多机并发系统中通信模型分层抽象的方法*)

齐 微 陈 平 李青山
(西安电子科技大学软件工程研究所 西安 710071)

摘 要 针对多机并发系统的复杂性,为了辅助用户能从多个角度和层次全面地理解并发系统,就需要逆向产生出能够反映软件系统框架结构的高层架构。基于此本文以进程为边界,提出了一种分层抽取多机并发系统通信模型的方法。此方法基于反射和开放编译的植入机制来获取所需要的动态信息,在此基础上运用分层抽象的策略,分别从系统、节点、进程三个层次对多机并发系统的通信结构和设计结构进行逆向恢复,最后对该方法进行系统的实验研究。结果表明,分层抽象所得到并发系统的通信模型能够正确、有效地反映系统设计时的高层架构关系。

关键词 多机并发系统,逆向工程,分层抽象,进程

Hierarchical Abstraction of Communication Model for Multi-host Concurrence System

QI Wei CHEN Ping LI Qing-Shan
(Software Engineering Institute, Xidian University, Xi'an 710071)

Abstract For the complexity of multi-host concurrence system, in order to help users comprehend a concurrence system at all aspects and levels, it is necessary to reversely recover and abstract the high-level architecture, which can reflect the framework and holistic behavioral features of the software system. An approach of hierarchical abstraction of communication model for multi-host concurrence system is brought forward. Based on the instrumental mechanism with the technology of reflective open compiler, dynamic information can be obtained, then the communications architecture and framework can be reversely recovered from three levels as system, node and process by using the method of hierarchical abstraction. Finally, a case study is given to verify this approach. The experimental results show that the recovered communication model is correct, effective and can reflect the high-level structure of the source system in detail.

Keywords Multi-host concurrence system, Reverse engineering, Hierarchical abstraction, Process

1 引言

逆向工程作为辅助程序理解的重要手段,逐渐成为软件工程领域的研究热点。随着软件复用技术的不断发展,使用的遗产系统相应增多。对于后续开发和维护人员来说,现有遗产系统的理解显得尤为重要,因此遗产系统的持续性演化变得十分重要。作为程序理解的有效支持手段,逆向工程在遗产系统的成功演化中扮演着重要的角色[1]。基于多机并发系统的并发、分布的动态本质,从逆向工程的角度来研究多机并发系统软件体系结构的恢复对于理解并发系统具有重要的意义。

目前逆向工程中常见的体系结构恢复方法大都是从子系统的层次和角度获取目标系统高层信息。子系统关系体现了软件系统总体功能方面高层结构的一个侧面,它强调的是用户设计面向对象软件系统时从需求特征和整体功能角度考虑的类的划分关系。然而,要充分理解大型的、具有分布并发特征的目标系统行为特征,就必须从多个层次和侧面分析并抽取系统组件间的结构关系以及进程之间的通信关系。在逆向工程中,进程结构和子系统层面的结构划分反映了用户理解系统高层架构的不同关注点[2]。针对于此本文提出了一种多机并发系统通信模型的分层抽象的方法:以进程为边界,自上而下,分别从并发系统、通信子节点、进程模块三个层次对多机并发系统的通信结构和设计结构进行逆向恢复。借助于逆向恢复的层次化的模型,用户可以更好地理解分布、并发的系统。

