Applied Mathematics & Information Sciences 5 (2) (2011), 211S-217S
– An International Journal

Research on Parallel Real-time Scheduling Algorithm of Hybrid Parameter Tasks on Multi-core Platform

Benhai zhou[1] , Jianzhong Qiao[1] and Shukuan Lin[1,2]

[1]College of Information Science and Engineering, Northeastern University, Shenyang，Liaoning Province 110004, China

[2]Key Laboratory of Software System and Development Generic Technology, Shenyang, Liaoning Province, 110004, China

Email Address: linshukuan@mail.neu.edu.cn

Received June 17, 2010; Revised March 4, 2011

Abstract: Nowadays, multi-core processors are becoming main stream in computer market due to their high performance, low cost and less power consumption characteristics. However, multi-core processors give rise to new problems when they are applied to real-time system. Little attention has been focused on tasks' DMR(deadline miss rate) by using previous multi-core real time scheduling algorithms under the overload condition. As a result, the domino effect will happen because of many tasks failing to meet their deadlines. Meanwhile, the system performance is leaded to drop sharply. In order to alleviate this problem, this paper proposes a new multi-core real time scheduling algorithm which extends the Pfair scheduling method using tasks' hybrid parameters. In addition, the paper will also discuss the tasks' allocation method which would decrease the switch cost. Experimental results for schedules demonstrate that our scheme enables the real time tasks to be scheduled more efficiently on multi-core platform by adopting hybrid parameter priority. Furthermore, the system performance has gained the robust characteristic, because more real time tasks can meet their deadline under the overload condition.

Keywords: Multi-core; Schedule; Hybrid parameter; DMR.

1 Introduction

Multi-core processors have been becoming popular due to their high performance, low cost and less power consumption characteristics. This trend already extends to embedded real time system designs with multi-core processors (e.g., for cell phones) which are capable of running one or more software simultaneously [1]. The most modern multi-core architecture comprises processors cores from 0 to N and each core has private L1 cache which consists of instruction cache (I-cache) and data cache (D-cache).

Because of the advantage of multi-core processors, the research of real time system on multi-core processors become growing up as widely studied by universities and institutions. Unfortunately, under the overload condition, the traditional real time scheduling algorithms on multi-core platform, such as EDF and Pfair [2,3], easily lead to more and more tasks missing their deadlines because of delay of previous tasks. As a result, the domino effect will lead to the performance of real time system drop sharply. To solve this problem, this paper proposes a new multi-core parallel real time scheduling algorithm which extends the Pfair scheduling method.

2 Related Work

Choffnes et al. put forth migration policies for multicore fair-share scheduling in the context of soft real-time systems [4]. Their technique minimizes migration costs while ensuring fairness among tasks by maintaining balanced scheduling queues as new tasks are activated. In contrast, our work targets hard real-time systems.

Li et al. present migration policies that facilitate efficient operating system scheduling in asymmetric multicore architectures [5]. Their work focuses on fault-and-migrate techniques to handle resource-related faults in heterogeneous cores and does not operate in the context of real-time systems. In contrast, our work aims at homogeneous cores and improves system performance by migrating tasks. Otherwise, the deadlines of tasks can be guaranteed.

Calandrino et al. propose scheduling techniques that account for common schedulability of tasks with respect to cache behavior [6,7]. They put the cooperating tasks which have the same period into the same group. Otherwise, their work solves the cache performance in soft real-time system. What's more, they do not consider the task migration.

3 Background

Pfair scheduling algorithm is global scheduling algorithm, and allocates processor time one quantum at a time. In Pfair scheduling [8], a time unit is actually a unit of processor allocation, and thus is referred to as a quantum. In EDF [9], a time unit is not necessarily a unit of allocation, but can be any convenient size such that execution costs and periods are integral. The quantity (execute time/ period time) is the utilization of T, denoted wt(T). Although Pfair scheduling has been proved optimal, but it can not work well in overload condition, because its sufficient and necessary condition is that tasks' total utilization, $\sum_{T \in \tau} wt(T)$, does not exceed processor cores M.

