非精确计算中基于反馈的 CPU 在线调度算法*

张尧学+, 方存好, 王 勇

(清华大学 计算机科学与技术系,北京 100084)

A Feedback-Driven Online Scheduler for Processes with Imprecise Computing

ZHANG Yao-Xue+, FANG Cun-Hao, WANG Yong

(Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China)

+ Corresponding author: Phn: +86-10-62782118, E-mail: zyx@moe.edu.cn, http://www.cs.tsinghua.edu.cn

Received 2002-09-02; Accepted 2003-05-27

Zhang YX, Fang CH, Wang Y. A feedback-driven online scheduler for processes with imprecise computing. *Journal of Software*, 2004,15(4):616~623.
http://www.jos.org.cn/1000-9825/15/616.htm

Abstract: With an increasing requirement of more flexible real-time applications, e.g. multimedia servers in home networks and real-time database servers, a real-time process scheduler using Worst-Case Execution Time (WCET) is inefficient for optimizing performance. Some soft and firm real-time models have been proposed to deal with this situation. This paper presents a feedback control approach for scheduling processes with imprecise computation, a firm real-time model to produce approximate result of an acceptable quality when the exact result of the desired quality cannot be obtained in time. By introducing feedback control to process scheduling, our approach aims to bind the deadline missing ratio under a varying system workload to reach a tradeoff between the deadline missing ratio and result precision.
Key words: real-time scheduling; imprecise computation; feedback control; PID

摘 要: 随着家庭网络中的多媒体服务器和实时数据库服务器这类应用对实时的灵活性的要求不断增加,传统实时基于最长执行时间(WCET)的调度算法已经不能满足它们对性能优化的要求.因此,产生了一些软实时的调度算法来解决这些问题.提出了一种由反馈环节控制的实时调度算法,该算法用于调度能使用不精确计算模型描述的进程.算法可以在各种负载条件下,通过在调度过程中引入的反馈控制,在计算精度和计算时间上直接取得折衷,将进程错过时限的比例控制在预定范围内.
关键词: 实时调度;不精确计算;反馈控制;PID
中图法分类号: TP316 **文献标识码**: A

---

* Supported by the National Grand Fundamental Research 973 Program of China under Grant No.G1998030409 (国家重点基础研究发展规划项目(973))

ZHANG Yao-Xue was born in 1956. He is a professor and doctoral supervisor at the Department of Computer Science and Technology, Tsinghua University. His research areas are network theory and application. **FANG Cun-Hao** was born in 1976. He is a Ph.D. candidate at the Department of Computer Science and Technology, Tsinghua University. His current research areas are network architecture, program mining and services management. **WANG Yong** was born in 1974. He is a Ph.D. candidate at the Department of Computer Science and Technology, Tsinghua University. His current research area is real-time system.

1 Introduction and Related Work

Real-time systems are defined as systems in which the operational correctness depends not only on the results of the computation but also on the time at which these results are produced. In real-time systems, time critical processes must be executed under timeliness limitation, which is usually described as a term deadline. Tasks must either be completed or its execution must start by the deadline. Severe consequences will occur if any time critical task fails to meet its deadline.

However, it is not always possible to schedule every task to meet all their deadlines. This phenomenon is known as transient overload, which represents a system state where the requests for service exceed the system's ability for a limited time causing missed task deadlines. One proposed technique to solve transient overloads in the Quality of Service (QoS) degradable real-time system is the imprecise computation model[1], which serves as a model of flexible applications, e.g. the compression and transformation of multimedia information, signal processing, a heuristic search engine and search algorithm of a large-scale database management system. It aims to make real-time systems both easier to develop and more robust when facing transient overload situations.

The research area of scheduling imprecise computation is very wide[2~11]. All these approaches are proposed for applications presenting various characteristics and requirements. This paper focuses on the problem of online scheduling of a set of sporadic real-time processes with imprecise computation by a feedback-driven scheduler. In particular, a well known Proportional Integrative Derivative (PID) controller is introduced to the scheduler to control the resource usage in a stable way.

