# 面向 QoS 保障的虚拟可信服务 *

杨晓宁, 杨志义, 隋玉磊, 杨 刚
(西北工业大学 计算机学院, 西安 710129)

**摘 要**: 针对服务计算模型中对应用的可信特性支持不足问题,提出面向 QoS 保障的虚拟可信服务 VTS,并依据 VTS 设计了一种基于反馈控制的自适应 QoS 保障机制。首先将 VTS 的 QoS 保障转换为反馈控制问题,给出一种自适应 QoS 保障框架,再在此框架下对 VTS 的 QoS 维护过程和策略进行建模,设计和实现了相应的动态组建、调节算法和实时评估策略。最后通过仿真实验的结果分析表明,该保障机制能够有效地增强 VTS 对服务实体运行时的 QoS 保障能力。

**关键词**: 可信计算; QoS 保障; 虚拟可信服务
**中图分类号**: TP393 **文献标志码**: A **文章编号**: 1001-3695(2010)05-1823-03
doi:10.3969/j.issn.1001-3695.2010.05.061

# Virtual trustworthy service for QoS guarantee

YANG Xiao-ning, YANG Zhi-yi, SUI Yu-lei, YANG Gang
(*School of Computer Science, Northwestern Polytechnical University, Xi'an* 710129, *China*)

**Abstract**: Currently, service computing paradigm lacks the support of Trustworthiness. This paper proposed a new methodology virtual trustworthy service (VTS) to guarantee the QoS of service. Firstly, presented an adaptive feedback control framework based on VTS to guarantee the QoS of service. Secondly, under this VTS based QoS guarantee framework, made several QoS maintain schemes: dynamic establishment algorithm, real-time assessment algorithm and dynamic adjustment algorithm. Finally through analysis of the simulation experiment, the results demonstrate that the VTS based adaptive QoS guarantee Framework can effectively guarantee the QoS of service, and provide run-time QoS guarantee support.

**Key words**: trustworthy computing; QoS guarantee; virtual trustworthy service

服务的分布性、异构、自治和动态变化等特点,引发了服务计算环境的松散化、独立性、异构性等多种不确定因素,从而对服务提供者的非功能属性,即服务质量(quality of service, QoS)提出更严格的要求[1]。QoS 通常包括实时性、安全性、可靠性、可用性等特性,用于衡量使用一个服务的满意程度,是面向服务的应用能否成功的关键因素之一。随着软件系统复杂度和人们对软件系统非功能属性要求的不断提高,保障服务计算系统的可信性成为平台设计者和服务应用开发者所面临的重要问题。另外,现实的服务计算环境中,应用往往由多个原子服务按照一定的逻辑关系组成,组合服务的可信性依赖于原子服务的可信度,所以对原子服务的 QoS 保障展开研究是十分有必要的。为了提高 Web 服务的可信性,已经有一些研究成果和方法被提出,如文献[2]介绍了一种 Web 服务中间件,目的在于提高 Web 服务消息传输的可靠性;文献[3]提出了一种基于网格架构的面向服务的中间件(SoM)支持高效的 QoS 管理,从而保障和满足用户对应用的需求;文献[4]设计并实现了一个 Web 服务框架,通过进行服务实体的多份拷贝,保障服务的可用性;文献[5]提出了一种冗余服务选择机制的服务组合方法来提高系统的可用性;文献[6]将反馈控制的思想应用到服务计算,提出了一种基于反馈中间件的可靠面向服务架构。然而,以上这些成果大多将重点停留在可靠性和安全性这两个方面,并且缺少对服务实体运行时的 QoS 监控,从而使之前的保障效果大打折扣。