The time of quantum length interval [t, t + 1), where $t \geq 0$, is called slot t. In each slot, each task is assigned to one processor core. Task migration is allowed. Per-quantum allocations are achieved by sub-dividing each task T into a sequence of quantum-length subtask. Each subtask Ti has an associated release r(Ti) and deadline d(Ti). The time-slot interval [r(Ti), d(Ti)) is called the window of Ti .

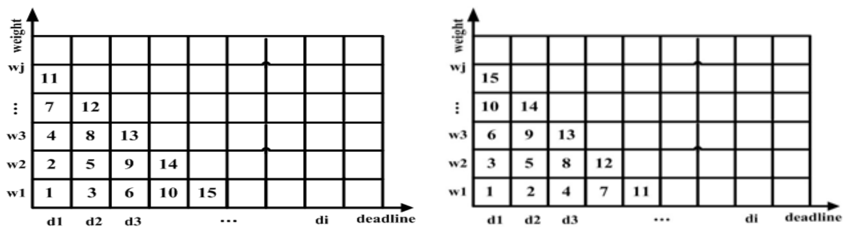
$$r(T_i) = \left\lfloor \frac{i-1}{wt(T)} \right\rfloor \wedge d(T_i) = \left\lceil \frac{i}{wt(T)} \right\rceil \qquad (3.1)$$

4 Scheduling Algorithm

4.1 The Definition of Task Priority

The algorithm utilizes the deadline and weight of the tasks to defining their priorities. Before describing the hybrid parameter scheduling method, we will give definition of tasks and concerned parameters. For task Ti, it concludes parameters as follows: 1) r(Ti)denotes release time of task Ti. That is the time when task is driven and ready for running; 2) T.e denotes execution time of tasks; 3) T.prio denotes the priority of tasks; 4) T.d denotes the absolute deadline of tasks, that is, the task should finish before deadline and generate the valuable result; 5) T.p denotes the period of tasks; 6) T.w denotes the weight of the tasks, value is T.e/T.p.

This paper assigns the task priority based on deadline and weight parameters. According to deadline parameter, task priority is $P(T_{ij}, d_i, w_j) = d_i$; Based on task weight parameter, task priority is $P(T_{ij}, d_i, w_j) = w_j$; Task priority can be confirmed by two dimensions (parameters) including deadline *d_i*, task weight value *w_j* as figure 4.1 shown.



a) The priority is inclined to weight b) The priority is inclined to deadline

Figure 4.1: Task priority is confirmed by deadline and weight parameters

From the figure 4.1, we can see that if only deadline parameter is adopted, the task weight will not affect the priority. The deadline and weight are ordered by ascending. Furthermore, utilizing the two characteristic parameters to consider the priority synthetically, the shorter deadline and the lighter weight is, the higher priority the task has. In this priority table, the priority rank of each task is $rank = d_i + w_j$. Some tasks have the same priority rank. In order to assign unique priority for each task, we can order the priority of tasks in accroding to whether the algorithm is inclined to deadline or weight. The priority can be denoted by the equation (4.1) and (4.2).

$$T._{prio} = \begin{cases} \dfrac{(d_i + w_j - 1)\times(d_i + w_j - 2)}{2} + d_i & \text{if the system is inclined to weight} \\ \dfrac{(d_i + w_j - 1)\times(d_i + w_j - 2)}{2} + w_i & \text{if the system is inclined to deadline} \end{cases} \qquad (4.1)$$

$$p=\begin{cases}(rank-1)(rank-2)/2+i & \text{if the system is inclined to weight}\\(rank-1)(rank-2)/2+j & \text{if the system is inclined to deadline}\end{cases} \tag{4.2}$$

Note that task priority assignment is not a static method. The dynamic priority allocation happens in the run time. We can also enhance the deadline or weight tendency by adding a tendency parameter α . If the system is more inclined to weight, the priority rank of the task can be denoted by $rank=(\alpha*i-\alpha+1)+j$. So, the task's priority can be simply expressed by equation (4.3).

$$p=\left[\alpha*(i-1-\beta)+2j-2\right]*(i+\beta)/2+i \tag{4.3}$$

The β is denoted by $\beta=\left[(j-2)/\alpha\right]$, where α is integer. If the system is more inclined to deadline, the task's priority can be simply expressed by equation (4.4) .

$$p=\left[\alpha*(j-1-\beta)+2i-2\right]*(j+\beta)/2+j \tag{4.4}$$

4.2 The priority policy of extended Pfair Algorithm

In this section, the task priority determination method is applied to Pfair scheduling algorithm. In hybrid parameter real time scheduling algorithm, relative priority of two subtasks is determined by applying the following rules.