The idea to use feedback information in process scheduling has been utilized by general purpose operating systems in the form of multi-level feedback queue scheduling for a long time. Applying feedback control theory, such as PID control, to real-time process scheduling has also been presented in earlier papers[12,13]. However, in terms of imprecise computation task model, the feedback information is first utilized by the scheduler in our approach to determine the degradation of QoS when transient overload happens. The results of the simulation show that feedback driven scheduler can bind the deadline-missing ratio to an expected value under various transient overload levels.

In the following of this paper, we first describe the specification of imprecise computation model in Section 2. The details of our approach are presented in Section 3. In Section 4, the simulation results and the performance analysis are reported. Finally, in Section 5, we summarize our work.

2 Imprecise Computation – The Basic Workload Model

The underlying imprecise computation model used in our approach is described in this section. Let T be a set of n processes, $T=\{\tau_1,\tau_2,\ldots,\tau_n\}$, where each process τ_i could be characterized by

$$\tau_i = (r_i,cm_i,co_i,\delta_i,d_i), \quad 1\le i\le n \tag{1}$$

where each imprecise process τ_i is decomposed into one mandatory part, τim, and one optional part, τio, whose computing times are respectively cm_i and co_i. Here, cm_i is the WCET of τim to guarantee the correctness of the result while co_i is the maximum execution time of τio to produce a precise result.

Let ready time be r_i at which the process τ_i becomes ready for execution. Processes are assumed to be ready for execution upon their arrivals, which means, for each τ_i, its ready time is equal to the arrival time.

Deadline d_i is the time by which τ_i must be completed. Obviously, $d_i\ge r_i+cm_i+co_i$.

Admitted Optional Execution time $\delta_i(0\le\delta_i\le co_i)$ is the part of co_i admitted by the online scheduler.

Some underlying workload assumptions are

A1. The sub-processes τim and τio are preemptable. They have hard and firm deadlines respectively.

Moreover, the processes' arrivals are sporadic.

A2. Partial execution of τio is allowed and τ_i is monotonic, i.e., the quality of the intermediate result is non-decreasing as τio executes longer.

A3. τim is pre-declared and pre-analyzed with known cm_i and co_i before τ_i is submitted to system. All the information has been made available to the scheduler and the admission control by the time of r_i.

A4. Process set T is independent, i.e., $\tau_i(1\le i\le n)$ can be executed in any order. For process set with dependence constraints, it is always possible to transfer it to an independent process set[10].

3 Feedback Driven Scheduling for Imprecise Computation

3.1 Overview of system framework

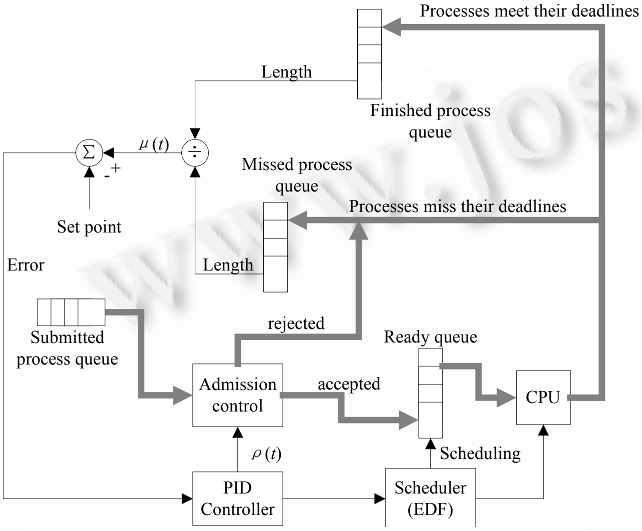


Fig.1 Framework of feedback-driven online scheduler

The framework of our approach contains three main components: The PID controller, scheduler and admission control. The system's framework is shown in Fig.1.

Definition 1. The Option Ratio at time t, notated by $\rho(t)$, is a value between 0 and 1 that for any process in T, its admitted optional execution time is $\sigma_i=\rho(t)co_i$.

