基于多特征动态优先级的网络实时调度算法

苏洵[1]，李艳芳[2]，宗宁[1]，魏巍[3]，李娟[1]，丁莹[1]
（1. 61623 部队，北京 100036；2. 61516 部队，北京 100074；
3. 北京航空航天大学机械工程及自动化学院，北京 100088）

摘 要：针对网络实时调度问题，提出实时调度系统体系结构与任务模型。综合考虑任务截止期、执行时间及间隔时间等属性，定义任务迫切度；根据不同任务的重要程度，提出基于服务质量的任务松紧度。通过迫切度和松紧度对优先级的动态调节，得到防止任务频繁切换的颠簸限度，保证了任务执行成功率与客户端资源利用率。仿真实验结果表明，与调度尽力交付（BE）算法、最早截止时间优先（EDF）算法相比，基于多特征动态优先级的网络实时调度算法提高了任务调度成功率，缩短了平均响应时间。

关键词：多特征动态优先级；迫切度；松紧度；颠簸限度
中图分类号：TP301.6
文献标识码：A
doi: 10.11959/j.issn.1000−436x.2020091

Network real-time scheduling algorithm based on multi-feature dynamic priority

SU Xun[1], LI Yanfang[2], ZONG Ning[1], WEI Wei[3], LI Juan[1], DING Ying[1]
1. 61623 PLA Troops, Beijing 100036, China
2. 61516 PLA Troops, Beijing 100074, China
3. College of Command Automation Beijing University of Aeronautics and Astronautics, Beijing 100088, China

Abstract: Real-time task scheduling system structure and task model were proposed aiming at the network real-time scheduling problem. The task degree of urgency was defined by considering the deadline of task, execution time and interval time between works. The task degree of tightness was proposed based on service-level assurance, according to functional importance of different tasks in the real-time task scheduling system. The thrashing limit for avoiding task switching frequently was acquired through dynamic regulation to task priorities by degree of urgency and degree of tightness, which guaranteed the success rate of tasks execution and utilization ratio of client execution. Test simulation results suggest that the multi-feature dynamic priority scheduling strategy improves the success rate of task scheduling and shorten the average response time, which suggests it has obvious superiority compared with BE and EDF scheduling algorithm.

Key words: multi-feature dynamic priority, degree of urgency, degree of tightness, thrashing limit

1 引言

实时系统是通过任务调度的方式，根据响应而对任务进行处理的系统。其应用已涉及计算机网络系统、互联网、工业控制及航天飞机系统等多个领域。实时调度作为实时系统的重要作业方式，是研究工作的重点内容。随着实时系统应用领域的延伸，任务对调度效率及全面性也有了更高的要求。

实时调度是网络调度中的一个热点课题，它是相对于传统的离线调度而提出的。实时调度决策基于即时的信息，也可以包含历史信息和未来信息。网络实时调度系统的调度指令与网络的状态有关，调度系统根据实时信息做出调度安排，并把调度安排发送给执行单元。

国内外学者对实时调度系统的调度算法相关问题进行了大量研究。Semghouni 等[1]提出了一种

收稿日期：2019−10−31；修回日期：2020−04−14

分组最早截止时间优先算法，通过使用所有任务截止时间的加权均值作为该组的优先级，提高系统处理过载情况的能力。Muhuri 等[2]提出了直观定义平滑隶属函数的算法，通过复合立方指数厄米插值参数曲线，使现有的基于截止时间和执行时间的调度算法更加现实化与具体化。Nasser 等[3]基于先来先服务分组调度机制，提出了动态多级优先级分组调度方案，将实时数据放置于最高优先级队列，非实时数据根据预计时间放置于其他 2 个队列，以此减少端到端时延。为解决传统调度方法时间复杂度问题，Benitez 等[4]提出了动态优先级交换调度策略，把故障和时延这 2 个特征参数作为扰动因素，来控制非线性时延耦合及固有局部故障的出现。洪雪玉等[5]提出了基于截止期和速率的调度算法，该算法优先级取决于重要性和紧急性这 2 个特征参数，提高了调度的效率和可行性。王永炎等[6]综合考虑了任务的截止期和价值这 2 个特征参数，从累积实现价值率、加权截止期保证率及差分截止期保证率这 3 个方面提出了一种基于优先级表的实时任务调度算法。陈辉[7]提出了基于价值密度及紧迫性这 2 个特征参数的动态分配算法。陈佐瓒等[8]提出了多服务器节点协同调度算法和基于节点计算能力的调度策略，保证了调度器运行的可靠性和子任务的自适应并行能力。

以上研究或是针对调度过程某一特征参数展开研究，或是将任务单一的时间属性和价值作为多特征研究，并没有针对网络实时调度问题进行全面深入的讨论。对此，本文提出了一种基于多特征动态优先级的网络实时调度算法。通过构建实时调度系统结构，建立调度系统的任务模型。根据任务的截止期、执行时间以及间隔时间等时间属性，提出了任务的迫切度；分析不同任务在实时调度系统中功能与重要度的相异程度，提出了基于服务质量保证的任务松紧度。通过对任务迫切度与松紧度进行动态调节，给出了防止任务频繁切换的颠簸限度，在减少任务切换次数的同时，保证了等待任务的抢占成功率，从而提高了全部任务执行的成功率与客户端执行的利用率。

2 网络实时调度系统任务模型的建立

2.1 实时调度系统体系结构的构建

基于城市–客户端体系结构构建的实时调度系统模型是一个类似树状的层次结构。以服务器表示系统控制中心，调度器处于服务器与城市节点之间，负责各任务的分配调度。调度器对各城市节点及其下属客户端的作业进行多城市节点协同调度的方案，客户端 C_{ij} 由其所属城市节点 U_i 直接管理，每个城市节点管理多个客户端，同时客户端的任务执行状态信息实时反馈到调度器，从而调整任务的动态优先级及排序，保证了服务器运行的可靠性。图 1 为基于城市–客户端的实时调度系统的体系结构。