针对上述问题,本文提出一种基于 VTS 的自适应 QoS 保障框架[7],将 QoS 保障问题转换为反馈控制问题,贡献如下:

a)对服务实体在一段时间内 QoS 的变化和 VTS 的适应性调整过程进行建模,以期寻找策略支持组合服务的长期可信运行。

b)设计相应算法,实现动态监测和动态评估特定服务资源的质量变化情况,使用获取到的当前 QoS 执行信息,通过信息融合计算判断最初协定的 QoS 需求是否正得到满足。

c)实现 VTS 的状态自适应调节,以适应服务资源、应用需求的动态变化,最终完成整个 QoS 动态保障机制。

## 1 自适应 QoS 保障系统

在网络环境中,由于服务器节点负载变化、网络通信环境不稳定、应用服务器失效或服务器节点崩溃等因素,服务实体在运行时的 QoS 能力是不断变化的,这就意味着其最终有可能无法按照既定 QoS 约束成功执行。为了满足原子服务的 QoS 约束、保障组合服务的整体 QoS 需求,VTS 必须具备自适应能力,对其状态进行实时的动态调节。

**收稿日期**: 2009-09-14; **修回日期**: 2009-10-30 **基金项目**: 国家"863"计划资助项目(2006AA01Z162);国家××预研基金资助项目(06004089);西北工业大学校科技创新基金资助项目(2008KJ02030)

**作者简介**:杨晓宁(1985-),女,陕西蒲城人,硕士研究生,主要研究方向为分布式与可信计算(yangxiaoning_1985@sina.com);杨志义(1952-),男,陕西蒲城人,教授,主要研究方向为嵌入式软件环境仿真;隋玉磊(1987-),男,河南洛阳人,硕士研究生,主要研究方向为分布式与可信计算;杨刚(1974-),男,陕西澄城人,讲师,主要研究方向为分布式与可信计算.

### 1.1 系统模型

#### 1.1.1 形式化定义

为了方便系统建模和算法设计,本文提出几个形式化定义:

**定义 1** 多维可信 QoS。使用三元组 $q(t,\theta,r)$ 来表示本文要着重研究的可信 QoS 的两个维度,即服务的实时性和可靠性。$t(t>0)$ 表示服务平均响应时间,含义为服务调用者从发出服务调用请求到正确接收服务成功执行返回的结果所需要的时间。可靠性被描述为一段时间 θ 内服务成功执行次数与服务收到的调用请求次数之比率 $r(0<r\leqslant 1)$。

**定义 2** 服务实体。服务实体被定义为三元组 $SE(f,d,Q)$。其中 $f$ 表示该服务实体所提供的原子服务功能,$d$ 表示该服务实体的访问方式(如访问地址、调用规则等),而 $Q(q_{best},q_{worst},q,q_\theta)$ 则描述了该服务实体可信 QoS 水平的最优值 $q_{best}$、最差值 $q_{worst}$、根据历史统计数据计算的初始值 $q$ 和根据最近 θ 时间内的实时数据计算的当前实际值 $q_\theta$。

**定义 3** VTS。它是指通过对服务实体资源进行有效管理从而为外界提供特定原子服务功能的、满足一定可信 QoS 约束的虚拟服务实体。VTS 本身不提供任何服务功能,实际功能提供者是具体的服务实体。当接收到一个服务调用请求后,VTS 就把该请求转交给其管理的某个具体服务实体处理,若不能成功执行,则转交给另一个,直至服务请求被成功执行或向服务调用者返回失败信息。

VTS 被定义为多元组 $VTS(f,d;q_r;\{SE_i\}_i,SPS)$。其中 $f$ 和 $d$ 的含义与服务实体定义中的相同,$q_r$ 是该 VTS 的额定 QoS 约束,$\{SE_i\}_i$ 指由该 VTS 管理的提供相同原子服务功能 $f$ 的服务实体集合,SPS 是该 VTS 的服务调度策略集,包含两方面内容:一是 VTS 的服务请求分配策略,二是当 VTS 在运行态中不满足既定的 QoS 约束时采用的动态调整策略。

#### 1.1.2 自适应 QoS 保障系统框架

本文将服务的 QoS 保障转换为自适应控制问题,VTS 管理的服务实体及其服务资源作为被控对象,服务的 QoS 保障策略作为可调节控制器,服务实体、保障策略和监测机制构成一个自适应反馈控制系统,如图 1 所示。其中,$q_r$ 是给定的 QoS 约束值,代表对 VTS 可信属性的期望;$q_\theta$ 是系统的反馈值,即监测器通过监控服务运行和服务资源,根据最近 θ 时间内的实时监测数据计算的当前实际 QoS 值;$q_e$ 是 VTS 控制器根据期望值和实际值的差确定的控制量,表示需要对 VTS 的可信系数进行调整的值。

