基于 ACE 并发编程模式的告警关联系统设计与实现①

Design and Implementation of Alarms Correlation System Based on ACE Concurrent Programming Pattern

朱建文 廖建新 朱晓民 王 纯
(北京邮电大学网络与交换技术国家重点实验室 北京 100876)
(东信北邮信息技术有限公司 北京 100083)

摘 要:对告警进行关联处理,减少冗余告警,增强告警的故障检测能力,是网络管理系统需要解决的关键问题。本文充分考虑电信运营网络管理可能出现的各种关联情况,定义了告警关联规则,运用 ACE(Adapter Communication Environment,自适配通信环境)的 ACE_task 组件设计并实现了告警关联系统。实验表明,该系统能有效剔除冗余告警,及时上报故障,方便客户定位故障。

关键词:ACE FSM 有向图

1 引言

ACE[1](Adaptive Communication Environment,自适配通信环境)是由美国华盛顿大学的 D. C. Schemit 教授及其研究组开发的用于构建高质量网络中间件的开放源代码、面向对象网络编程架构的工具包,在其中实现了许多用于并发通信软件的核心模式;它提供了丰富的可复用 C++ Wrapper Fa? ade(包装外观)和框架组件,便于并发的面向对象网络应用和服务的开发。

ACE_Task[2][3]主要用来实现 Active Object(主动对象)模式,所有的主动对象都必须从此类派生。该框架有助于增强并发的面向对象网络化应用的模块性和可扩展性,比如在对象的上下文中派生线程,以及在执行不同线程的对象之间传递消息和对消息进行排队,ACE_Task 构成了常见的并发编程的基础,比如 Active Object 和 Half-Sync/Half-Async(半同步/半异步)

本文设计实现的告警关联系统目的就是从大量的告警事件中通过告警关联操作,帮助用户方便快捷地找出故障,并提供简单的故障解决措施。本文首先从定义告警关联规则出发,介绍关联处理流程以及使用 ACE 并发编程模式设计实现告警关联系统逻辑流程图,最后通过实验数据分析来评价该告警关联系统的关联有效性。

2 告警关联规则

首先定义几个名词的概念:

(1) 告警(ALARM)。即简单的告警事件,包含告警编码(alarmcode)、告警标题(alarmtitle)、告警网元(alarmunit)、告警级别、告警时间、告警原因、告警解决措施等数据信息。告警网元由产生告警的集群、主机、帐户唯一确定。本文中所有出现形如 alarmi($i \in N$,自然数)的词泛指一个告警。

(2) 故障(FAULT)。是对单个或多个简单告警的综合或抽象,代表的是维护人员或监控人员需要关心、处理的问题。本文中所有出现形如 faulti($i \in N$,自然数)的词泛指一个故障。

(3) 告警关联规则(RULE)。即由单个或多个简单告警综合或抽象出故障的规则。本文中所有出现形如

① 基金项目:国家杰出青年科学基金(No. 60525110);国家 973 计划项目(No. 2007CB307100,2007CB307103);新世纪优秀人才支持计划(No. NCET-04-0111);电子信息产业发展基金项目(基于 3G 的移动业务应用系统);国家高技术产业化信息化装备专项项目(支持数据增值业务的移动智能网系统)

rulei(i∈N，自然数)的词泛指一个规则。

通过总结各电信运营网络的管理、运营、维护人员的经验以及对历史告警数据的分析，告警关联规则可以划分为组合规则、重复规则、独立规则、空规则四种。

2.1 组合规则

组合规则描述的是在某个时间窗内，多个不同的告警($alarm_i$、$alarm_j$、$alarm_k$、...,i,j,k∈N，自然数)按照时序先后发生后，标志某个故障(faultl,l∈N，自然数)的发生。比如在某电信智能网络中发现，当"数据库宕机"这个故障发生时，将先后上报"主控进程重启"、"与前台连接断开"、"数据库操作失败"等一系列与该故障相关的告警，像这种就可以用组合规则表达该关联情况。

根据组合规则中各个告警的告警编码及告警网元相同或相异，组合规则可以进一步细分为三种不同类型：

(1) 发生关联的 $alarm_i$、$alarm_j$、$alarm_k$、... 属于同类告警(即告警编码相同)，但产生自不同主机，他们之间组合起来产生某个故障。