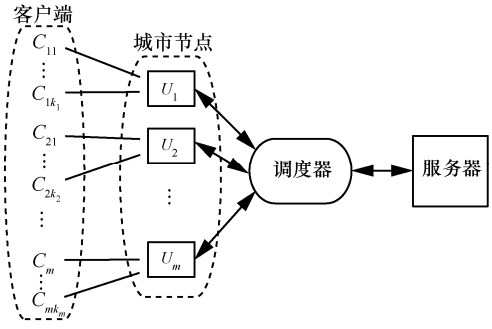

图 1 基于城市–客户端的实时调度系统的体系结构

服务器处于系统体系结构的顶层，这种并不复杂的树状结构具有更高的处理效率与可靠性。服务器通过调度器与各城市节点进行信息传输，完成任务分配及监控。客户端在系统结构中被对应城市节点 U_i 分成小组$\{C_{ij}\}$，从而完成调度器下派并行任务的运算，其关系与客户机–服务器模型的运行模式相似。

2.2 任务模型的建立

在实时调度系统中被调度的任务被称为实时任务，能被系统调度与执行的任务单元称为作业[9]。用 $T=\{\tau_1,\tau_2,\cdots,\tau_n\}$ 表示一列分派的任务集，$\tau_i=\{T_1,\ T_2,\cdots,T_n\}$ 表示 T 的第 i 个子任务集，任务集中每个任务 T_i 定义为 $T_i=\{E_i,d_i,t_i,p_i,\beta_i,\mathrm{TD}_i\}$，并定义 T_{ij} 为任务 T_i 的第 j 个作业。任务模型中，E_i 是任务 T_i 理论执行时间，b_i 是任务 T_i 的开始时间，d_i 是任务 T_i 的绝对截止期，t_i 是当前任务已执行的时间，p_i 是 2 个作业执行的最小间隔时间，β_i 是任务 T_i 的执行迫切度，TD_i 是任务 T_i 服务质量保证的松紧度。图 2 为任务集 T 的时间属性。

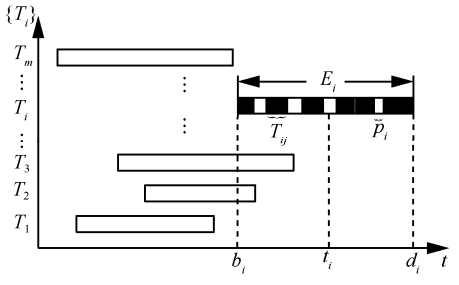

图 2 任务集 T 的时间属性

若一个任务集 T 中全部作业都能确保在绝对截止期前执行完毕，则称 T 可调度[9]。实时调度系统体系平台包含 $N_C=\sum_{i=1}^{m}k_i$ 个客户端，城市节点 U_i 下属的客户端用 $C_{ij}(1\leqslant i\leqslant m,1\leqslant j\leqslant k_i,j$为$U_i$下属第$j$个客户端) 表示。用 $T(C_{ij})$表示任务集 T 中分配给客户端 C_{ij} 的任务子集，此处作业包含 2 个部分：分配到此客户端的任务 T_i 的所有作业和分配到此客户端的任务 T_i 的部分作业。客户端 C_{ij} 的执行利用率 $u_{C_{ij}}$ 与客户端数量 N_C 作为反馈指标，来影响调度器对任务的分派，如式(1)所示。

$$u_{C_{ij}}=\sum_{T_i\in T(C_{ij})}\frac{E_i}{p_i} \tag{1}$$

系统的总客户端资源利用率 U_C 为

$$U_C=\sum_{i=1}^{N}\sum_{j=1}^{k_i}u_{C_{ij}} \tag{2}$$

一般地，每个城市节点接受服务器分派下来的任务数量 z 远小于任务总数 n。例如某个任务集有 10 000 个任务，即 $n=10\,000$，若有 20 个城市节点，则每个城市节点平均分配了 500 个任务，即 $z=500$。此时，城市节点调度有 3 种情况，其中 IN_{ic} 表示 U_i 下属空闲客户端数量。

1) $\mathrm{IN}_{ic}<z$。不能满足所有分派的任务立即投入运行。城市节点先向下属空闲客户端分派任务并进入执行操作，剩余的任务反馈回调度器中进行调度，等待分派。

2) $\mathrm{IN}_{ic}=z$。城市节点下属的空闲客户端数量恰好满足全部分派的任务投入运行。这种理想状态下，城市节点恰好将全部任务分派给每一个空闲客户端执行操作。

3) $\mathrm{IN}_{ic}>z$。能够满足所有分派的任务运行之外还有空余，资源处于充足状态，理论上要把任务尽可能分配到处理能力强的客户端运行。

3　实时调度系统的动态特征参数与测量指标

3.1　调度系统的任务迫切度

客户端经过需求界限函数（DBF, demand bound function）算法[10]可以实现合理的调度，传统的基于任务时间属性的调度策略一般仅依据任务的截止期或间隔时间来评判任务执行的紧迫性[11-12]，而这种单一的时间属性决策未能全面地表现出实时系统的迫切性。本文综合考虑实时调度系统中任务的截止期、任务的剩余执行时间及间隔时间等时间属性，提出任务迫切度来度量评价任务执行的迫切性，较全面地考虑了任务的时间约束，能更准确地对任务执行的迫切程度进行度量。

分析 E_i、t_i、d_i 的关系，把绝对剩余时间与完成任务 T_i 的执行时间的比值定义为任务执行率 α_i，即

$$\alpha_i=\frac{d_i-t}{E_i-t_i} \tag{3}$$

其中，t 为调度系统的时间。在 t 与 t_i 随着执行时间的增加而增加的过程中，α_i 在不断地变大，即执行任务的任务执行率要随着距离截止期越近而变得越大，从而提高其在截止期前完成任务的机会。