In rule 1, if the priority value of subtask Ti is higher than subtask Uj, Ti.prio is less than Uj.prio. When the priority value of task Ti is equal to task Uj, the extended rule 2 continues to compare with the priority value of subtask Ti+1 and subtask Ui+1.

4.3 Tasks Allocation Method

Because our work refers on global scheduling policy, the task migration is necessary. It is important to minimize the communication cost between processor cores. So we should try to keep adjacent tasks in the same core. In this way, when a task passes messages to another task, the state can efficiently be transferred through local registers. Our work puts the high related tasks together on single core in order to reduce the cores' communication. We also adopt task duplication policy for achieving workload balance. We should consider the expected execution time and the frequency of the task reusing.

4.4 The Algorithm Description

The global real time scheduling algorithm of hybrid parameter tasks is described as figure 4.2.

```
Initialization: initial the task list
Step 1: determine the priority
1.static OS_EDF_PRIO ( Task)
2.{order ready tasks by deadline in EDF list;
3.Find the task's position i from EDF list;
4.Return i; }
5.static OS_Weight_PRIO (Task)
6.{order ready tasks by deadline in weight list;
7.Find the task's position j from weight list;
8.Return j; }
9.INT16UPRIO(INT8UOS_EDF_PRIO,
10.OS_Weight_PRIO)
11.{INT8U OS_PRIO;
12.OS_Prank=
OS_EDF_PRIO+OS_Weight_PRIO ;
13.OS_PRIO=(OS_Prank-1)(OS_Prank-2)
/2+OS_EDF_PRIO ;
Return OS_PRIO;//return to OS_Task->PRIO}
Step 2: extend Pfair scheduler
14.Static exPfair_schedululer()
15.{find the task owned minmum of OS_PRIO
running
16.if(OS_Task[i]->PRIO==OS_Task[j]->PRIO)
17.{Jugde the priority of OS_Task[i+1]
and OS_Task[j+1]};
18.the scheduler adopts step 1 to select
highest priority task to run on each
schduling time;}
Step 3: processor's cores allocation
20:{ while ∃ ei j with tj unmapped do
21: find the node connected by the
heaviest utilized edge;
22: if (tasks allocated to(C) )≤ M
23: {m(Ti) ←C //assign task Ti to core C
24: C ←Allocation_task (k,C); //map the
next node}
25: else{m(Ti) ← C + 1 //assign task Ti to
core C + 1
26: C ←Allocation_task (k,C + 1) }
27: return C;)
28: while $\sum_{i=1}^{T} d_i < CORES$
29: { j ← the ID of node with maximum
workload.
30: dj ← dj + 1;}// task duplication policy
```

Figure 4.2. The description of extend Pfair algorithm

5 Experiment and Analysis

To assess the efficacy of our new algorithm, we conducted experiments using the SESC Simulator [10], which is capable of simulating a variety of multi-core architectures. We chose to use a simulator so that we could experiment with systems with more cores than commonly available today.

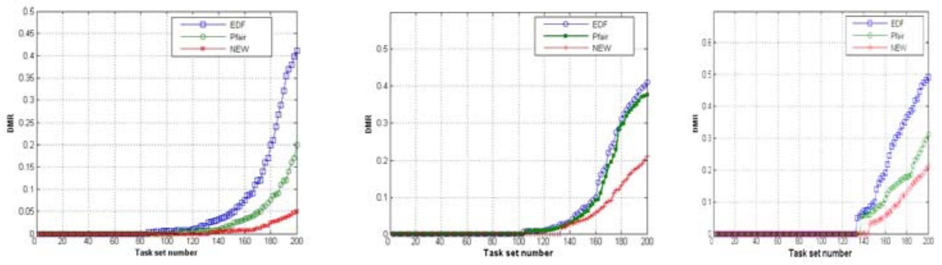
The experiment selects the periodic task sets consisting of 50 and 500 tasks in steps of 25. For each task-set size, we measured ten task sets generated randomly for a total of 200 task sets per scheduling algorithm. The utilizations of Tasks are distributed differently for each experiment using three uniform and three bimodal distributions. The ranges for the uniform distributions were [0.001, 0.1] (light), [0.1, 0.4] (medium), and [0.5, 0.9] (heavy). Similarly, we considered three uniform task period distributions with ranges [3ms; 33ms] (short), [10ms; 100ms] (moderate), and [50ms; 250ms] (long). Note that all periods were chosen to be integral.