Definition 2. The Deadline-Missing Ratio at time t, notated by $\mu(t)$, is the average possibility of processes to miss their deadlines during $[t-Tm,t]$, where Tm is a constant value. In terms of our system framework, Tm is considered to be the maximum time for a process staying in the Missed Process Queue.

When a new process τ_i is submitted to system, it will stay in the Submitted Process Queue until the Admission Control decides whether τ_i should be accepted with respect to the system schedulability. If τ_i is acceptable, Admission Control calculates σ_i with $\rho(t)$ provided by the PID controller. All processes accepted by Admission Control are sent to the Ready Queue. The Scheduler then assigns the CPU time to processes in Ready Queue by EDF (earliest deadline first), a well-known dynamic online real-time scheduling algorithm. Meanwhile, the scheduler monitors the workload and calculates $\mu(t)$ with respect to the current state of the Missed Process Queue during each sampling period. Then, $\mu(t)$ is fed back to the PID Controller to adjust $\rho(t)$ so that the system workload is under control.

3.2 Admission control

We choose Earliest Deadline First (EDF) as the scheduler because it is a deeply studied online scheduling algorithm for dynamic real-time systems. Other online scheduling algorithms are also available in our approach.

EDF is an optimal dynamic scheduling algorithm in a resource sufficient environment. However, it's impossible to guarantee the system's resources are always sufficient in a dynamic system. The performance of EDF algorithm decreases dramatically when the system is overloaded[13]. So, the key point of our approach is to degrade the system's QoS to reduce the workload in an overload situation by truncating the execution of co_i when a process τ_i enters system. In our approach, this is done by Admission Control in a way illustrated in Fig.2.

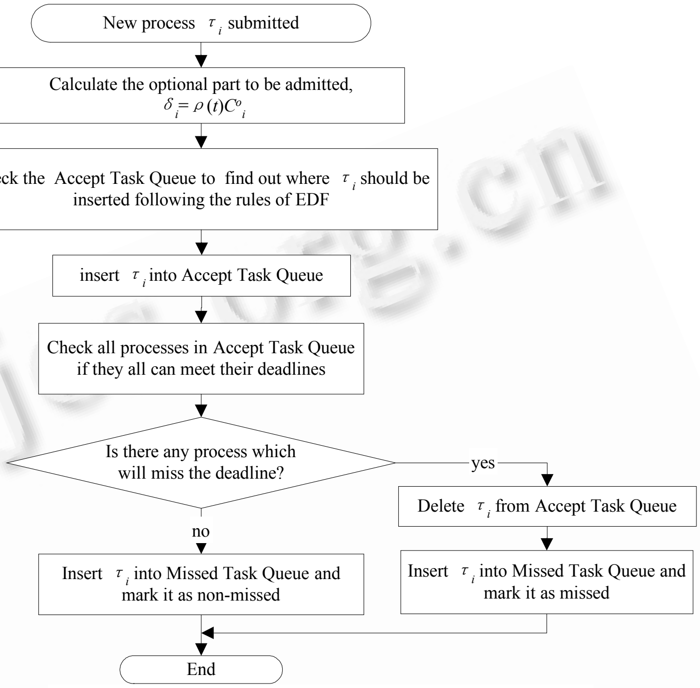


Fig.2 Admission control of τ_i

3.3 Feedback controller

The Feedback Controller is the core of our approach. Although numerous advanced feedback control schemes have been developed so far, the conventional linear PID feedback controller is prevalent in many areas due to its desirable properties such as robustness, reliability and simplicity. Therefore, our approach utilizes the popular PID controller with the EDF scheduler and the Admission Control be viewed as plants to be controlled. However, the control literature offers a wide set of solutions to the same problem. Restricting to the field of linear adaptive control, there are families of controllers other than PID. One of our future works will be testing these possibilities.

