# 自主测试优先级软硬协同调度策略

郑 伟 孟晓风 张卫军

(北京航空航天大学 仪器科学与光电工程学院, 北京 100191)

**摘 要**: 自主测试是构建航天器自治体系的关键技术之一. 针对实时操作系统自主测试优先级调度问题, 提出一种软硬协同混合调度策略. 通过分析复杂工程背景下影响自主测试任务重要性的多种因素, 提出重要性综合评价指标实现优先级静态分配, 利用自组织特征映射网实现测试任务集聚类. 在此基础上, 建立仲裁代理机制进行优先级再分配, 以可信性为调度判据实现基于现场可编程门阵列 (FPGA, Field Programmable Gate Array)的优先级动态调度. 实验表明该策略对自治系统合理分配优先级资源、高效调度测试任务具有参考意义.

**关 键 词**: 自治系统; 测试; 调度

**中图分类号**: TP 315; V 448.25

**文献标识码**: A **文章编号**: 1001-5965(2008)07-0759-04

# Software-hardware cooperate priority scheduling strategy for autonomous test

Zheng Wei Meng Xiaofeng Zhang Weijun

(School of Instrument Science and Optoelectronics Engineering, Beijing University of Aeronautics and Astronautics, Beijing 100191, China)

**Abstract**: Autonomous test is one of the key technologies for building spacecraft autonomy. For the problem of task priority scheduling in real time operation system, a mix scheduling strategy with software-hardware cooperate was presented. After analyzing the influence factors of task importance in the complex engineering background, a synthesizing estimate index of task importance was proposed to assign priorities statically. The huge task set was clustered by self-organizing feature map afterwards. Based on this work, a priority arbitrage agent was established to reassign priorities. Then the reliability criterion was introduced to realize the priority dynamical scheduling based on FPGA (field programmable gate array). The experiment shows the strategy has reference value for autonomous system to assign priority resource rationally and to schedule test task in high speed.

**Key words**: autonomous agents; testing; scheduling

上世纪末, 美国宇航局新千年计划将航天器自主技术列为重点发展技术[1], 其中, 自主测试是构建航天器自主体系的支撑技术之一. 作为嵌入系统集群体, 航天器引入嵌入式实时操作系统调度多任务运行已成为应对复杂空间任务的重要措施, 将自主测试任务纳入多任务机制是实现在线自主测试的一种有效方法. 目前大部分实时系统采用基于优先级的调度方法, 优先级高的任务先予执行. 优先级调度算法分静态调度和动态调度2类. 静态调度是在系统运行前确定任务优先级, 常见的有速率单调调度[2], 截止时间单调调度[3]等; 动态调度是在系统运行过程中确定任务优先级, 如最早截止时间优先算法[4]和最短空闲时间优先算法[5]等. 现有调度策略在应用于自主测试时有如下问题: ①由于测试背景的复杂性, 自主测试任务难以建立明确的量化评价标准, 单一指标或若干指标简单组合的调度策略不适宜自主测试; ②现有策略主要基于周期任务, 不能移植用于具有异步非周期特点的自主测试任务; ③实时操作系统通常支持有限的优先级, 不能满足测试

收稿日期: 2007-06-06

作者简介: 郑 伟(1975-), 男, 重庆人, 博士生, zw3475@163.com.

任务集可能高达几百个的复杂系统需求; ④现有策略主要是软件行为, 调度过程本身也将占用系统资源, 不适合某些任务切换时间粒度要求在微秒级的强实时系统. 针对上述问题, 本文利用软件灵活性与硬件高速性, 将静态调度与动态调度结合, 提出自主测试优先级软硬协同调度策略, 为复杂自治系统的自主测试提供了一种合理高效的任务优先级调度方法.

## 1 协同调度技术架构

协同调度由静态调度模型和动态调度模型两部分组成. 静态调度模型按任务重要性综合评价指标为自主测试任务集分配优先级, 同时对任务集进行自组织特征映射聚类解决任务众多但优先级个数有限的问题; 动态调度模型采用优先级仲裁代理机制, 实现静态调度基础上的优先级代理分配, 同时, 在任务子集内部利用现场可编程门阵列 ( FPGA, Field Programmable Gate Array) 高速性实现可信性动态调度. 协同调度技术架构见图 1.