为了降低执行任务 T_i 因没有足够的执行时间而夭折的风险，根据任务的执行率，本文提出了执行任务迫切度 β，它表示为确保任务在任务截止期前完成，要求执行此任务的迫切程度。任务 T_i 的迫切度 β_i 为

$$\beta_i=p^{\alpha_i}=p^{\frac{d_i-t}{E_i-t_i}} \tag{4}$$

其中，p 为调节任务执行利用率 α_i 对迫切度 β_i 的作用程度的参数，定义为任务任意相邻 2 个作业的间隔时间与任务作业最小间隔时间的比值，显然 $p\geqslant 1$，能够得出 $\beta_i\in\left[p,p^{\frac{d_i}{E_i}}\right]$。执行任务 T_i 的迫切度会随执行时间的增加而增加，从而保证了正在执行的任务保护自己的执行状态而不被其他任务抢占。

3.2　基于任务服务质量保证的松紧度

3.2.1　实时系统的监测服务质量保证

实时系统的监测服务质量保证（SLA, service-level assurance）是与被测实时系统之间进行的约定。根据约定的不同，SLA 主要分为以下 3 种情况。

1) 城市/监测客户端服务质量保证。对每个城市监测客户端数以及任务监测客户端组中的城市数均进行约定，即约定了城市客户端乘积数。

2) 城市服务质量保证。对城市监测总数进行约定。

3) 监测客户端服务质量保证。对监测客户端总

数进行约定。

在 SLA 下，初始优先级由调度器中的任务属性与参数决定；除此之外，为了均衡任务分发过程，定义了任务分发调节因子 θ，确保优先调度完成情况较差的任务。任务分发调节因子主要与作业的完成情况相关，由任务执行的剩余城市数或客户端数决定。SLA 下的优先级策略的设计应该遵循以下原则。1) 为了保证监测数据全面客观，基于监测城市的作业优先级应尽量高于基于监测客户端的作业；2) 为了保证每个监测任务都有机会被执行，应尽量照顾作业分发情况较差的作业。

由于任务的分发情况和剩余时间会随着监测任务的执行不断发生变化，因此 θ 也相应变化。当任务分发和执行情况发生变化或者新增删减任务时，需要重新计算任务的优先级。

实时调度系统在满足任务的时间约束条件与 SLA 时，下一层的客户端节点由确定的上一层城市节点管理，避免因动态性太强所引起的节点间的低效率传输和搜索，使系统更加适合于实时调度系统这样的主−从系统的应用。这样，实时调度系统体系结构的服务质量保证受城市数量以及其下属客户端数量的共同作用影响。

3.2.2 基于 SLA 的松紧度

由于调度器下发的不同任务在系统中功能有所差异，相同的任务在执行或等待状态中对系统的重要程度也会变化，因此定义任务服务质量保证的松紧量来量化任务 T_i 对实时调度系统不同情况的重要性。显然，松紧量并非在任务完成那一时刻才产生，而是伴随任务执行过程而如弹力一般渐渐增大的。

当某客户端的实时任务 T_i 开始执行 t_i 时间后，其积累的服务质量松紧量记为 TQ_i，即松紧量是随时间变化表示任务分发情况有关的函数，松紧量的计算式为

$$\mathrm{TQ}_i = \int_0^{t_i} f(\mathrm{IN}_x)\mathrm{d}t \tag{5}$$

显然，$f(\mathrm{IN}_x)$ 表示单位时间内松紧量的变化程度，是与时间无关的一个变量。

任务 T_i 预期的松紧量与其相应执行时间的比为平均松紧量 $\overline{\mathrm{TQ}_i}$，即 $\overline{\mathrm{TQ}_i} = \dfrac{\mathrm{TQ}_i}{E_i}$。可以看出，$\overline{\mathrm{TQ}_i}$ 只和 T_i 自身属性相关，而与 T_i 执行过程无关。不过 $\overline{\mathrm{TQ}_i}$ 反映的是一段执行时间 t_i 内任务的平均松紧度，不能反映 T_i 的即时松紧质量，因此定义服务质量保证的松紧度 TD_i，TD_i 反映任务 T_i 松紧量的变化速度，可表示为

$$\mathrm{TD}_i = \lim_{\Delta t \to 0} \frac{\Delta \mathrm{TQ}}{\Delta t} \tag{6}$$

由式(5)和式(6)，显然有式(7)成立。

$$\mathrm{TD}_i = f(\mathrm{IN}_x) \tag{7}$$

根据 SLA 约定的分类不同，松紧度计算式也不完全相同，而由于每个任务只能选择一种 SLA，因此任务松紧度有以下 3 种情况。

1) 城市/监测客户端服务质量保证

$$f(\mathrm{IN}_{uc}) = \mathrm{UCS}\frac{\mathrm{IN}_{uc}}{\mathrm{NN}_{uc}}\left[\left(1-\frac{1}{\mathrm{NN}_{uc}}\right)^{\alpha_{uc}}\mathrm{UCS}\right] \tag{8}$$

其中，IN_{uc} 为城市/监测客户端服务质量保证任务未分配的作业总数，NN_{uc} 为城市/监测客户端服务质量保证分派任务的作业总数，UCS 为城市/监测客户端服务质量保证调节系数，任务分发调节因子 $\theta = \mathrm{UCS}\dfrac{\mathrm{IN}_{uc}}{\mathrm{NN}_{uc}}$。$f(\mathrm{IN}_{uc})$ 为关于 IN_{uc} 的单增函数，当 IN_{uc} 从 NN_{uc} 变化到 0 时，松紧度的取值范围为 $\left[\mathrm{UCS}^2\left(1-\dfrac{1}{\mathrm{NN}_{uc}}\right)^{\alpha_{uc}},0\right]$。