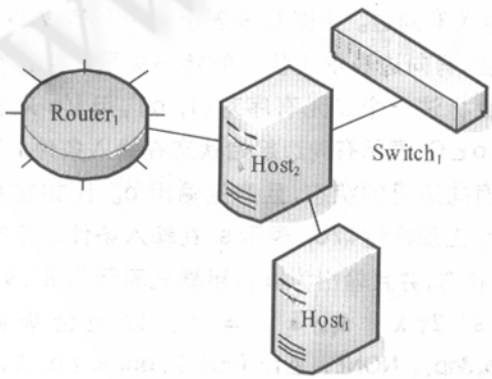


图 1　网络逻辑拓扑图

以图 1 为例，可以看出 Host2 与 Host1、Router1、Switch1 都有逻辑关系，假设这里逻辑关系表示为链路检测，那么当 Host1、Router1、Switch1 都上报"与 Host2 的链路断开"的告警时就可以推断出"Host2 主机宕机"或者"Host2 网络链路断开"的故障了。

(2) 发生关联的 $alarm_i$、$alarm_j$、$alarm_k$、... 属于不同类告警(即告警编码不同)但产生自同一个主机，比如某个主机的所有进程都上报了各自的进程重启告警(不同的进程重启告警具有不同的告警码)，那么就可以推断出该主机重启的故障。

(3) 发生关联的 $alarm_i$、$alarm_j$、$alarm_k$、... 属于不同类告警且产生自不同的主机，比如前面提到的数据库宕机的例子。

组合规则可以用表达式{$alarm_i$ &$alarm_j$ &$alarm_k$ & ⋯}→ $fault_l$ 表示，其中 i,j,k,l∈N，自然数。

2.2 重复规则

重复规则定义的是同一类告警在规定时间窗内连续发生 n 次，即告警数量(alarmcount)为 n，则可以推断出产生了一个故障。比如"主控进程重启"告警频繁发生，在初始化条件正确的情况下，可以推断出"主控进程不稳定"的故障。

重复规则用可以用表达式{$alarm_i$ | n} → $fault_l$ 表示，其中 i,j∈N，自然数。

2.3 独立规则

独立规则定义的是当一些比较重要的告警发生时，本身就代表一个故障的出现。比如"主机宕机"、"硬盘空间不足"等告警的产生本身就会影响网络业务，而且不会自动复原。这种类别的告警就可以通过独立规则关联到对应的故障。

独立规则可以用表达式{$alarm_i$} → $fault_j$ 表示，其中 i,j∈N，自然数。

2.4 空规则

这类告警并不反映故障，只是作为一种事件上报，属于知识告知，不需要关注。

空规则可以通过表达式{$alarm_i$} → null 表示，i∈N，自然数。

关联规则还需要考虑网络拓扑情况，可以通过对规则划分有效范围来完成，定义同一规则内各告警发生的范围为如下几种：NONE，SAME_CLUSTER，SAME_HOST，SAME_USER。NONE 表示规则内的告警可以不用约定具体来自的网元；SAME_CLUSTER 表示同一个集群，集群即划定的主机的集合；SAME_HOST 表示同一规则内的告警只有来自同一个主机才有效；SAME_USER 表示同一规则内的告警只有都来自同一个帐户才有效。

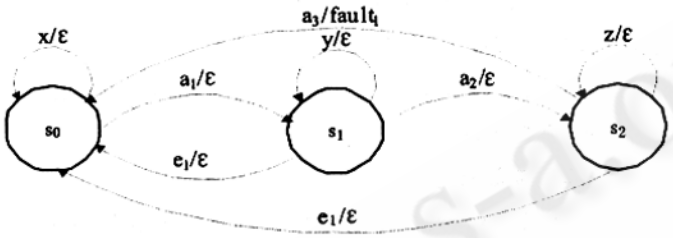
分析各种关联规则的表达式，再综合规则范围域以及时间窗信息，关联规则可以表示为四元组 RULE：RULE = (A，D，T，F)