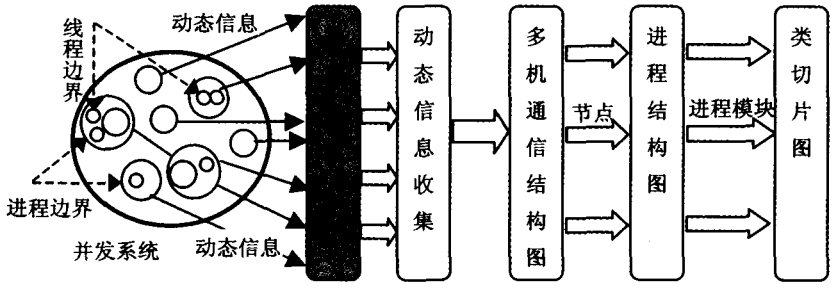

图1 分层抽象的模型

*)基金支持:国家自然科学基金(项目编号:60473063),国家教育部博士点基金(项目编号:20030701009)及"十五"国防预研项目(项目编号:41306060106)。齐 微 硕士研究生,研究领域为逆向工程,面向对象技术和软件体系结构。陈 平 博士,教授,博士生导师,主要研究领域为电子信息系统软件开发理论与技术,面向对象技术,软件工程,逆向工程。李青山 博士,副教授,主要研究方向为逆向工程,程序理解,面向对象和软件体系结构。

2 分层抽象的模型框架

多机并发系统[3]是运行在一系列自治处理节点上的系统，每个处理节点有各自的物理存储器空间并且信息传输延迟不能忽略不计，在这些处理单元间有紧密的合作。从物理的观点来看，多机并发系统由多个计算机节点组成，各节点之间通过通信网络互相连接。从逻辑观点看，多机并发系统各处理节点之间的通信其实质仍然是通过进程通信机制(IPC)。因此，对于复杂的、具有并发特征的面向对象的多机并发系统逆向工程，进程交互关系和通信结构的抽取具有很重要的价值。所以本文以进程为边界，采用基于反射和开放编译的植入机制[4,5]和共享内存机制来获取系统运行时的动态信息，进而分析进程间交互和进程内部的对象间交互，自上而下，分别从系统、节点、进程三个层次对目标系统的通信结构和设计结构进行恢复。其分层抽象的模型框架如图 1 所示。

3 分层抽象的实现

3.1 系统层次的多机通信结构图的逆向恢复

为了帮助用户从系统的层次上把握多机并发系统的高层通信架构，多机通信结构图必须能够反映多机并发系统中各物理节点间的通信框架，包括通信依赖关系、通信依赖类型。

定义 1 多机通信结构图 MCSG(Multi-host Communication Structure Graph)：多机并发系统中，MCSG＝(V, R, Ψ)，其中 V 为各关注节点的集合，R 为节点间通信关系的集合，R＝{Tcp_Udp, Rpc, Corba, Dcom}，Ψ:$V\times V\to R$，设 r＝φ(v1,v2)∈R，表示 v1 通信依赖于 v2，通信类型为 r。

多机通信结构图为有向图，为多机并发系统中关注节点之间的通信依赖关系的图形表示，从系统层次上展现了目标系统的最高层通信架构。通过运行植入后的目标系统来收集动态信息，在此基础之上过滤得到多机通信文件 HostsComm. xml，将其解析后，产生对应的模型视图来进行呈现，算法如图 2 所示。

```
算法：多机系统通信结构图的恢复
输入：多机通信信息文件 HostsComm.xml
输出：多机通信结构图 MCSG
步骤：
do
    V=∅; Ψ=∅;
    加载通信信息文件 HostsComm.xml;
    foreach host node HN of HostsComm.xml do
        V ← HN  //向 V 中追加 HN，添加节点包信息
    od  // End of foreach HN
    foreach host node HN of HostsComm.xml do
        foreach dependency node DN of HN do
        根据 DN.type 构造 HN 和 DN 间通信关系 r，添加节点间
            通信依赖关系
        Ψ ← r  //向Ψ 中追加 r
        od  // End of foreach DN
    od  // End of foreach HN
od
```

图 2 多机通信结构图的抽取

该算法的执行效率取决于目标系统中所有通信依赖关系的数目，为各节点依赖关系的总和。所以，本算法不但收敛，而且实现所花费的时间代价并不高。

3.2 节点层次的进程结构图的抽取

为了促进用户对多机并发系统中节点内部进程间交互剧情和进程实体间通信结构的理解，针对多机通信结构图中的每个节点，产生其对应的进程结构图。

定义 2 进程模块：进程多次运行的抽象，对应于程序特定的一段代码。

定义 3 进程结构图 PSG(Process Structure Graph)：对于多机并发系统每个节点，PSG＝(V, R, Ψ)，其中 V 为节点执行过程中所有进程模块和资源模块的集合，设进程模块集合为 PM，资源模块集合为 RM，则 $V=\{v\mid v\in PM$ 或 $v\in RM\}$，R 为模块间关系的集合，包括创建关系和通信关系，Ψ:$V\times V\to R$，设 $r=\varphi(v1,v2)\in R$，若 $v1\in PM$，$v2\in PM$，则 r 为创建关系，$v1$ 创建 $v2$；若 $v1\in PM$，$v2\in RM$，则 r 为通信关系，$v1$ 通过 $v2$ 与其它进程模块通信。