2) 城市服务质量保证

$$f(\mathrm{IN}_u) = \mathrm{US}\frac{\mathrm{IN}_u}{\mathrm{NN}_u}\left[\left(1-\frac{1}{\mathrm{NN}_u}\right)^{\alpha_u}\mathrm{US}\right] \tag{9}$$

其中，IN_u 为城市服务质量保证任务未分配的作业总数，NN_u 为城市服务质量保证分派任务的作业总数，US 为城市服务质量保证调节系数，任务分发调节因子 $\theta = \mathrm{US}\dfrac{\mathrm{IN}_u}{\mathrm{NN}_u}$。$f(\mathrm{IN}_u)$ 为关于 IN_u 的单增函数，当 IN_u 从 NN_u 变化到 0 时，松紧度的取值范围为 $\left[\mathrm{US}^2\left(1-\dfrac{1}{\mathrm{NN}_u}\right)^{\alpha_u},0\right]$。

3) 监测客户端服务质量保证

$$f(\mathrm{IN}_c) = \mathrm{CS}\frac{\mathrm{IN}_c}{\mathrm{NN}_c}\left[\left(1-\frac{1}{\mathrm{NN}_c}\right)^{\alpha_c}\mathrm{CS}\right] \tag{10}$$

其中，IN_c 为监测客户端服务质量保证任务未分配的作业数，NN_c 为监测客户端服务质量保证分派任务的作业总数，CS 为监测客户端服务质量保证调节系数，任务分发调节因子 $\theta = CS\frac{IN_c}{NN_c}$。$f(IN_c)$ 为关于 IN_c 的单增函数，当 IN_c 从 NN_c 变化到 0 时，松紧度的取值范围为 $\left[CS^2\left(1-\frac{1}{NN_c}\right)^{\alpha_c},0\right]$。

根据 SLA 下优先级策略的设计原则，任务选择城市服务质量保证优先于城市/客户端服务质量保证，最后选择客户端服务质量保证，并结合工程实际应用数据，选取 $\alpha_{uc}=20$，$\alpha_u=10$，$\alpha_c=10$。

由基于 SLA 的任务的松紧度可知，任务只在完全完成后才会给实时调度系统产生任务的松紧量“放松”，否则，其松紧度的“弹力”将一直存在而无法放松，若在绝对截止期前尚未释放弹力，则表示任务未能成功执行；而若夭折一个已产生松紧量的任务，那么不仅不能为系统释放“弹力”，还用掉了任务执行时消耗的系统资源。对于本文的实时调度系统，为了保护等待任务在绝对截止期前能够得到“放松”，根据 SLA 选择原则及计量关系，在任务分派过程中，未分配的作业在减小，即执行任务的松紧度在降低，这增加了等待任务抢占执行位置的机会，从而为等待任务的作业分配空闲的客户端，并提高其动态优先级，在一定程度上减少等待任务 T_i 夭折的可能性，最终提高了实时调度系统的执行成功率。

3.3　实时任务调度算法的性能测量指标

基于城市−客户端体系结构的网络实时调度系统的调度算法应用如表 1 所示的性能测量指标。

以上实时任务调度算法的性能测量指标中，调度成功率是任务集全部完成的决定性评价标准。在执行任务时，需要设定有关迫切度 β_i 和松紧度 $TD(t_i)$ 的相关参数，图 3 为迫切度和松紧度对资源利用率的影响。

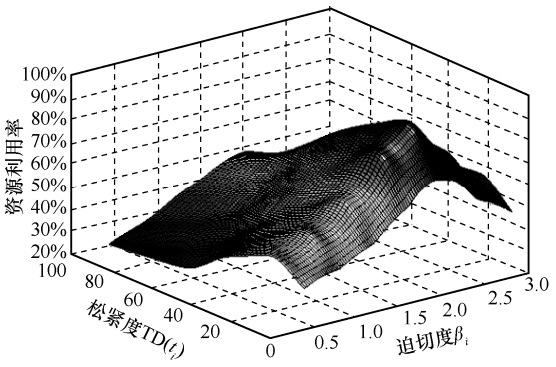

图 3　迫切度和松紧度对资源利用率的影响

4　基于多特征动态优先级的调度策略定义及流程

4.1　颠簸限度

在实时调度系统中，多个任务优先级交替上升而导致任务相互交叉抢占执行，而每次抢占都发生一次任务切换，这种频繁切换现象称为系统颠簸现象[13]。颠簸现象会导致系统的额外开销大大增加，消耗系统资源。为此本文提出了颠簸限度来提高任务的抢占门限，避免优先级差别很小的任务之间的过度抢占。

在任务集 T 中，2 个任务 T_1 与 T_2 之间设定一个颠簸限度 ω，其中 T_1 是正在执行的任务，T_2 是等待任务，它们在 t 时刻的动态优先级分别是 $DP(T_1)$ 和 $DP(T_2)$。只有在满足式(11)的条件下，才能够让 T_2 抢占 T_1，从而避免频繁抢占。

$$DP(T_2) > \omega DP(T_1) \tag{11}$$

假设在 SLA 的城市/监测客户端服务质量保证下，在 t_i 时刻，执行任务 T_1 服务质量保证的松紧度最大，为 $UCS^2\left(1-\frac{1}{NN_{uc}}\right)^{20}$，执行迫切度最小，为 β_i；等待任务 T_2 服务质量保证的松紧度最小，为 0，执行迫切度也最小，为 β_i。经过 t_i 时间后满足颠簸门限，T_2 抢占 T_1，T_1 从执行状态转变为等待状态，其等待迫切度将保持不变，服务质量保证的松紧度逐渐增大；T_2 从等待状态转变为执行状

