,张　威. 基于量化指标分析的软件质量度量方法[J]. 北京化工大学学报,2007,34(增刊Ⅰ): 126-129.

(上接第 45 页)

[9] Yan S C,Li M J, Zhang H J,et al. Ranking Prior Likelihood Distributions for Bayesian Shape Localization Framework [C]//in Proceeding of the 9th International Conference on Computer Vision. ICCV'03. France, Nice:[s. n.],2003.

[10] 任爱华. 计算机图形学[M]. 北京:北京航空航天大学出版社,2005.

[11] 王新成. 高级图像处理技术[M]. 北京:中国科学技术出版社,2001.

[12] 靳国栋,刘衍聪,牛文杰. 距离加权反比插值法和克里金插值法的比较[J]. 长春工业大学学报(自然科学版),2009,28(4):813-818.

(上接第 49 页)

portable debug environment-independent client and non-portable platform-specific server[M]. USA:ACM,1998.

[6] 杨　帆,张彩丽. 一种通用远程监控软件的设计与实现[J]. 计算机测量与控制,2006,14(10):1417-1419.

[7] 王　亮,张石磊,张春雷. 调试嵌入式系统方法研究[J]. 硅谷,2008(10):22-22.

[8] 吕国芳, 唐海龙,李　进. 基于 Modbus RTU 的串口调试软件的实现[J]. 计算机技术与发展, 2009,19(9):236-238.

[9] 曹　力,刘晓平. 异域网中分布式仿真系统的通信模型[J]. 系统仿真学报,2007,19(13):2951-2954.

[10] 郑大虎. 分布式日志服务系统关键技术的研究[D]. 西安:西安电子科技大学,2005.

[11] 张　龙. 工业实时以太网:现场总线与以太网技术在工业自动控制系统中的必然趋势[J]. 自动化博览,2007,24(6):44-48.

[12] 何　进,谢松巍. 基于 Socket 的 TCP/IP 网络通信模式研究[J]. 计算机应用研究,2001,18(8):134-135.

[13] 刘利强,吴永英,王勇智. IPv6 下 Socket 网络编程的研究与实现[J]. 计算机技术与发展,2006,16(6):201-205.

[14] 黄显信. 利用 Socket 通信进程实现不同数据库间数据的同步[J]. 计算机与数字工程,2003,31(5):28-30.

[15] 翟廉飞,柴天佑,高忠江,等. 制粉系统智能解耦控制的分布式仿真实验平台[J]. 系统仿真学报,2006,18(7):1824-1828.