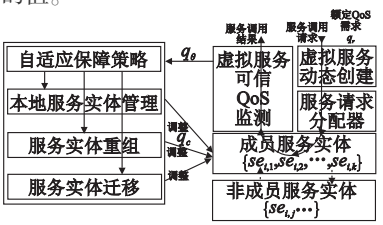



**图1 自适应QoS保障系统框架**

可信 QoS 监测组件实时测量并记录每个服务调用请求的执行状况,自适应保障策略模块周期性地向可信 QoS 监测组件查询最近 θ 时间内 VTS 的可信 QoS 水平。通过与额定可信 QoS 指标比较,保障策略模块判断 VTS 的可信 QoS 约束是否正按照计划得到满足。若发现当前可信 QoS 指标与计划偏离,保障策略模块会及时通过服务节点本地服务管理、服务重组、服务迁移等措施对 VTS 运行状态进行调整,这构成了整个自适应 QoS 保障系统的反馈控制环。

### 1.2 VTS 的动态创建

#### 1.2.1 VTS 的可信 QoS 映射

在服务实体集合 $\{SE_1,SE_2,\cdots,SE_n\}$ 确定的前提下,整个 VTS 的可信 QoS 水平与其分配服务调用请求的策略和各服务实体的可信 QoS 指标有关,这种映射关系可以形式化地表示为

$$q_{VTS}(t_{VTS};\theta,r_{VTS})=\Psi(\{SE_1,SE_2,\cdots,SE_n\},RDP)=\Psi(\{q_1,q_2,\cdots,q_n\},RDP)$$

其中:$q_{VTS}$ 表示 VTS 按照历史统计数据计算的当前可信 QoS 水平;RDP 指 VTS 的服务请求分配策略,不同服务请求分配策略下,Ψ 具有不同的形式。以本文采用的等概率服务请求分配策略(按照相同的概率分配服务请求)为例,设服务实体 $SE_i$ 获得服务请求的概率为 $p_i$,因为 $\sum_{i=1}^{n}p_i=1$,所以 $p_1=p_2=\cdots=p_n=\frac{1}{n}$,则 VTS 的当前可信 QoS 指标与服务实体可信 QoS 指标的计算关系如下:

$$q_{VTS}=\Psi(\{q_1,q_2,\cdots,q_n\},RDP_{ep})=\Psi_{ep}(q_1,q_2,\cdots,q_n)=\left(\sum_{i=1}^{n}p_i\cdot t_i;\theta,1-\prod_{i=1}^{n}(1-r_i)\right)=\left(\frac{1}{n}\sum_{i=1}^{n}t_i;\theta,1-\prod_{i=1}^{n}(1-r_i)\right)$$

#### 1.2.2 动态创建算法

VTS 的创建算法从一个给定的服务实体集 $\{SE_1,SE_2,\cdots,SE_n\}$ 中取其一个子集 $V$,使得由 $V$ 所组成的 VTS 的可信 QoS 指标 $q_{VST}$ 能够满足约束 $q$,即 $t_{VTS}=\frac{1}{n}\sum_{i=1}^{n}t_i\leqslant t_r$,且 $r_{VTS}=1-\prod_{i=1}^{n}(1-r_i)\geqslant r_r$。

本文设计了一个基于回溯法的带有两个剪枝函数的 create_VTS算法,其基本思想是:将所有的服务实体按照 $t$ 值由小到大排序,并自顶向下以服务实体作为节点,以选或不选作两条边,组成解空间二叉树。按深度优先遍历解空间树,将不满足剪枝条件的子树剪掉,就可以确定所有满足条件的结果并输出。两个剪枝函数设计如下:

a)按时间约束剪枝。排序后,每增加一个实体成员,$t_{VTS}$ 呈递增趋势,所以当走选中该节点的边时,若新生成的 $t_{VTS}>t_r$,这个子树就要被剪掉。

b)按可靠性约束剪枝。每增加一个实体成员,$r_{VTS}$ 也呈递增趋势,所以如果走不选该节点的边,且当前的成员和该节点后面的所有服务实体组成的 VTS 的可靠性 $r=1-(1-r_{now})(1-r_{rest})<r_r$,此子树也要被剪掉;