进程结构图是一种静态体系结构图，是运行的目标系统中进程及其通信关系的抽象模型，其反映出节点内进程模块间交互关系。它是利用植入代码后的目标系统执行时产生的动态信息而抽取出每个节点内部的进程模块通信信息_PreProcessInfo. xml 文件，然后再对其解析，生成该节点对应的进程结构图。为了抽取能反映出节点内部相对完整的进程模块和这些进程模块间尽可能全面的依赖关系，就需要利用多次运行植入后目标系统而产生的反映进程间交互的动态信息增量式地构造 PSG。进程结构图的抽取算法如图 3 所示。

```
算法：针对多机并发系统中关注节点 N，生成进程结构图
输入：进程模块通信文件_PreProcessInfo.xml
输出：进程结构图
do
  PM=∅; RM=∅; Ψ=∅;
  加载节点 N 对应的_PreProcessInfo.xml
  foreach Node of _PreProcessInfo.xml do
      if (Node.type 是通信关系)
      then do
        if (Node.rm∉ RM) then RM←Node.rm fi
        if (Node.pm∉ PM) then PM←Node.pm fi
          构造 Node.pm 和 Node.rm 间通信关系 r
          Ψ ← r  //追加 PM、RM 及依赖关系至 PSG 中
          od
      else od  //创建关系
        if (Node.p_pm∉ PM) then PM←Node.p_pm fi
        if (Node.c_pm∉ PM) then PM←Node.c_pm fi
            构造 Node.p_pm 和 Node.c_m 间创建关系 r
            Ψ ← r  //追加 P_PM, C_PM 及创建关系至 PSG
      od
      fi
od  // End of foreach
od
```

图 3 进程结构图的抽取算法

该算法的执行效率取决于节点 N 内进程交互情况，为进程模块间创建关系数目和进程模块与资源模块间通信关系数目的总和。而对于一个软件系统，进程通信信道和进程模块的个数总是有限的，而且相对稳定，所以，本算法不但收敛，而且实现所花费的时间代价并不高。

3.3 进程模块层次的类切片图的抽取

在使用面向对象语言开发的系统中，真正执行应用程序和业务逻辑计算的基本单位是类的实例化对象，即，进程实例化类对象，对象真正完成计算的执行。为了让用户把握目标系统的设计结构，充分了解进程运行中所涉及的类结构信息，针对进程结构图中的每个进程模块，产生其对应的类切片图，建立进程模块到类结构之间的映射关系，从而反映出进程模块执行中所用到的类的属性及其方法等信息。

定义 4 类切片图 CAG(Class Aspect Graph)：进程模块一次或多次执行过程中类切片集合的图形表示。

在收集动态信息阶段，对每个节点均收集了其内部的静

态模型信息_StaticInfo. xml 文件，对该信息进行解析，可以抽象出各进程模块运行时所用到的类结构信息，进而产生各进程模块对应的类切片图。其算法如图 4 所示。

```
算法：针对各节点中每个进程模块 PM，生成类切片图
输入：静态模型信息文件_StaticInfo.xml
输出：类切片图
do
  加载静态信息模型文件_StaticInfl.xml;
  foreach Node P of _StaticInfo.xml  do
    if (P. pm∈ PM )
     then do
       foreach class aspect P_CA of P do  // CA 对应类为 cls
         if (PM_CA∈ PM)  // cls 对应的类切片信息 PM_CA
           then  PM_CA=PM_CA ∪ p_CA;   if
         else
           PM_CA=P_CA;
       od
    if
    foreach class aspect information of PM  do
       添加对应的类切片 CA 至 PM 的 CAG
     od
od
```