The key to building a stable PID controller with high performance is to choose the right parameters. According to the control theory, the PID controller is characterized by these variables:

Controlled Variable: the quantity of the plant's output that is measured and controlled. Obviously, it is $\mu(t)$ in our approach.

Set Point: the expected value, which is denoted by $\mu d(t)$, of the controlled variable. It's the target of the whole control system. In some cases, it appears to be perfect for $\mu(t)$ to be 0, but it must sacrifice the performance for resource utilization. Therefore, for many flexible real-time applications, e.g. multimedia server, a tradeoff should be made to achieve both system feasibility and performance of resource utilization and system throughput.

Error: which is the difference between the set point and controlled variable. It's represented by $e(t)=\mu d(t)-\mu(t)$.

Manipulated Variable: the variable that is regulated by the PID controller to affect the value of the controlled variable. As we mentioned before, $\mu(t)$ is sensitive to system load which is controlled by $\rho(t)$, so, we choose $\rho(t)$ as our system's manipulated variable.

After having identified these variables, we illustrate the block diagram of our scheduling system in Fig.3, where D(s) and G(s) are transfer functions of the PID controller and its plant respectively.

Both the state and the output evolution follow a nonlinear law. Moreover, the system is excited by a random signal: the cim and cio which can not be externally controlled. So, G(s) is very hard to derive for some reason.

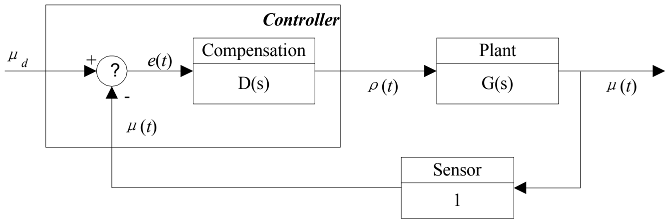


Fig.3 System block diagram

Designing a control law directly tailored on the nonlinear and stochastic system model is a hard process. Thus, in this paper, we decide to estimate an approximate model for tuning the controller's parameters.

The scheduling system calculates the system is deadline-missing ratio, $\mu(t)$, during every sampling period and feeds it back to the controller. The algorithm to calculate $\mu(t)$ is shown in Fig.4.

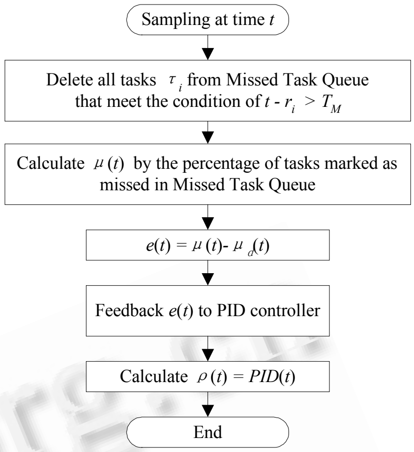


Fig.4 Calculation of $\mu(t)$ in every sampling period

4 Numerical Study

4.1 Simulator

We have developed a uni-processor simulator to study the performance of our approach. In our simulation, processes with imprecise computation are generated randomly following Poisson distribution which means the probability of k processes generated during T is $(\lambda T)k \cdot e\text{-}\lambda T/k!$, where λ is the average arriving rate. Initially, the arrival of new process is considered to be persistent for simplicity, i.e. λ is fixed. Unless otherwise stated, the process is specified by Cm=20, Co=60 and the deadline is fixed at 80 time units after it is submitted.

The digital PID controller periodically monitors the controlled variable $\mu(t)$ and computes the manipulated variable $\rho(t)$ by following formula:

$$PID(k) = K_P e(k) + K_I T \sum_{j=0}^{k} e(j) + \frac{K_D}{T}\left[e(k) - e(k-1)\right] \tag{2}$$