A 是规则对应的告警序列，由一个或多个不同的二元组 ALARMPAIR = (ALARMINFO，alarmcount)按照一

定的先后顺序以符号 &组合而成，其中 ALARMINFO 表示告警信息，包含告警编码、告警标题、告警网元，可用三元组 ALARMINFO =（alarmcode，alarmtitle，alarmunit）表示，其中 alarmunit 为 ANY 时表示任何网元。

D 是规则应用的范围域，取值为 NONE，SAME_CLUSTER，SAME_HOST，SAME_USER 之一。

T 是规则有效的时间窗，即形成规则最长的有效时间，以秒为单位。



$x \in ALARM \& x \neq a_1$　$y \in ALARM \& y \neq a_2$
$z \in ALARM \& z \neq a_3$　ε为null
$e_1 \in TimeOut(>\Delta t)$　s_0为初始状态　$S=(s_0,s_1,s_2)$

图 2　状态转换图

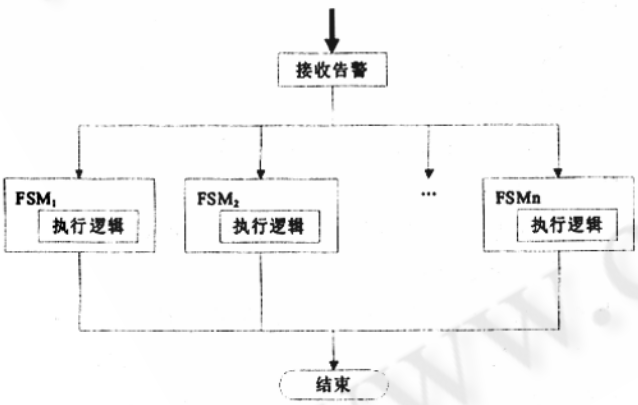


图 3　整体流程图

F 是规则对应的输出故障。只有在时间窗 T 内，在 D 范围域输入 A，才形成输出 F。

3　关联处理流程

3.1 FSM 概念

FSM（finite state machine）即有限状态机，可以定义为六元组 M[4]：

$M = (I, O, S, s0, \delta, \lambda)$

I 是有限非空的输入符号集合

O 是有限非空的输出符号集合.

S 是有限非空的状态集合.

$s0 \in S$ 是初始状态.

$\delta: S \times I \to S$，是状态转移函数。当有限状态机 M 处于状态 $s \in S$ 时，接收到输入 $i \in I$，转移到下一个状态 $\delta(s, i) \in S$.

$\lambda: S \times I \to O$，是输出函数。当有限状态机 M 处于状态 $s \in S$ 时，接收到输入 $i \in I$，则输出 $\lambda(s, i) \in O$.

每一条告警关联规则可以对应到一个有限状态机，I 可表示为告警关联规则相应的输入告警序列，O 表示为告警关联规则输出故障，S 为告警关联规则实例在各个不同时刻等待点。可以用有向图来形象表示规则在状态机中的状态转换：一个有向图 G =（V，E）是一个有序的二元组，其中 $V \neq \Phi$，是 G 的顶点集合，E 是笛卡尔积 $V \times V$ 的多重子集，其元素称为有向边。用图 G 来表示一个 FSM：V 与 S 一一对应，有向边表示了状态的转移关系。并且在有向边上标注一个二元有序对（i，o），用 i/o 来表示，$i \in I$，$o \in O$，表示有向边起始状态在输入条件 i 下转换到有向边目的状态，且产生输出 o。比如在有向边 s_1s_2 上加标注 i_1/o_1，表示 s_1 在输入条件 i_1 下转换到 s_2 状态，并且输出为 o_1，用算式表示为 $\delta(s_1, i_1) = s''2$，$\lambda(s_1, i_1) = o_1$。以组合规则 $\{ap_1 \& ap_2 \& ap_3, NONE, \Delta t, fault_1\}$，$ap_1 = (a_1, 1)$，$ap_2 = (a_2, 1)$，$ap_3 = (a_3, 1)$，$a_1 = (alarmcode_1, alarmtitle_1, unit_1)$，$a_2 = (alarmcode_2, alarmtitle_2, unit_2)$，$a_3 = (alarmcode_3, alarmtitle_3, unit_3)$ 来举例，有向图如图 2 所示。