图.4　类切片图的抽取算法

在该算法中，在修改 PM 类切片信息时必须要进行检查相关类的属性，方法信息不能重复记录。算法的执行效率与 PM 执行中方法调用信息及属性读写信息的数量有关。

4　实验研究

本文所提出的多机并发系统通信模型的逆向分层抽象的方法已经在我们开发的逆向工程的工具 XDRE[6]（XiDian Reverse Engineering）中实现了，并将逆向恢复得到的多机通信结构图、进程结构图以及类切片图无缝集成到 Rational Rose 环境中。为了验证该方法的有效性，通过一个具体例程对所做的工作进行验证。

该例程采用开源聊天系统——爱盟聊天系统，一个具有典型多机并发通信的程序。该系统采用 C++语言开发，运行于 Windows 平台。该聊天室支持多人同时聊天（用户和用户之间可以发送密语）；用户之间可进行文件收发；管理员作为超级用户来管理聊天室，发布广告，清理聊天记录，以及将一些非法用户提出聊天室等等。

在本次实验中，分别对目标系统子节点植入和收集动态信息。首先从系统的层次上对目标系统的体系结构进行恢复，根据整合后得到关于该系统通信架构的动态信息 HostComm. xml 文件如图 5 所示，对该文件进行解析得到相应的多机通信结构图，如图 6 所示。其次再从节点的层次上恢复系统的模型框架，根据收集到的关于目标系统通信子节点内部的进程模块通信信息_PreProcessInfl. xml 文件，对该文件进行解析得到相应的进程结构图，如图 7 所示。最后根据收集到的关于每个节点的静态模型信息_StatciInfo. xml 文件，得到相应的类切片，目标系统服务器端 ChatRoom 进程模块所对应的类切片如图 8 所示。