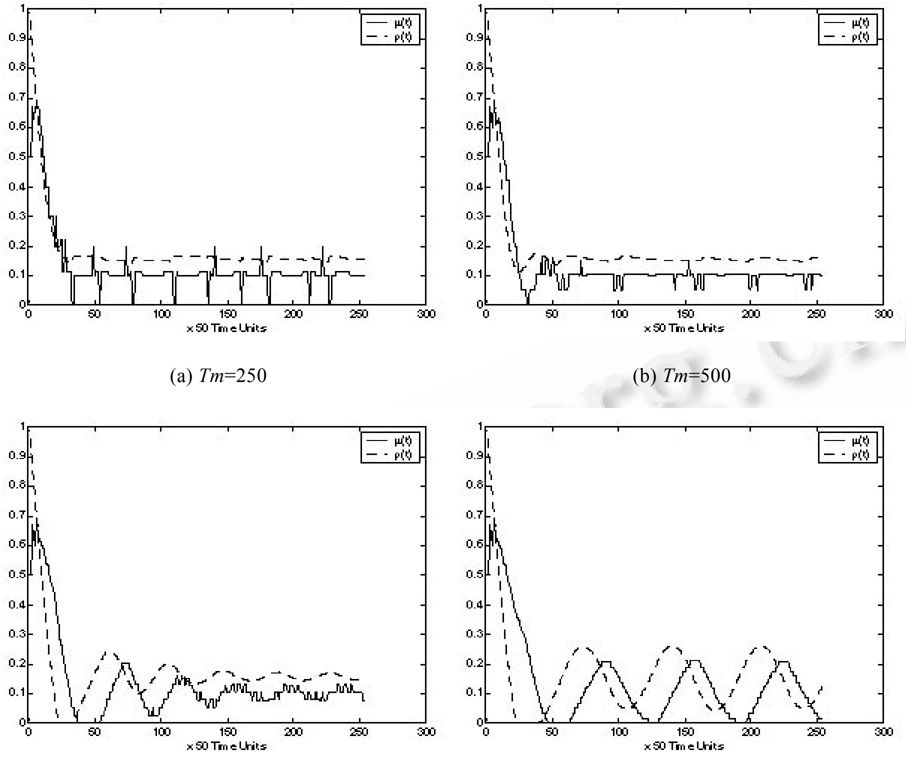
where $PID(k)$ and $e(k)$ are the control and error signals at time $tk=k \cdot SP$ (SP is the sampling period).

The transient behavior analysis of the feedback scheme focuses on Tm, the feature of processes and the sampling period. The tuning of digital PID controller itself is fully discussed in control theory. Because of the difficulty of obtaining accurate transfer function models for our approach, we use the most common guidelines: Ziegler-Nichols rules to compute the optimum gain values for the controller with ultimate-cycle method[14].

As mentioned by Stankovic[11], the derivative control is not suitable for systems with high noise, so we set KD=0 in all the experiments and experiments are run with KP=0.1 and KI=0.02, which are obtained by the ultimate-cycle method.

4.2 Effect of *Tm*

As we have said in Section 3, Tm is the maximum time for a process to stay in the Missed Process Queue. We estimate the system deadline-missing ratio at time t, i.e. $\mu(t)$, by the average possibility of processes to miss their deadlines during $[t-Tm,t]$. Figure 5 illustrates the system's step responses with Tm=250, 500, 1000 and 1500.



(a) *Tm*=250 (b) *Tm*=500

(c) *Tm*=1000 (d) *Tm*=1500

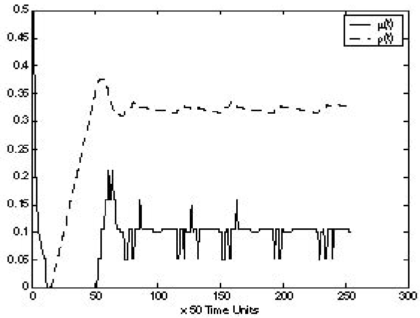
Fig.5 Effect of *Tm*

It is observed that Tm has a great effect on the settling time of system step response. In Fig.5(a), the scheduler with *Tm*=250 achieves a 0.17 steady state $\rho(t)$ after 25 sampling periods while *Tm*=1500, only neutral stability can be achieved. So, a smaller value of Tm makes the scheduler more agile to the change of system load. We also find out that the step response of scheduler with a larger *Tm* is smoother.