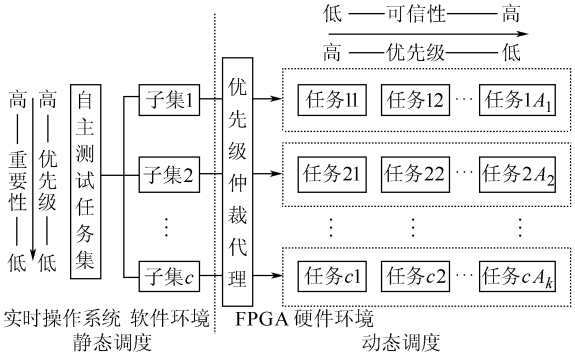



图 1 软硬协同调度技术架构图

## 2 综合评价静态调度模型

### 2.1 自主测试任务重要性影响因素

对一个复杂嵌入系统而言, 影响自主测试任务重要性的因素很多, 本文考虑如下因素:

1) 故障出现时对系统影响程度. 影响越大, 说明针对该故障执行的自主测试任务重要性越高. 设参数最高为 1, 最低为 0.1 小数.

2) 测试时间. 测试时间越短, 意味着测试时效越高, 相应赋予重要性越高. 设参数最长为 13 s, 最短为 0.03 s, 小数.

3) 故障可修复难易. 如果测试任务对应的故障越难修复, 说明该测试应优先执行以尽快开展修复工作. 设最难为 10, 最容易为 0.1 小数.

4) 测试对象有无交叉调用. 测试对象集合其它元素对其调用越多, 测试任务重要性越高. 设最多为 10, 最少为 0.1 整数.

### 2.2 自组织特征映射网

自组织特征映射网 ( SOFM, Self Organizing Feature Map) 是进行数据聚类的一种基于竞争的神经网络[6]. 它由输入层和竞争层构成, 输入层神经元和竞争层神经元均有权值连接, 竞争层节点相互间也可能有局部连接. 结构示意图见图 2

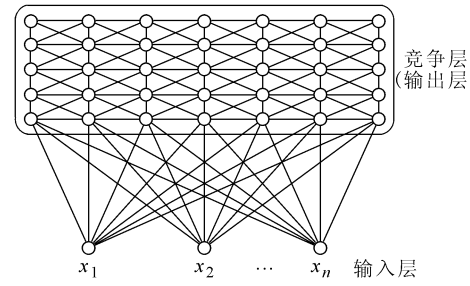



图 2 SOFM 结构示意图

SOFM 可进行无导师聚类, 能将任意维输入模式在输出层映射成一维或二维离散图形, 并保持其拓扑结构不变. 考虑到自主测试任务集合具有数目庞大、特征复杂、容量不确定等特点, 本文选用 SOFM 网络对其进行特征聚类.

### 2.3 综合评价静态调度模型

建立复杂工程背景下的自主测试优先级静态调度模型. 步骤如下:

1) 建立综合评价指标矩阵, 对有 $k$ 个评价任务, $n$ 个评价指标的自主测试集而言, 其综合评价指标矩阵为

$$\boldsymbol{J}_s = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{k1} & a_{k2} & \cdots & a_{kn} \end{bmatrix}$$

其中, $a_{ij} = f_j(x_i)$, 表示第 $i$ 个评价任务 $x_i$ 的第 $j$ 个评价指标值, $i = 1, 2, \cdots, k$; $j = 1, 2, \cdots, n$.

2) 上述评价项目属于不同领域范畴, 具有不同量化单位, 对 $\boldsymbol{J}_s$ 进行归一化得到规范阵 $\boldsymbol{R}$, 处理如下 ( $\vee$, $\wedge$ 分别为模糊取大、取小算子 ):

指标值和重要性同向变化, 相对指标值为

$$r_{ij} = \frac{a_{ij}}{\bigvee\limits_i a_{ij} + \bigwedge\limits_i a_{ij}} \tag{1}$$

指标值和重要性逆向变化, 相对指标值为

$$r_{ij} = \frac{1/a_{ij}}{\bigvee\limits_i (1/a_{ij}) + \bigwedge\limits_i (1/a_{ij})} \tag{2}$$

$a_{ij} = 0$ 时, $a_{ij} = a_{ij} + \sigma$, $\sigma$ 为小常量, 例如 0.01.