3.2 FSM 的关联处理流程

对于每一条告警关联规则建立一个对应的状态机，那么根据所有的告警关联规则构建成一个 FSM 集合{FSM1、FSM2…FSMn}，当接收到一条新告警时，轮循 FSM 集合中的状态机，执行每一个状态机的逻辑处理流程，修改状态机原有实例的状态或者形成新的状态机实例，整体流程图如图 3。

对每一个状态机来说，从接收告警到故障输出，执

行的逻辑流程图如图 4。

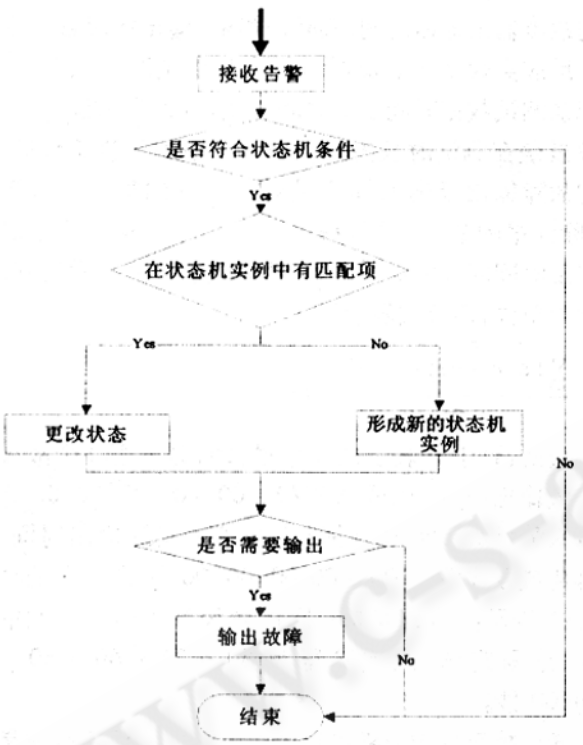


图 4　状态机告警处理流程图

4　基于 ACE_Task 的告警关联并发处理

由第 3 节的分析可知，程序中必须妥善解决多个状态机对告警处理流程的并发操作问题。单线程模式的解决方法是轮循状态机队列集，逐一执行各个状态机告警处理流程，等所有状态机轮循到之后再接收处理下一个告警。该模式的优点是告警接收、处理时序性较强，程序逻辑简单。缺点在于效率较低，实时性较差，各状态机之间耦合性较强，不利于维护。改善的有效途径是以多线程模式实现多个状态机告警处理流程的并发操作。ACE 拥有多种用于创建和管理多线程程序的类，利用它们可以设计出不同的解决方案，本系统采用 ACE_Task[1,2] 任务类实现多线程模式。

ACE_Task 任务类实现多线程模式方案用到包含多线程模式的主动对象。传统方式中所有对象都是被动的代码段，在对它发出函数调用的线程中执行。而主动对象则拥有自己的线程，在这些线程中自动发起对本对象任何函数的调用。主动对象设计模式使函数执行与函数调用去耦合，增强了并发处理能力，适用于应用广泛的生产者/消费者应用模型，我们用此模式来构造、处理本文中的问题。在 ACE 中使用任务处理结构的基类（ACE_Task 类）来实现主动对象模式，它是一种比线程更高级的构造，可用于多线程编程。应用 ACE_Task 类解决告警关联并发处理问题的序列图参见图 5。