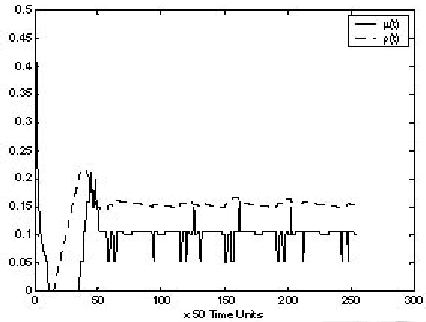
4.3 Effect of *Cm* and *Co*

According to Section 3, the key point of our approach is to execute only part of the *Co* to reduce system load once overload happens. So, processes of a larger *Co* show more potential for adapting to an overload situation. Figure 6 illustrates the scheduler's transient behavior with different *Co* ranging from 10 to 40.

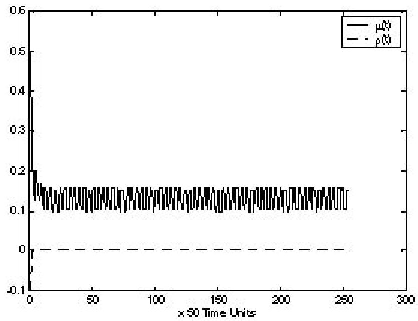
In Fig.6(a), $\mu(t)$ is fixed at the set point, i.e. 0.1, after the system reaches its steady state with $\rho(t)$=0.32. In scenario of Fig.6(b), the system has a higher load (*Cm*=20) and the scheduler can still adjust the total system load to bind $\mu(t)$ to 0.1 by reducing $\rho(t)$ to 0.15. However, when *Cm* increases to 30, the system can only achieve a steady state error of 0.04 with $\rho(t)$ reduced to 0 which is shown in Fig.6(c). With the increasing of system load, the steady state error also increases. So, the ability of scheduler to adjust the system load is limited by the features of processes submitted to the system in our scheme.



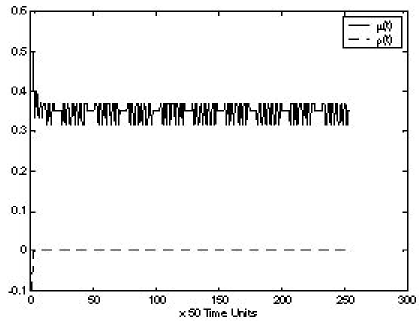
(a) *Cm*=10, *Co*=60



(b) *Cm*=20, *Co*=60



(c) *Cm*=30, *Co*=60

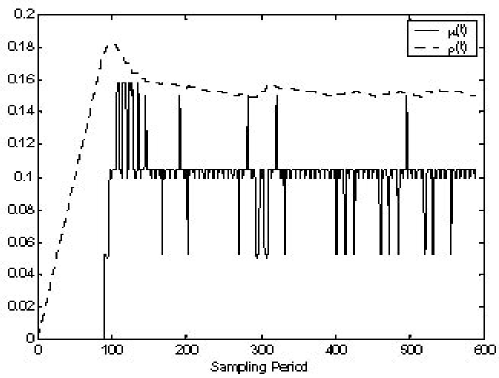


(d) *Cm*=40, *Co*=60

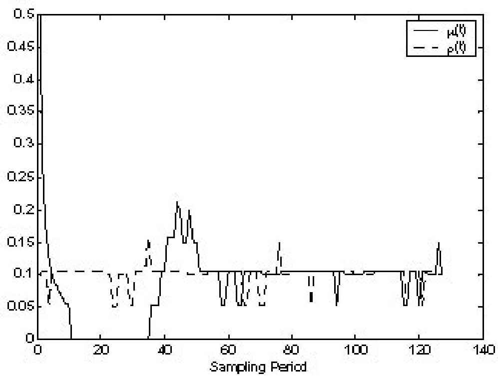
Fig.6 Effect of feature of processes

4.4 Effect of sampling period: *SP*

Digital devices can handle mathematical relations and operations only when expressed as a finite set of numbers rather than as infinite-valued functions. Thus, any continuous measurement signal must be converted into a set of pulses by sampling. Figure 7 illustrates *SP*'s effect on the transient action of the proportional controller with *KP*=0.1. It is observed that the response of the scheduler with a short *SP* is quicker. In terms of control theory, the sampled data system behaves like an analog system at high sampling rates. But a high sampling rate will also introduce a large overhead into the scheduling system.