**算法 1** (create_VTS)

1. 将 $\{SE_1,SE_2,\cdots,SE_N\}$ 按照 $t$(响应时间)值由小到大排序;

```
void create_VTS (int i){
    2. 自顶向下,按深度优先遍历解空间树,if(到达树叶)
    输出结果;
    3. se[i]=1;/*数组 se[N]用于存储本算法的结果,0 表示相应服务实体未被选中,1 表示选中*/
    if(!(T(i)>t_r))
        create_VTS(i+1);
/*剪枝条件一:如果当前结果的平均时间 T(i)>t_r,则剪枝,否则继续解空间树的下一层搜索*/
    4. se[i]=0;
    if(!(1-(1-R(i))*(1-R(剩余服务实体))<r_r)
        create_VTS(i+1);
/*剪枝条件二:如果当前结果中所选中服务实体加上所有剩余的服务实体的可靠性仍小于 r_r,则剪枝,否则继续解空间树的下一层搜索*/
}
```

### 1.3　VTS 的可信 QoS 评估与调节

#### 1.3.1　实时评估方法

系统监测机制需要与服务的中间状态进行交互,获取其执行进度,以此来评估当前 VTS 的 QoS 状况。但一直以来,Web service 被认为是无状态的,这给 QoS 评估带来很大的困难[8]。本文借助 Web 服务资源框架(Web service resource framework, WSRF)提出的 WS-Resource 概念将有状态的服务资源和无状态的服务结合起来,使用其提供状态资源模型创建状态资源。创建成功后,返回给服务请求端点如下 XML 文档:

```
〈wsa:EndpointReference〉
〈wsa:Address〉
http://some.com/webservice
〈/wsa:Address〉
〈wsa:ReferenceProperties〉
〈tns:resourceID〉XX〈/tns:resourceID〉
〈/wsa:EndpointReference〉
```

通过对具体的状态资源的访问,实现对有中间状态的 Web service 的执行进度的获取,如果监测机制发现最近 θ 时间内服务状态资源的变化与计划偏离,就对 VTS 进行动态调节。

#### 1.3.2　动态调节算法

成员服务实体重组是 VTS 运行时调节最常采取的措施,它要解决的关键问题是:当 VTS 的 QoS 不满足既定约束时,为了及时恢复 QoS 的供给水平,应将系统当前可用的哪些服务实体选取添加到 VTS 中。

设当前 VTS 的成员集为 $\{SE_1,\cdots,SE_2,\cdots,SE_n\}$,系统当前可用的服务实体集合为 $\{SE_{v_1},SE_{V_2},\cdots,SE_{v_m}\}$,最近 θ 时间内所检测到的实际 QoS 水平表示为 $q_{\mathrm{VTS},\theta}$(其中 $t_{\mathrm{VTS},\theta}>t_r$ 或 $a_{\mathrm{VTS},\theta}<a_r$)。由于 VTS 的可靠性会随着成员服务实体的增添而提高,添加服务实体成员时优先考虑满足服务响应时间要求,该算法的基本思想是:从 $\{SE_{v_1},SE_{v_2},\cdots,SE_{v_m}\}$ 中选取一组能够使 $t_{\mathrm{VTS},\theta}$ 变得小于 $t_r$ 且使 $|t_r-t_{\mathrm{VTS},\theta}|$ 值最小的服务实体;然后基于已选中的服务实体再判断 VTS 的可靠性是否也一同得到了满足,如果 VTS 可靠性仍然小于 $r_r$,则继续添加可用的服务实体直至 VTS 的服务响应时间和可靠性约束均得到满足。