表 1　　**实时任务调度算法的性能测量指标**

性能测量指标	定义	特性
调度成功率	在截止期内成功完成的任务执行进度占总任务执行进度的比率	$\mu = \frac{\text{已完成执行进度}}{\text{总执行进度}} \times 100\%$
切换次数	任务执行状态下切换的总次数	作业切换次数越少，系统的开销越小
客户端资源利用率	任务的执行时间与任务连续 2 个作业间最小时间间隔的比率，如式(1)和式(2)所示	客户端资源利用率越大越好

态，其服务质量保证的松紧度保持不变，执行迫切度逐渐增大。到 t_i 时刻，T_1 执行迫切度从最小值增加到最大值 β_j，有

$$\omega \geqslant \max\left(\frac{\mathrm{DP}_j(T_1)}{\mathrm{DP}_j(T_2)}\right)=\frac{\max(\mathrm{DP}_j(T_1))}{\max(\mathrm{DP}_j(T_2))}\geqslant 1 \quad (12)$$

故对于任意任务集 T，一定存在某个 ω，使调度系统避免出现颠簸现象。如初始优先级较高的任务 A 和优先级稍低一些的任务 B 处于等待任务状态，某一时刻任务 A 进入执行状态，若 ω=3，那么只有当 $\mathrm{DP}(T_i)>3\mathrm{DP}(T_j)$ 时，任务 B 才能抢占成功，进入执行状态。

4.2 算法流程及步骤

1) 系统实时调度算法的流程

在系统调度过程中，对所有任务的动态优先级都进行实时监控，并进行重新排序，然后将执行任务的优先级与等待任务中优先级最高的任务进行比较，依据结果选用对应策略进行调度，从而使系统的运行性能最佳[12]。

基于多特征动态优先级的调度策略（MDPSS, multi-feature dynamic priority scheduling strategy）的流程如图 4 所示。当实时系统开始运行时，服务器将任务集下发到调度器继而分派到城市节点，此时初始优先级最高的任务率先抢占城市客户端，进入

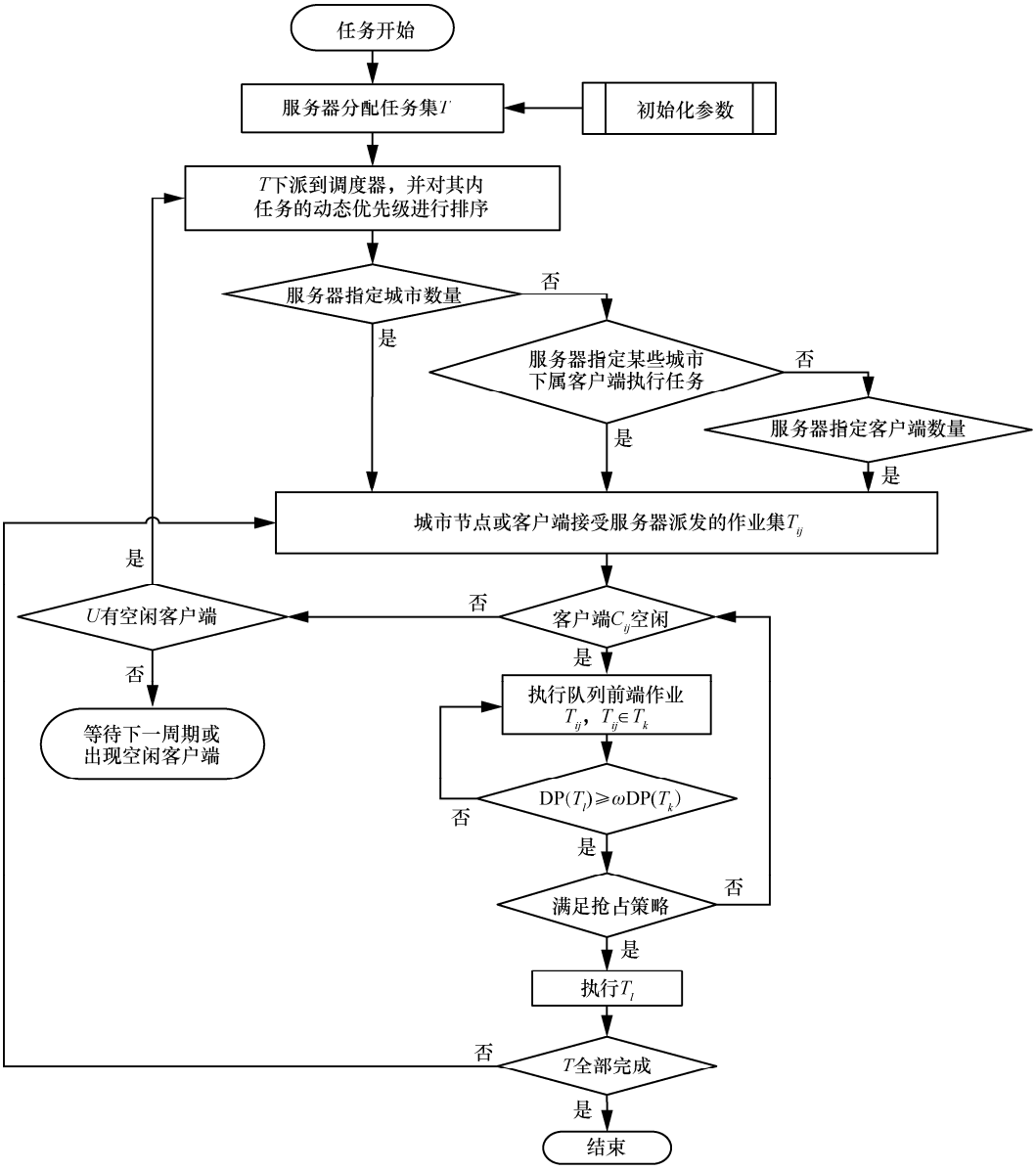

图 4 基于多特征动态优先级的调度策略的流程

任务执行状态，而所有任务的优先级都伴随执行任务的进行而动态变化。对客户端中的执行任务，其执行迫切度不变，但是其服务质量保证的松紧度随着已执行时间的增加而不断变大，“弹力”的张紧可以避免被非执行任务过早抢占而发生颠簸现象的产生；对调度器中的等待任务与活动任务，其松紧度不变，但是其执行迫切度随着其等待时间的增加而变大，“等待”的迫切程度增大了它抢占任务执行权的机会。