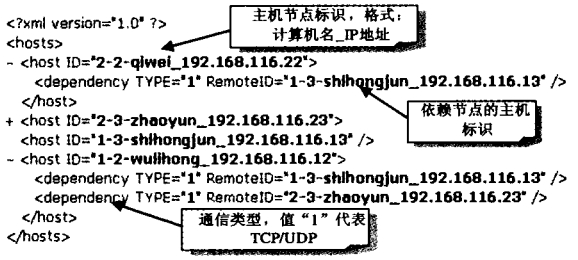

图 5　HostComm. xml 文件

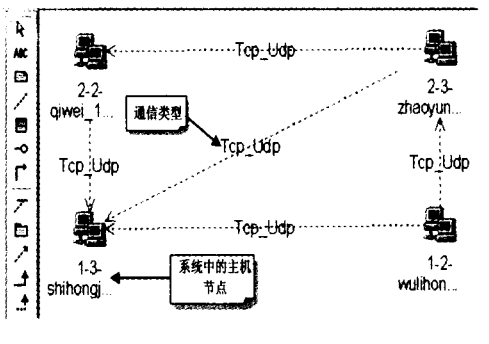

图 6　多机通信结构图

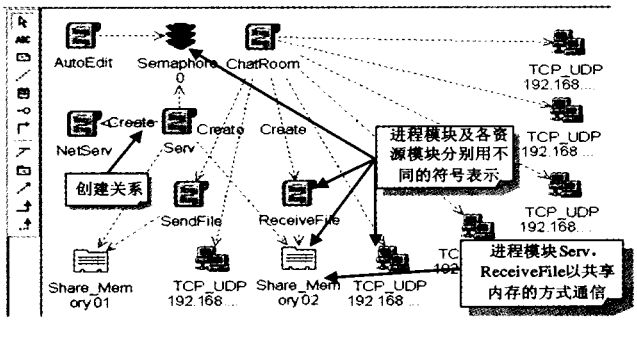

图 7　服务器端进程结构图

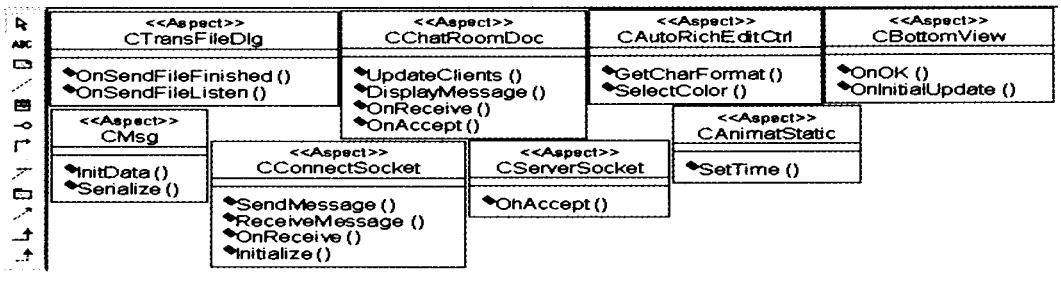

图 8　ChatRoom 进程模块对应的类切片图

结束语　要充分理解具有并发、分布特征的多机并发系统整体行为特征和把握进程间交互关系和通信结构，就需要从多个层次和角度抽取系统组件间的结构关系。基于此，本文描述了一种逆向恢复多机并发系统通信模型的方法，完成的工作如下：

（1）研究相关的软件体系结构的恢复技术，给出多机并发系统通信结构逆向恢复的方法，确定从系统、节点、进程三个层次对多机并发系统的通信结构和设计结构进行逆向恢复。

(2)详细设计并实现多机并发系统通信模型的分层抽象。

(3)最后通过系统的实验研究,验证了该方法的有效性。

目前仅从系统、节点、进程层次对多机并发系统通信结构进行恢复,如何处理多线程并发问题是下一步的研究重点。

参 考 文 献

1 Chikofsky E J, Cross J H II. Reverse Engineering and Design Recovery: A Taxonomy. IEEE Software, 1990, 7(1): 13~17

2 Kiczales G, Lamping J, Mendhekar A, et al. Aspect-Oriented Programming. In: Proceedings of the European Conference on Object-Oriented Programming (ECOOP), June 1997

3 Liu M L. Distributed Computing Principles and Application. 清华大学出版社, 2004. 4

4 陈平. 反射结构和对象标识的研究:[博士学位论文]. 西安电子科技大学, 1991

5 李青山,陈平,王伟,宋海鸿. 逆向工程中反射植入的研究. 计算机学报, 2004, 4

6 Li Q S, Chen P. XDRE 2.0 Reference Manual Software Engineering Institute. Xidian University, 2004

7 杜宽利. 多机通信结构图抽取与状态图抽象的研究:[硕士学位论文]. 西安电子科技大学, 2005

(上接第 242 页)

令 $\overrightarrow{sp}=\{x, c, smsg, srmsga, msg, armsg, omsg, dmsg, bomsg, bdmsg\}$,

则服务器端相应的进程表达为:

$$Sa(\overrightarrow{sp})\triangleq([s=synchrmous]x(smsg).\bar{x}(stmsg) + [s=asychromous]x(amsg).x(c).c(armsg) + [s=oneway]x(omsg) + [s=datagram]x(dmsg) + [s=batched]([t=batchoneway]x(bomsg)+ [t=batchdatagran]x(bdmsg)).Sa$$

4.2 利用协议实体对于协议进行刻画

本节对 Ice 协议进行完整的刻画。说明在一次请求调用的过程中,各协议实体如何协同地工作,从而反映 Ice 协议分布、并发的特性。如图 4,分别定义客户端应用层、消息层、编码/解码层、传输层进程为 Clp, Cm, Cc 和 Ct;服务器端对应层进程为 Sap, Sm, Sc 和 St。在一个请求调用过程中,各协议实体的进程代数刻画如图 4。