**算法 2**　服务实体重组

1. 将 $\{SE_{v_1},SE_{v_2},\cdots,SE_{v_m}\}$ 按照 $t_{v_i,\theta}$ 值由小到大排序,得到 $SES=\{SE_{s_1},SE_{s_2},\cdots,SE_{s_m}\}$;

2. $V_1\leftarrow\{SE_{s_1},\cdots,SE_{s_e}\}$,$V_2\leftarrow\{SE_{s_{e+1}},\cdots,SE_{s_m}\}$,其中 $\forall 1\leqslant k\leqslant e$, $t_{s_k}<t_r$;$\forall(e+1)\leqslant k\leqslant m$,$t_{s_k}\geqslant t_r$;

3. while　$t_{\mathrm{VTS},\theta}>t_r$　do

4. foreach　$SE_k\in V_1$　do

if($t(\{SE_i\}_{i=1}^{n}\cup S\cup\{SE_k\})\leqslant t_r$)　then

/* $t$ = VTS 原有的成员集∪已经添加的服务实体集 $S$∪此刻添加的服务实体所组成的服务组的平均响应时间 */

if($|t-t_r|<$上一个差值)　then　选择该服务实体

endfor

5. 将选中的服务实体加入集合 $s$ 中

6. endwhile

## 2　实验与分析

本实验将配置为 2 GHz Pentium 4CPU、512 BM RAM 的 12 台微机分为三组,第一组不使用任何保障机制;第二组采用经典的备份算法;第三组使用本文提出的保障机制,将若干相同功能的服务实体部署至多个服务节点,从原子服务的可靠性和实时性两个方面着手,监控服务实体的响应时间和每分钟成功执行率,并对三种情况下实际可信 QoS 供给水平作了对比。

实验具体步骤为:实验驱动程序启动后,三组机器连续运行若干小时,在此期间不断发送服务调用请求,并在服务实体运行时监测和记录可信 QoS 执行信息。第一组采用随机选择服务实体的方式运行;第二组随机选择服务实体,并为每个服务实体创建一个备份;第三组收到服务调用请求后,VTS 管理器在服务组管理节点上启动 VTS 创建进程,VTS 创建进程根据实验驱动程序的 QoS 需求创建 VTS 实例,并根据动态调节算法进行运行时的 QoS 保障。在连续 1 h 的实验中,详细记录下了三组服务的响应时间和可靠性数据,如图 2、3 所示。

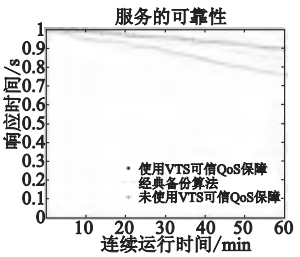



图2　原子服务可靠性对比

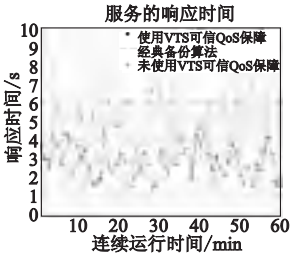



图3　原子服务响应时间对比

在服务运行过程中,节点负载的变化是影响 QoS 主要的因素,从图 2、3 可以看出,未使用 VTS 自适应 QoS 保障机制时,原子服务的平均服务响应时间和可靠性受服务节点负载变化的影响很大。当服务节点负载过高时,平均服务响应时间变得很低,无法满足用户的可信 QoS 需求;反之,虽然服务实体的平均服务响应时间和可靠性仍然受到服务节点负载的影响,但由 VTS 所呈现的平均服务响应时间能及时得到调整,保持平稳,提高了用户对服务 QoS 需求的满足率。

图 4 显示了三种情况下原子服务的 QoS 接纳率,即用户提出的 QoS 要求得到满足的次数与服务调用请求次数的比例。可以看出,在服务运行的开始阶段,三组的用户 QoS 需求接纳率,得到满足的比例都维持在较高的水平,但随着运行时间的增加,其余两组由于没有使用 VTS 保障机制对服务请求进行合理分派,导致个别服务实体响应时间过长,从而降低了整个服务组的 QoS 接纳率。