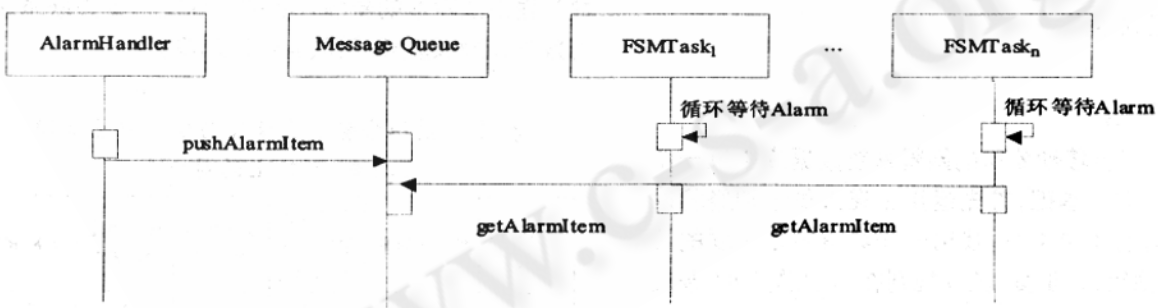


图 5　告警关联并发处理序列图

从以上流程可以看出，如果设计出的告警关联系统采用同步方式，即每一个告警顺序轮循各个状态机并执行其告警处理流程，那么告警处理效率得不到保证，故障输出响应时间受告警关联规则数量影响很大，实验表明，当系统告警关联规则数量超过 300 时，平均故障输出延迟时间大于 4 分钟，这对实时性要求较高的电信运营网络来说，远远达不到要求。因此，需要采用异步方式，使用多线程并行执行各个状态机告警处理流程。

图 5 中，AlarmHandler 是接收告警并将告警存入告警队列的线程。TFSMTask 是继承自 ACE_Task 的类，在该类的启动点 svc( ) 函数里循环等待接收告警，并进行告警关联逻辑处理。FSMTask1、FSMTask2... FSMTaskn 是 TFSMTask 类的对象，每一个对象代表对一种告警关联规则的处理，也即状态机，所有 TFSMTask

对象由 TFSMTaskQueue 类来维护，TFSMTaskQueue 类负责创建、销毁具体的 TFSMTask 对象。MessageQueue 是通过 TMsgQueue 类来实现的，该类包含一个告警队列，以及三个成员函数：pushAlarmItem()、getAlarmItem()和 onTimer()。TMsgQueue 类采用单件模式[5]，当 AlarmHandle 线程接收到新告警时，调用 TMsgQueue::getInstance()->pushAlarmItem()将新告警插入到告警队列中；当 TFSMTask 对象通过 getAlarmItem()成员函数取出一个告警时，将该告警被调用数量加 1；定时执行函数 onTimer()每隔一段时间从队列头开始，检测告警被调用数量是否大于等于 TFSMTask 对象数量来确定该告警是否被所有消费者消费完毕，如果被消费完毕就将该告警从告警队列中移除，并销毁该告警，否则退出定时执行函数。

在应用程序中主要编程实现以下三个部分：

(1) 实现 TMsgQueue 类。

(2) 创建继承自 ACE_Task 基类的 TFSMTask 类。

(3) 系统初始化时，创建 TMsgQueue 类单件实例；创建 TFSMTaskQueue 类单件实例，该单件实例根据告警关联规则列表实例化 FSMTask1、FSMTask2 … FSMTaskn，并分别调用 FSMTask::active()激活任务。

使用主动对象模式可以解除各状态机对告警关联处理流程的耦合，提高并发处理能力。并发处理能力是通过允许告警接收线程和各状态机告警处理流程同时运行来增强的。

5 实验分析

我们以某省移动公司的网管系统数据库中近一个月的告警数据为基础，首先运用工程人员维护经验以及告警之间的逻辑关系，找出一些具有固定关联的关联规则，包括组合、重复、独立规则在内的总共 87 条规则；然后运用基于 WINEPI(WINdow EPIsode)序列模式挖掘算法[6]，设置最小支持度(支持度反映一个告警序列在所有告警序列中出现的频繁程度)为 20%，最小置信度(置信度反映两个告警序列之间的蕴含强度；强度越大表示关联性越强)为 75%，得出规则数 231 条，再利用人工经验，删除无意义的规则，最后得到规则 142 条。总规则数 87 + 142 = 229 条。