2) 实时调度算法的抢占策略及步骤

根据任务属性以及调度流程，为任务之间的抢占制定一些抢占策略。在实时调度系统中，当任务集 T 中处于等待状态的 T_2 满足抢占 T_1 的条件，且没有其他影响的情况下，无论 T_2 是否抢占 T_1，最终 T_1 与 T_2 都满足截止期，可以完成执行任务，此时有以下 2 个抢占策略可以选择。

策略 1　积极抢占策略——T_2 抢占 T_1。中断任务 T_1 的执行进程，执行任务 T_2，T_2 完成后再继续执行 T_1。

策略 2　消极抢占策略——T_2 不抢占 T_1。T_1 全部完成后 T_2 才开始执行，此时设 T_2 优先级为 $\max(\mathrm{DP}'(T_1),\mathrm{DP}'(T_2))$，$\mathrm{DP}'(T_1)$ 与 $\mathrm{DP}'(T_2)$ 是任务在切换时刻各自的动态优先级。

根据调度流程以及任务抢占策略，调度算法的具体步骤如下。

步骤 1　服务器向调度器下发任务集 $T=\{T_1, T_2,\cdots,T_n\}$，初始化任务的优先级。

步骤 2　将任务在调度器中按初始动态优先级 $\{\mathrm{DP}_0(T_i)\}$ 进行排序，城市节点 U_i 把接收的作业集 $\{T_{ij}\}$ 按调度策略分派到下属的客户端 C_{ij}。

步骤 3　服务器会实时监控调度器的运行信息。一种情况是，由服务器根据 SLA 不同情况，指定某城市的某些客户端执行任务；另一种情况是，分析城市节点 U_i 接收的任务数量 n_i 与其对应空闲客户端节点数 IN_{C_i} 的关系。

1) 当 $\mathrm{IN}_{C_i} \geqslant m$ 时，说明资源比较充裕，此时任务分配到负载最小的城市节点，即选择下属空闲客户端最多的城市节点进行下一步的调度。

2) 当 $\mathrm{IN}_{C_i} < m$ 时，服务器根据任务要求，可仅分配现有的空闲客户端，也可以将所有任务都调度出去等待有客户端空闲后，将任务投入下个周期运行。

步骤 4　在客户端 C_{ij} 的某时刻，动态优先级最高的任务 T_k 处于执行状态，且已执行 t_k 个单位时间，其动态优先级记 $\mathrm{DP}(T_k)$；在等待状态的任务中，T_l 为动态优先级最高任务，且已执行 t_l 个单位时间，其动态优先级记记 $\mathrm{DP}(T_l)$。

1) 若 $\mathrm{DP}(T_l) < \omega\mathrm{DP}(T_k)$，则 T_l 没有满足抢占 T_k 条件，T_k 与 T_l 均保持原状态。

2) 若 $\mathrm{DP}(T_l) \geqslant \omega\mathrm{DP}(T_k)$，则 T_l 满足抢占 T_k 条件，判定进入积极抢占策略（T_l 抢占 T_k）还是消极抢占策略（T_l 不抢占 T_k）。

3) 若 T_l 在 T_k 完成后再开始执行，T_l 依旧可保证其截止期；而如果 T_l 抢占 T_k，T_k 在 T_l 完成后再继续执行，T_k 也能确保其截止期，则按积极抢占策略或消极抢占策略执行。

4) 如果 T_l 不抢占 T_k 就不能保证截止期，则 T_l 必须执行抢占。在这种情况下，若 T_k 在 T_l 完成后继续执行仍能满足截止期，则执行积极抢占策略；若 T_l 抢占 T_k 后，T_k 不能满足其截止期而夭折，则将 T_k 未完成部分自动分配到下一个周期或其他空闲客户端。

步骤 5　服务器返回任务分派的客户端的任务信息和调度器中各城市节点所分配的客户端数，调度完成。

5　实例分析与仿真实验

为了验证 MDPSS 算法调度性能的优越性，将本文所提算法结合某“高并发业务性能的监测任务”相关项目实施，研究对监测客户端的动态调度问题，要求根据任务列表产生任务切片，通过计算任务切片的优先级生成调度队列，按任务调度监测客户端完成监测任务。

实验采用 MATLAB 软件进行仿真。在仿真中模拟局域网中的运行环境，调度 10 000 个分布在各地的模拟监测客户端，实验中的参数设定为 $\beta_i=2$，$\mathrm{TD}(t_i)=15$，与之相比较的尽力交付（BE, best effort）算法中 V=1，最早截止时间优先（EDF, earliest deadline first）算法中 Gr=0.4[14]。从调度成功率、任务切换次数、平均响应时间 3 个方面进行仿真结果比较。硬件测试平台的配置为 Intel Core i5 4590 处理器、DDR3 1600 8 GB 内存、HD4600 显卡、Windows severe 2008 操作系统。

5.1　算法调度成功率

调度成功率是衡量调度算法的一个重要指标，目前流行的 3 种算法均会优先考虑成功率的问题，再去优化其他性能指标。为验证算法的优越性，本节分别依据 3 种算法在客户端资源利用率变化时抽取样本进行测试，仿真出调度成功率的变化曲线，如图 5 所示。