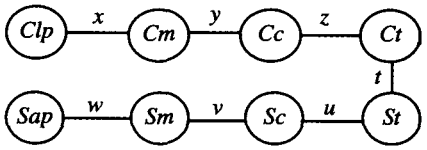

图 4 Ice 协议的流图

令 $\overrightarrow{cp}=\{x, requestmsg, replymsg\}$,则

$Clp(\overrightarrow{cp})=\bar{x}(requestmsg).x(replymsg).Clp$

Clp 进程沿 x 通道发出调用请求 $requestmsg$,沿 x 通道接收响应 $replymsg$。

令 $\overrightarrow{cmp}=\{x, y, reqestmsg, replymsg, msg, rmsg\}$,

则

$$Cm(\overrightarrow{cmp})=(x(reques\ tmsg).\bar{y}(msg) + y(rmsg).\bar{x}(replymsg)).Cm$$

Cm 进程沿 x 通道接收请求 $requestmsg$,组合成 Ice 的协议消息 msg,沿 y 通道送出去;或者接收沿 y 通道送来的消息 $rmsg$,转换成应答消息 $replymsg$ 沿 x 通道回送给 Clp。

令 $\overrightarrow{ccp}=(y, z, msg, marshalingmsg, unmarshalingmsg\}$,

则

$$Cc(\overrightarrow{ccp})=(y(msg)\bar{z}(marshalingmsg)+ z(marshalingmsg).\bar{y}(unmarshalling)).Cc$$

Cc 进程主要负责消息的编码/解码工作,沿 y 通道接收原始消息 msg,编码成 $marshalingmsg$ 消息并沿 z 通道送出去;或者接收来自 z 通道的编码消息 $marshalingmsg$,解码成 $unmarshalling$ 消息,并沿 y 通道送出。

令 $\overrightarrow{ctp}=\{z, t, msg\}$,则

$Ct(\overrightarrow{ctp})\triangleq(z(msg).\bar{t}(msg)+t(msg).\bar{z}(msg).Ct$

Ct 进程主要负责在服务器和客户端之间传输消息。沿通道 z 接收消息 msg,并沿 t 通道传输到服务器端;或者沿 t 通道接收消息,并沿 z 通道传回。

整个客户端的行为如下:

$client(t, msg)\triangleq Clp|Cm|Cc|Ct$

同理可以描述服务器端各进程,本文不再赘述。

结论 本文利用 pi 演算对于 Ice 协议建模,从交互和协议实体两方面刻画了协议。pi 演算着眼于并发进程之间传递的消息,通过描述消息交换来说明系统成分与环境之间的通信行为。它以直观、透明的方式反映通信行为,省略了传统的形式描述技术必须考虑的大多数诸如程序变量以及变量值这样的细节,特别适用于说明通信协议。同时 pi 演算是基于严格代数基础的形式化方法,利用 pi 演算描述 Ice 协议,消除了理解上的二义性,而且为进一步的演算提供了形式化基础,可以在此基础上讨论 Ice 协议的活动性与安全性问题。

参 考 文 献

1 Henning M. A new approach to object-oriented middleware. IEEE Internet Computing, 2004, 8(1): 66~75

2 Henning M, Spruiell M. Distributed programming with ICE. www.zero.com. 2003

3 Milner R, Parrow J, Walker D. A Calculus of mobile process, parts I and II. Information and Computation, 1992, 100: 1~77

4 Milner R. The polyadic π-calculus: a tutorial:[Technical report]. Department of Computer Science, University of Edinburgh, 1991

5 Milner R. Communication and mobile systems: the pi-Calculus. Cambridge, Cambridge University Press, UK: 1999

6 Park J, Miller R. A compositional approach for designing multifunction time-dependent protocol [A]. In: Proc. of 5th Intl. Conf. on Network Protocol. Los Alamitos, USA. IEEE Computer Society, 1997. 105~112

7 Carbone M, Nielsen M. A formal model for trust in dynamic networks. In: Proc. of the Software Engineering and Formal Methods, SEFM'03. IEEE Computer Society Press, 2003

8 Albert T R, Esterline C. Using the pi-Calculus to model multiagent systems. Lecture Notes in Computer Science, 2001, 1871: 164~179

9 焦文品,史忠植. 形式化多主体系统中的交互及交互协议. 软件学报, 2001, 12(8): 1177~1182