3) 设指标 $j$ 对任务的贡献度为 $w_j$ ( $j = 1, 2, \cdots, n$ ), 该贡献度由专家系统评分, 设满分为 10,

综合指标表示为

$$\left.\begin{aligned} g_i(r_{ij}, w_j) &= \sum_{j=1}^{n} r_{ij}\Delta w_j \\ \Delta w_j &= w_j \Big\backslash \sum_{j=1}^{n} w_j \end{aligned}\right\} \quad (3)$$

4) 设系统分配给自主测试的优先级个数为 $m$, 对规范阵 $\boldsymbol{R}$ 进行 SOFM 聚类, 算法如下:

① 设置连接权向量: $\boldsymbol{W}_j = (w_{j1}, w_{j2}, \cdots, w_{jn})$, $j = 1, 2, \cdots, m$, 初始化赋予较小权值. $t$为学习次数, 初始 $t = 0$;

② 输入模式: $\boldsymbol{R}_i = (r_{i1}, r_{i2}, \cdots, r_{in})$, $i = 1, 2, \cdots, k$;

③ 计算欧式距离:

$$d_j = \| \boldsymbol{R}_i, \boldsymbol{W}_j \| = \sqrt{\sum_{p=1}^{n} (r_{ip} - w_{jp})^2} \quad (4)$$

取 $d_{j^*} = \min_j(d_j)$, $d_{j^*}$ 对应的神经元 $j^*$ 为获胜单元;

④调整 $j^*$ 及其邻域 $S_{j^*}$ 神经元权值:

$$w_{jp}(t+1) = w_{jp}(t) + \eta(t)[r_{ip}(t) - w_{jp}(t)] \quad (5)$$

$\eta(t)$为随时间下降的增益项, $0 < \eta(t) < 1$;

⑤更新 $\eta(t)$及邻域 $S_{j^*}(t)$, 取

$$\eta(t) = \eta(0)\left(1 - \frac{t}{T}\right) \quad (6)$$

$$S_{j^*}(t) = \mathrm{INT}\left[S_{j^*}(0)\left(1 - \frac{t}{T}\right)\right] \quad (7)$$

其中, $T$为最大学习次数; $\mathrm{INT}(x)$为取整符号;

⑥ $t = t + 1$, 重复上述过程, 直到 $t = T$.

5) 计算 $m$ 个优先级聚类中综合指标均值, 按均值大小排列任务子集优先级高低.

## 3　可信性动态调度模型

静态调度的结果使各个聚类集合包含多个测试子任务. 尽管这些子任务由于共享优先级而处于同一等级, 但子任务也存在对优先级分配权的争夺问题. 另一方面, 如果一个测试任务受故障触发后执行, 测试一次, 该测试对象的可信度应增加, 在同一任务聚类中发生优先级争抢时, 可信度高的任务应滞后执行. 受上述思路指导, 进一步提出可信性动态调度策略.

### 3 1　优先级仲裁代理

在嵌入系统的接口总线上, 设计优先级仲裁代理电路, 原理框图见图 3 该电路通过将测试优先级仲裁权下放到若干动态调度模块, 在系统底层硬件实现优先级扩展. 工作原理如下:

1) 设静态调度划分的 $c$个测试任务集为 $\{T_{11}, T_{12}, \cdots, T_{1A_1}\}$, $\{T_{21}, T_{22}, \cdots, T_{2A_2}\}$, $\cdots$, $\{T_{c1}, T_{c2}, \cdots, T_{cA_c}\}$, 各个任务集对应一组故障中断向量, 系统工作时各组中断向量状态被实时扫描进 $c$个动态调度模块;

2) 动态调度模块对中断向量进行处理, 处理结果锁存输出到一个 $C \times 1$多通道或门;

3) 多通道或门的输出测试申请信号 TRQ 上传主机后获取测试应答信号 TCK;

4) TCK 信号发出, 各个受操作系统调度运转的测试程序按既定地址读取锁存在或门输入端的数据线值, 确定是否执行自己管辖的任务集. 随后, 测试程序按既定地址读取数据线, 获取子任务向量编码, 确定需要执行的测试子任务.