(a) *SP*=10



(b) *SP*=50

Fig.7 Effect of sampling period: *SP*

5 Conclusions

In this paper, we propose a new feedback-driven approach for online scheduling of processes with imprecise computation. Our approach uses a feedback control, e.g. PID, to determine the percentage of optional part to be admitted when processes are submitted to system. By studying the experiment results, it turns out that the adaptive control is an effective choice for adjusting the precision of each process dynamically to bind the system deadline-missing ratio to the set point under a varying system workload.

References:

[1] Liu JWS, Shih WK, Lin KJ, Bettati R, Chung JY. Imprecise computations. Proc. of the IEEE, 1994,82(1):83~94.

[2] Choi KH, Jung GY, Kim T, Jung SH. Real-Time scheduling algorithm for minimizing maximum weighted error with $O(N\log N+cN)$ complexity. Information Processing Letters, 1998,67(6):311~315.

[3] Mittal A, Manimaran G, Murthy CSR. Integrated dynamic scheduling of hard and QoS degradable real-time tasks in multiprocessor systems. Journal of Systems Architecture, 2000,46(9):793~807.

[4] Castorino A, Ciccarella G. Algorithms for real-time scheduling of error-cumulative tasks based on the imprecise computation approach. Journal of Systems Architecture, 2000,46(7):587~600.

[5] Baruah SK, Hickey ME. Competitive on-line scheduling of imprecise computations. IEEE Trans. on Computers, 1998,47(9):1027 ~1032.

[6] Ho KIJ, Leung JYT, Wei WD. Scheduling imprecise computation tasks with 0/1-constraint. Discrete Applied Mathematics, 1997, 78(1,3):117~132.

[7] Manimaran G, Murthy CSR. A new scheduling approach supporting different fault-tolerant techniques for real-time multiprocessor systems. Journal of Microprocessors and Microsystems, 1997,21(3):163~173.

[8] Dey JK, Kurose J, Towsley D. On-Line scheduling policies for a class of IRIS (increasing reward with increasing service) real-time tasks. IEEE Trans. on Computers, 1996,45(7):802~813.

[9] Chantrapornchai C, Tongsima S, Sha EHM. Imprecise task schedule optimization. In: Proc. of the 6th IEEE Int'l Conf. on Fuzzy Systems. 1997. 1265~1270.

[10] McVey CB, Atkins EM, Durfee EH. Development of iterative real-time scheduler to planner feedback. In: Proc. of the 15th Int'l Joint Conf. on Artificial Intelligence. August 1997. 1267~1272.

[11] Stankovic JA, Lu C, Son SH, Tao G. The case for feedback control real-time scheduling. In: Proc. of the 11th EuroMicro Conf. on Real-Time Systems. York, England, 1999. 11~20.

[12] Steere DC, Goel A, Gruenberg J, McNamee D, Pu C, Walpole J. A feedback-driven proportion allocator for real-rate scheduling. In: Proc. of the 3rd USENIX Symposium on Operating Systems Design and Implementation. New Orleans, Louisiana, 1999. 145~158.

[13] Huang J, Stankovic JA, Towsley D, Ramamritham K. Experimental evaluation of real-time transaction processing. In: Proc. of the 10th IEEE Real-Time Systems Symp. Santa Monica, 1989. 144~153.

[14] Palm WJ. Modeling, Analysis, and Control of Dynamic Systems. Wiley TextBooks, 1998.