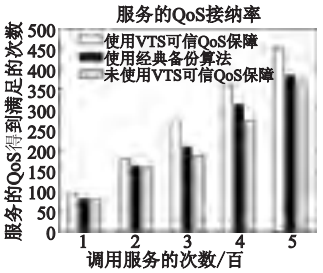


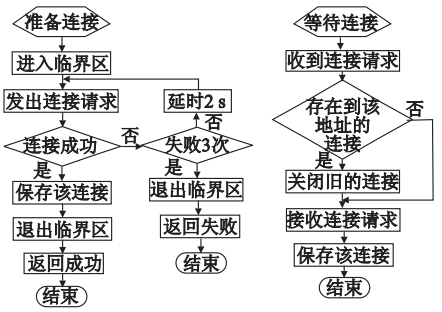

图4　原子服务QoS接纳率对比

## 3　结束语

本文设计并实现了一个保障原子服务 QoS 的自适应反馈框架,通过对 VTS 的动态创建、服务实体运行时的实时监测与评估、VTS 成员服务实体重组等问题的研究,给出了合理有效的算法解决方案,并验证了该模型的有效性及稳定性,完成了依据 VTS 基于反馈的可信 QoS 保障机制的设计与实现。

**参考文献:**

[1]　邵凌霜,李田,赵俊峰,等. 一种可扩展的 Web service QoS 管理框架[J]. 计算机学报,2008, 31(8):1458-1470. (下转第 1829 页)




(a)发送连接的建立流程　　(b)接收连接的建立流程

图9　发送与接收连接建立流程

## 2　实验测试数据与分析

### 2.1　块内存管理封装机制

基于 VxWorks 商用嵌入式操作系统支撑层块内存申请和释放性能测试结果如表 1 所示。从表 1 中的测试数据可以看出,支撑层的内存管理操作相对于直接底层操作系统接口的调用,时间性能上略有降低,但支撑层中增加了内存操作的统计、保护和调试等措施;在面向通信领域的软实时应用中,相比较而言,由此产生的时间性能降低是值得的。

表 1　固定大小的内存申请和释放性能测试　　ms

| | Vxworks get/free | OSS VOS_get/VOS_free |
|---|---|---|
| 5 000 个 64 Byte 的内存 | 9.945/10.408 | 11/14 |
| 2 000 个 8 KB 的内存 | 4.481/4.640 | 4/5.5 |
| 1 000 个 64 Byte 的内存 | 1/1 | 2/3 |
| 500 个 8 KB 的内存 | 0/1 | 1/1 |

### 2.2　二级调度

基于 VxWorks 的一级调度(无次任务的调度)和二级调度切换性能测试结果如表 2 和 3 所示。

从性能测试数据对比上看,对于一级调度,任务的切换时间在 2 ~ 4 μs 之间,且与任务数目有关,任务采用发送消息的方式引起的任务切换时间为 28 ~ 36 μs;对于二级调度,次任务消息引起的次任务切换时间在 7 ~ 11 μs 之间。对于消息传递所引起的次任务和任务切换,二级消息调度方法明显优于一级消息调度。

## 3　结束语

本文针对无线网络控制器(RNC)的应用特点提出了一种操作系统支撑层方案,并对其中的调度、定时器、内存管理、I/O 驱动及任务间通信(IPC)等模块给出了更高效和稳定的封装机制。该方案为无线网络控制器中的 ATM 传输网络子系统、无线信令子系统、数据库子系统和操作维护子系统的软件提供了一个统一的分布式编程平台和运行平台。通过无线网络控制器产品的实际软件开发实践证明,该方案具有较高的应用价值。

表 2　一级调度切换时间　　μs

| 测试序号 | 任务个数 | 总切换次数 | 切换时间(包括信号量) | 实际切换时间(减去信号量) |
|---|---|---|---|---|
| 1 | 2 | 4 000 000 | 2.498 4 | 2.000 2 |
| 2 | 4 | 4 000 000 | 2.500 0 | 2.001 8 |
| 3 | 8 | 8 000 000 | 2.500 2 | 2.002 0 |
| 4 | 16 | 8 000 000 | 2.500 0 | 2.001 8 |
| 5 | 32 | 8 000 000 | 2.746 9 | 2.248 7 |
| 6 | 64 | 8 000 000 | 3.999 8 | 2.501 6 |
| 7 | 80 | 8 000 000 | 4.314 4 | 3.822 4 |
| 8 | 100 | 8 000 000 | 4.498 5 | 4.000 3 |
| 9 | 160 | 8 000 000 | 4.499 0 | 4.000 8 |
| 10 | 200 | 8 000 000 | 4.500 2 | 4.002 0 |