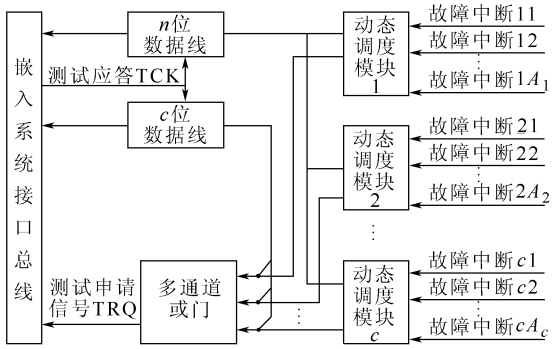



图 3　优先级仲裁代理电路原理图

### 3 2　可信性动态调度

动态调度模块 5个子模块构成, 见图 4. 工作原理如下:

1) 输入缓冲子模块控制中断输入流动速度;

2) 排序调整子模块, 根据上次可信性指标输出结果调整输出中断的优先次序, 可信性指标越低的中断其优先级越高;

3) 按排序结果将优先级最高的中断输出到串行输出子模块, 同时将该中断在向量中的位置编码输出到编码输出子模块以供辨别是哪路中断;

4) 可信性计算子模块为每路中断提供可信性计数器, 初始计数值设为 0, 每当有向量编码输出时该编码对应的中断可信指标加 1, 表示经过一次测试后该测试对象可信赖度增加;

5) 最新可信指标计算结果输出到排序调整子模块作为下一轮排序判据.

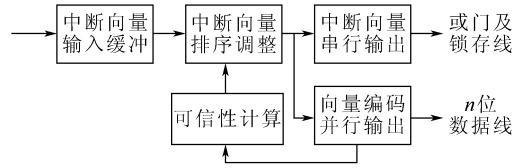



图 4　可信性动态调度模块组织框图

## 4 实 验

按 2.1节所述, 设自主测试重要性指标为系统影响度、测试时间、修复难度、交叉调用 4类, 自主测试任务共计 12个, 综合评价指标见表 1.

表 1 自主测试任务综合指标评价表

| 任务 | 系统影响度/级 | 测试时间/s | 修复难度/级 | 交叉调用/个 |
|---|---|---|---|---|
| 1 | 0.44 | 11.40 | 6.4 | 1 |
| 2 | 0.56 | 3.35 | 5.6 | 2 |
| 3 | 0.75 | 3.32 | 2.7 | 1 |
| 4 | 0.32 | 8.45 | 4.8 | 2 |
| 5 | 0.20 | 2.34 | 1.9 | 3 |
| 6 | 0.57 | 0.87 | 4.4 | 6 |
| 7 | 0.34 | 2.32 | 6.9 | 2 |
| 8 | 0.83 | 6.23 | 3.8 | 3 |
| 9 | 0.46 | 7.27 | 5.5 | 5 |
| 10 | 0.35 | 2.14 | 0.4 | 2 |
| 11 | 0.54 | 0.58 | 3.6 | 7 |
| 12 | 0.25 | 0.28 | 2.8 | 2 |

按表 1数据计算规范阵 $\boldsymbol{R}$, 对 $\boldsymbol{R}$ 进行 SOFM 聚类, 表 2是训练 10, 100, 600步的聚类结果.

表 2 SOFM 不同迭代次数聚类结果

| 任务 | 迭代次数 | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| 10步 | 1 | 4 | 2 | 4 | 6 | 2 | 5 | 2 | 6 | 2 | 2 | 6 |
| 100步 | 1 | 1 | 3 | 1 | 5 | 4 | 1 | 2 | 2 | 5 | 4 | 6 |
| 600步 | 4 | 4 | 6 | 4 | 5 | 1 | 4 | 6 | 2 | 5 | 1 | 1 |

根据 SOFM 算法原理, 训练步数越高, 聚类时对特征的考虑越细致. 训练 600次后任务集被分为 5类: {1, 2, 4, 7}, {3, 8}, {9}, {6, 11, 12}, {5, 10}. 比较表 1原始数据可知, 上述分类符合任务特征指标.