将上述方法得到的规则信息输入到实验数据库中，并通过将一个月以来的告警数据导出到本地文件。生成模拟工具，该工具完成从告警文件中顺序读取告警，并以与实际告警产生时间相同的时间间隔(实验间隔为 30 天)，依次输入到告警关联系统。观察故障输出情况是否与现网近一个月来所产生的故障在数量以及类型上大体一致。表 1 通过测试时长、输入告警量、平均故障输出延迟时间、实际故障数、输出故障数等字段来反应实验情况。测试时长为从测试时间点开始，到记录测试数据时间点的时间长度，以天为单位；输入告警量是在测试时长内输入到系统中的告警总数量；平均故障输出延迟时间通过对本系统的每一个故障(排除误报的故障)的系统输出时间减去实际产生时间以及时间间隔的结果求和，然后除以故障总数得到，以秒为单位，用公式表示为：

$$\frac{\sum_{i=1}^{n}(soti - roti - \Delta t)}{n} \quad (1)$$

公式(1)中 n 表示故障数量；sot_i 表示第 i 个故障的系统输出时间，化成从 1970 - 00 - 00 00:00:00 开始以来的秒数；roti 表示第 i 个故障的实际输出时间，同样化成从 1970 - 00 - 00 00:00:00 开始以来的秒数；Δt，表示实验时间与实际的告警产生时间之间的间隔，是固定值，此实验里为 30 * 24 * 60 * 60 = 2764800 秒。

通过检查平均故障输出延迟时间可以确定故障是否更快捷地定位并进行了输出；实际故障数为在当前的输入告警量情况下，现网实际产生的故障；输出故障数为当前系统实际输出的故障数，通过比较实际故障数与输出故障数的差额可以初步确定告警关联系统的有效性，差额越小即关联有效性越高。

表 1　模拟告警输出故障与实际产生故障在数量、时延上的对比表

测试时长(天)	输入告警量	平均故障输出延迟时间(s)	实际故障数 RFN	输出故障数 OFN	OFN - RFN
1	8475	4.35	4	4	0
5	49357	4.28	19	20	1
10	100781	4.36	35	38	3
30	283467	4.30	108	117	9

从表 1 可以看出，平均故障输出延迟时间不随输入告警量的多少而改变，但随着告警输入量的增大，OFN - RFN 在逐步递增，原因是我们采用了 ACE_Task 并行策略，每一个告警关联规则的告警处理流程相互独立，当有包含规则发生时，比如

alarminfo1 = (1237, "与前台连接断开", ANY)

alarmpair1 = (alarminfo1, 1)

alarminfo2 = (1234, "主控进程重启", ANY)

alarmpair2 = (alarminfo2, 1)

alarminfo3 = (1012, "数据库服务器停止工作", ANY)

alarmpair3 = (alarminfo3, 1)

rule1 = {alarmpair1&alarmpair2&alarmpair3, NONE, 50, "数据库宕机"}

rule2 = {alarmpair2, NONE, 50, "主控进程重启"}

那么当发生告警序列 1233、1237、1234、1012 时同时会输出"数据库宕机"和"主控进程重启"两个故障，而实际只有"数据库宕机"一个故障，因此会出现输出故障数比实际故障数多的情况，这种情况会随着故障数量的增大而有更大的发生几率，也就体现出表 1 中数据反应出的现象。

6 结束语

本文根据告警关联的需要，引入 FSM 到告警关联处理流程中，提出了基于 ACE_Task 主动对象模式进行告警关联的并发处理，设计了一个告警关联系统，该系统在实际的网络环境中已得到应用，正确地进行了告警关联，有效地减少了冗余告警的发生，能够快速、准确地定位出实际故障，但还是存在上报冗余故障的问题，下一步需要在如何去除冗余故障、增加并行处理线程的通信等方面进行重点考虑。

参考文献

1 Syyid U 著，马维达译，自适配通信环境中文技术文档[EB/OL]，http://www.flyingdonkey.com/ace/，2003-6.

2 Schmidt, D C. Tasks and Active Objects: Patterns for Concurrent Programming [M]. Pattern Languages of Program Design, MA: Addison-Wesley, 1995.

3 Douglas C Schmidt, Stephen D Huston 著，马维达译，C++网络编程，卷 2：基于 ACE 和框架的系统化复用[M]，北京：电子工业出版社，2004-2.

4 唐勇、张欣、周明天，一种基于 FSM 的告警事件关联方法，计算机应用研究，2006.9. 23(9):243-246.

5 Alan Shalloway, James R. Trott 著，熊节译，设计模式精解，机械工业出版社，2006.1.

6 李永忠、孙彦、罗军生，WINEPI 挖掘算法在入侵检测中的应用，计算机工程，2006.12. 32(23):159-161.