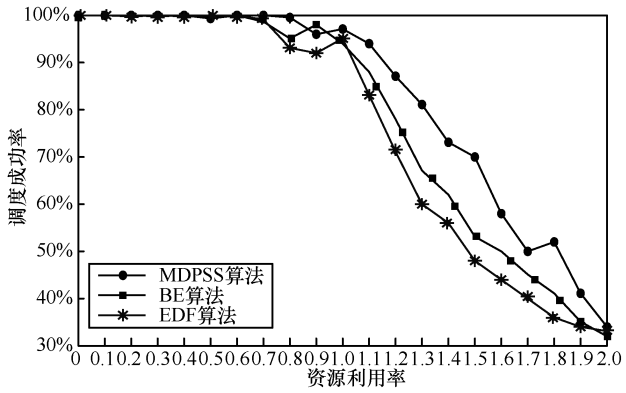

图 5 调度成功率随资源利用率的变化曲线

由图 5 可知，当资源丰富时，即客户端资源利用率在不足 1 的条件下，3 种算法的调度成功率几乎都能达到 100%；而在资源不足的条件下，随着任务超载量的增加，调度成功率越来越低，此时 MDPSS 算法的优越性逐渐体现出来。

5.2 任务切换次数

在实施抢占策略的实时调度系统中，当满足抢占条件时，正在执行的任务会暂时中断而将资源让给优先级更高的任务。在空闲客户端数量有限的条件下，任务的相互抢占过程会引起资源的额外开销，从而对系统性能造成严重的影响。在任务调度过程中，随着客户端资源利用率的变化，3 种算法的切换次数如图 6 所示。

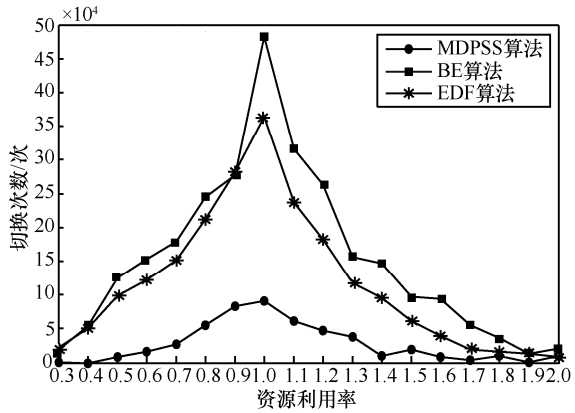

图 6 3 种算法的切换次数

由图 6 可知，3 种算法在资源充足时，随着客户端资源利用率的增加，切换次数也在增加。当客户端资源利用率为 1 时，切换次数达到最大值；当客户端资源利用率大于 1 时，随着资源的超载，3 种算法均会更加顾及资源在其他方面的需求，随着超载量的增加而减少了切换次数。与 BE 算法和 EDF 算法相比，MDPSS 算法由于引入了颠簸限度，因此切换次数远远小于这 2 种算法。

5.3 算法平均响应时间

平均响应时间表征执行任务的速度。采用以下方式对网络监测任务调度时间进行统计：模拟测试环境主要在局域网中搭建，包括一台服务器，运行任务分发程序，安装 MySQL 数据库；一台测试终端用以运行测试脚本，并记录测试结果。在正常工作条件下随机抽取 100 ms，将返回数据记录结果以直方图的形式进行统计，如图 7 所示。

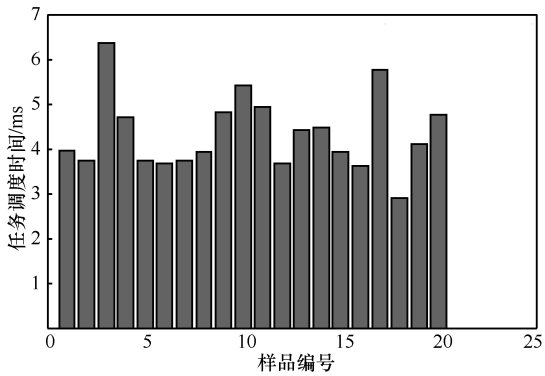

图 7 网络检测任务调度时间直方图

通过对测试数据进行分析可以看出，在局域网环境中，在网络时延分组丢失忽略不计的情况下，平均每次任务调度时间可控制在 7 ms 以内，而绝大多数任务的调度时间集中在 3~5 ms，因此每小时可调度任务片可达 4×10^9~7×10^9 个，高效地完成了任务的调度。

为验证算法在响应时间上的优越性，需要在客户端资源利用率变化时，尤其是在超载的情况下，比较 3 种算法的调度平均响应时间，分别依据 3 种算法抽取样本，拟合出客户端资源利用率在 0.5~1.8 上变化时的调度平均响应时间曲线，如图 8 所示。

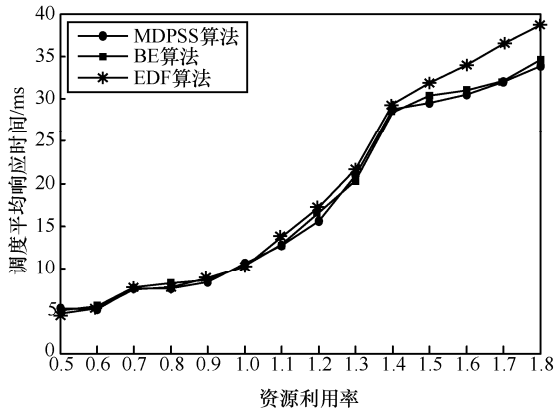

图 8 调度平均响应时间随资源利用率的变化曲线

由图 8 可以看出，随着客户端资源利用率的增加，调度平均响应时间也相应地逐渐增加，在资源丰富的条件下，即客户端资源利用率小于 1 时，3 种算法的平均响应时间相差不大；随着过载量的增加，MDPSS 算法的优势逐渐显现出来，平均响应时间与 BE 算法相差不大，且优于 EDF 算法。