表 3　二级调度测试结果表

| 切换类型 | 测试序号 | 次任务个数 | 切换次数 | 切换时间/μs | |
|---|---|---|---|---|---|
| 不同任务 | 1 | 2 | 1 000 000 | 28.35 | |
| | 2 | 4 | 1 000 000 | 30.54 | |
| | 3 | 8 | 1 000 000 | 31.00 | |
| | 4 | 16 | 1 000 000 | 32.81 | |
| | 5 | 32 | 1 000 000 | 36.04 | |
| | 6 | 63 | 1 000 000 | 36.03 | |
| 同一任务 | 1 | 1 | 8 000 000 | 7.3 | 性能提高倍数 |
| | 2 | 2 | 8 000 000 | 7.5 | 3.78 |
| | 3 | 4 | 8 000 000 | 10.0 | 3.05 |
| | 4 | 8 | 8 000 000 | 10.7 | 2.90 |
| | 5 | 16 | 8 000 000 | 10.8 | 3.04 |
| | 6 | 32 | 8 000 000 | 11.0 | 3.28 |
| | 7 | 64 | 8 000 000 | 11.2 | 3.22 |

**参考文献:**


[1] 何先波,钟乐海,芦冬昕. 嵌入式操作系统支撑层的设计与实现[J]. 计算机应用,2003,23(5):89-91.

[2] 何先波,李志蜀,芦东昕,等. 一种面向通信领域的高效嵌入式操作系统进程队列模型[J]. 哈尔滨工程大学学报, 2006,27(2):263-267.

[3] 刘飞,罗克露,黄烨明,等. 通信领域嵌入式系统调度管理的设计与实现[J]. 计算机应用,2004,24(7):186-188.

[4] 刘飞, 芦东昕, 缪敬. 面向通信领域通用内存管理单元的算法和实现[J]. 计算机工程, 2003,29(22):80-82,105.

[5] 何先波,张芝萍,徐立峰,等. 一种嵌入式实时操作系统中内存分配的方法,中国:CN1722106[P]. 2004.

[6] 钱静,芦东昕,谢鑫,等. 嵌入式软件虚拟内存管理技术的研究和实现[J]. 计算机应用, 2005,25(2):180-182.


(上接第 1825 页)


[2] ERRADI A, MAHESHWARI P. WsBus: QoS-aware middleware for reliable Web services interactions[C]// Proc of IEEE International Conference on eTechnology, E-Commerce and E-Service. 2005.

[3] SHETH A, CARDOSO J, MILLER J, *et al*. QoS for service-oriented middleware [C]// Proc of the 8th International Conference on Information Systems Analysis and Synthesis. 2002.

[4] ALWAGAIT E, GHANDEHARIZADEH S. DeW: a dependable Web services framework[C]// Proc of the 14th International Workshop on Research Issues on Data Engineering Web Services for E-Commerce and E-Government Applications . 2004.

[5] GUO Hui-peng, HUAI Jin-peng, LI Huan, *et al*. ANGEL: optimal configuration for high available service composition[C]// Proc of ICWS. 2007.

[6] HUANG Gang, LIU Xuan-zhe, HONG Mei. SOAR: towards dependable service-oriented architecture via reflective middleware[J]. International Journal of Simulation and Process Modelling, 2007, 3(1/2):55-65.

[7] GUO Hui-peng, HUAI Jin-peng, LI Yang, *et al*. KAF: Kalman filter based adaptive maintenance for dependability of composite services [C]// Proc of CAiSE. 2008.

[8] 张皓. 基于 Web 服务资源框架的在线实验服务设计与实现[D]. 武汉:华中科技大学,2006.