上述过程在通用 PC机上完成. 按照聚类结果, 利用 FPGA硬件实现优先级仲裁代理及动态调度. 以 5个中断输入的动态调度模块为例, 从 ModelSim 环境截取 250~400 ns的时域数据向量(二进制)见表 3.

由表 3可知, 系统初始化时 5个中断优先级次序由高到低为中断 0-1-2-3-4, 可信指标均为 0. 在 300 ns发生中断 0和中断 3, 中断 0由于优先级最高先输出, 其可信指标增 1, 随后中断 0优先级降为最低, 其余中断优先级依次升高 1级. 在 350 ns系统再次捕捉中断输入源变化时, 中断 3本应输出, 但由于发生了中断 1, 其优先级比中断 3高, 故中断 1抢占了输出, 其可信指标增 1. 系统根据新一轮可信指标计算结果对中断优先级重新排序, 在下一轮输出中断 3. 动态调度由于是 FPGA 硬件实现, 每次轮转时间开销约为几十纳秒, 相对微秒级以上的软件调度方案而言更适用于强实时系统.

表 3 5个中断输入的动态调度模块时域数据表

| 项目 | 时间/ns | | | |
|---|---|---|---|---|
| | 250 | 300 | 350 | 400 |
| 中断输入 | 00000 | 01001 | 01011 | 01001 |
| 中断串行输出 | 0 | 1 | 1 | 1 |
| 中断编码输出 | 111 | 000 | 001 | 011 |
| 任务 4优先级 | 100 | 011 | 010 | 001 |
| 任务 3优先级 | 011 | 010 | 001 | 100 |
| 任务 2优先级 | 010 | 001 | 000 | 000 |
| 任务 1优先级 | 001 | 000 | 100 | 011 |
| 任务 0优先级 | 000 | 100 | 011 | 010 |
| 任务 4可信指标 | 0000 | 0000 | 0000 | 0000 |
| 任务 3可信指标 | 0000 | 0000 | 0000 | 0001 |
| 任务 2可信指标 | 0000 | 0000 | 0000 | 0000 |
| 任务 1可信指标 | 0000 | 0000 | 0001 | 0001 |
| 任务 0可信指标 | 0000 | 0001 | 0001 | 0001 |

## 5 结 束 语

本文结合自主测试任务特点, 就实时操作系统自主测试优先级调度问题提出了一种软硬协同调度策略. 其中, 综合评价静态调度模型基于软件实现, 可信性动态调度模型基于硬件实现, 两者通过优先级仲裁代理机制实现互补协同. 自主测试是一个复杂的工程课题, 本文策略在综合考虑测试任务复杂背景的前提下实现了对庞大测试任务集的高速调度, 为构建自治智能系统的自主测试体系提供了技术支持.

## 参考文献 (References)


[1] Hdberg S. AI coming of age: NASA uses AI for autonomous space exploration[J]. IEEE Expert, 1997, 12(3): 13-15

[2] Naghibzadeh M, Kim K H. A modified version of rate-monotonic scheduling algorithm and its efficiency assessment[C]//Proceeding of the Seventh IEEE International Workshop on Object-Oriented Real-time Dependable System. San Diego: IEEE, 2002: 289-294

[3] Hongryeol K, Joom in K, Daewon K. Development of coordinated scheduling strategy with end-to-end response time analysis for CAN-based distributed control systems[C]//IEEE/RSJ International Conference on Intelligent Robots and Systems. Sendai: IEEE, 2004, 3: 2099-2104

[4] Quan Z, Chung J M. A statistical framework for EDF scheduling [J]. IEEE Communications Letters, 2003, 7(10): 493-495

[5] Oh S H, Yang S M. A modified least-laxity-first scheduling algorithm for real-time tasks[C]//Fifth International Conference on Real-Time Computing Systems and Applications. Hiroshima: IEEE, 1998: 31-36

[6] Kohonen T. Self-organizing neural projections[J]. Neural Networks, 2006, 19(6): 723-733


(责任编辑: 吴 琼)