6　结束语

本文通过构建实时调度系统体系结构，建立起任务模型，研究了实时调度系统中任务关于时间属性的迫切度与任务服务质量保证的松紧度的特性及其关系。根据任务的 3 个时间属性，改进了一种 DBF 算法，提出了任务的迫切度；通过分析不同任务在城市−客户端系统中功能与重要度的相异程度，定义了任务的服务质量保证的松紧度。此外，综合这 2 个方面的特征，提出了颠簸限度来提高任务的抢占门限，由此提出了 MDPSS 算法。

仿真实验结果表明，与 BE 算法和 EDF 算法相比，MDPSS 算法不仅能根据任务动态优先级有效而安全地进行任务切换，在减少任务切换次数的同时，也保证了等待任务的抢占成功率，从而提高了全部任务的调度成功率与客户端资源利用率，缩短了平均响应时间，具有明显的优越性。

参考文献：

[1] SEMGHOUNI S, AMANTON L, SADE B, et al. On new scheduling policy for the improvement of firm RTDBSs performances[J]. Data & Knowledge Engineering, 2007, 63(2): 414-432.

[2] MUHURI P K, SHUKLA K K. Real-time scheduling of periodic tasks with processing times and deadlines as parametric fuzzy numbers[J]. Applied Soft Computing, 2009, 9(3): 936-946.

[3] NASSER N, NIDAL K, LUTFUL T, et al. Dynamic multilevel priority packet scheduling scheme for wireless sensor network[J]. IEEE Transactions on Wireless Communications, 2013, 12(4): 1448-1459.

[4] BENITEZ P H, BENITEZ P A, ORTEGA A J, et al. Networked control systems design considering scheduling restrictions and local faults[J]. International Journal of Innovative Computing Information and Control, 2012, 8(12): 8515-8526.

[5] 洪雪玉, 张凌, 袁华. Linux 下的实时调度算法[J]. 华南理工大学学报, 2008, 36(4): 104-109.
HONG X Y, ZHANG L, YUAN H. Real-time scheduling algorithm of Linux[J]. Journal of South China University of Technology, 2008, 36(4): 104-109.

[6] 王永炎, 王强, 王宏安, 等. 基于优先级表的实时调度算法及其实现[J]. 软件学报, 2004, 15(3): 360-370.
WANG Y Y, WANG Q, WANG H A, et al. A real-time scheduling algorithm on priority table and its implementation[J]. Journal of Software, 2004, 15(3): 360-370.

[7] 陈辉. 实时任务优先级动态分配策略[J]. 小型微型计算机系统, 2010, 31(7): 1385-1388.
CHEN H. Dynamic priority assignment strategy for real-time task[J]. Journal of Chinese Computer Systems, 2010, 31(7): 1385-1388.

[8] 陈佐瓒, 徐胜超. 面向互联网计算资源共享的自适应调度模型[J]. 计算机工程与应用, 2010, 46(21): 86-89.
CHEN Z Z, XU S C. Adaptive-parallelism scheduling for Internet-based volunteer computing system. Computer Engineering and Applications, 2010, 46(21): 86-89.

[9] 巴巍. 实时系统动态优先级任务调度算法的研究[D]. 大连: 大连理工大学, 2010.
BA W. Research of real-time system dynamic priority assignment scheduling algorithm[D]. Dalian: Dalian University of Technology, 2010.

[10] BAURAH S, FISHER N. The partitioned dynamic-priority scheduling of sporadic task systems[J]. Real-Time Systems, 2007, 36: 199-226.

[11] HARITSA J R, LIVNY M, CAREY M J. Earliest deadline scheduling for real-time database systems[C]//Proceedings of the 12th IEEE Real-Time Systems Symposium. Piscataway: IEEE Press, 1991: 232-243.

[12] 夏家莉, 陈辉, 杨兵. 一种动态优先级实时任务调度算法[J]. 计算机学报, 2012, 35(12): 2685-2695.
XIA J L, CHEN H, YANG B. A real-time tasks scheduling algorithm based on dynamic priority[J]. Journal of Computer, 2012, 35(12): 2685-2695.

[13] 金宏, 王宏安, 王强, 等. 改进的最小空闲时间优先调度算法[J]. 软件学报, 2004, 15(8): 1116-1123.
JIN H, WANG H A, WANG Q, et al. An improved least-slack-first scheduling algorithm[J]. Journal of Software, 2004, 15(8): 1116-1123.

[14] 萧伟, 冯治宝, 应启戛, 等. 改进型 EDF 调度算法的研究与实现[J]. 计算机工程, 2009, 35(18): 231-233.
XIAO W, FENG Z B, YING Q G, et al. Research and implementation of improved earliest deadline first schedule algorithm[J]. Computer Engineering, 2009, 35(18): 231-233.

[作者简介]

苏洵（1965– ），女，北京人，61623 部队正高级工程师，主要研究方向为数据通信、计算机网络建设规划、运维管理及性能优化、数据中心建设规划及运维管理等。

李艳芳（1973– ），女，湖北武汉人，61516 部队高级工程师，主要研究方向为计算机网络运维管理、网络性能优化、信息服务。

宗宁（1980– ），女，山东济宁人，61623 部队高级工程师，主要研究方向为计算机网络运维、网络性能优化、信息服务等。

魏巍（1982– ），男，辽宁沈阳人，博士，北京航空航天大学副教授、博士生导师，主要研究方向为智能制造技术及应用。

李娟（1979– ），女，河北固安人，61623 部队工程师，主要研究方向为计算机网络运维管理、网络性能优化、信息服务、物联网等。

丁莹（1981– ），女，山东沾化人，61623 部队工程师，主要研究方向为计算机网络运维、网络性能优化、信息服务等。