5.1 Tasks DMR Comparison

Deadline Missing Rate (DMR) is an important measure value which judges whether real-time performance meets the demand situation or not in practical running. DMR is defined as follows. Suppose that the task number of missing deadline is n and the task number finished normally is m, then DMR can be represented by equation (5.1):

$$DMR = \frac{n}{m+n} \tag{5.1}$$

The 200 task sets are scheduled by using EDF, Pfair and proposed algorithms separately on 2,4 and 8 cores. Figure 5.1 displays the comparison about the DMR of EDF, Pfair and proposed algorithm. From DMR curve, it can be seen clearly that DMR of using extended Pfair algorithm is lower than that adopting EDF and Pfair on 2, 4 and 8 cores.

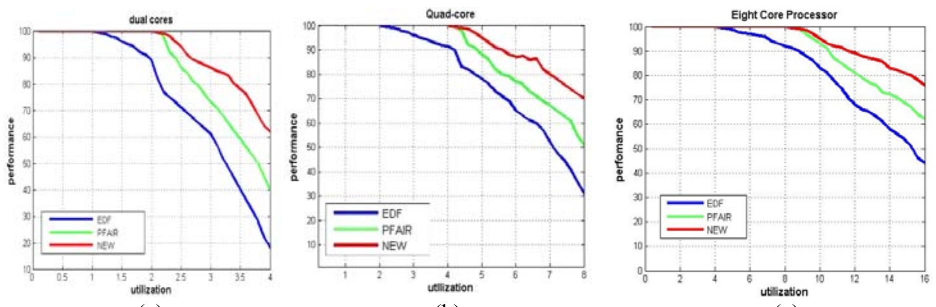


(a) (b) (c)

Figure 5.1. Comparing with DMR of three algorithms on a) 2, b) 4 and c) 8 cores

5.2 Performance Evaluation of real time system

Next, we test the performance of each algorithm when the total tasks' utilization exceed in M (processor cores number). Figure 5.2 shows that the real time system performance curve when tasks are scheduled by EDF, Pfair and proposed algorithms. We can draw the conclusion that EDF can maintain good performance before utlization is less than M/2 (M means number of cores).When utilizaiton exceed M/2, the performance drops sharply. Unfortunately, Pfair scheduling algorithm represents the low performance when utilizaiton exceed M. However, the proposed algorithm achieves the better performance than other algorithms when the system under the overload conditon.



(a) (b) (c)

Figure 5.2: Comparing with performance of three algorithms on a) 2, b) 4 and c) 8 cores

6 Conclusion

This paper analyzes and designs the global hybird real time scheduling method which adopts synthesis parameter judging priority. For DMR is an important measure value to estimate real-time performance of the system, this paper compares the DMR of using EDF, Pfair and proposed algorithm respectively on 2, 4 and 8 cores. The experimental results indicate that DMR is the lowest when scheduler adopts the global real time scheduling algorithm of hybird parameter tasks. Meanwhile, we compared with performance of three algorithms when total utilizaitons increase. The results shows that our scheduling algorithm represent the robust characteristic in overload system. As a result, the global real time scheduling algorithm of hybird parameter tasks s can improve real time system performance effectively in multi-core environment.

References

[1] Tera-scale research prototype: Connecting 80 simple sores on a single test chip. ftp:// download.intel.com/research/platform/terascale/terascaleresearchprototypebackgrounder.pdf.

[2] JR. Haritsa, M. Livny and MJ.Carey, *Earliest deadline scheduling for real-time database systems*, in: Proceedings of the 12th IEEE Real-Time Systems Symposium, IEEE press, Los Alamitos, 1991, 232-243.

[3] ED. Jensen, CD.Locke and H .Toduda, *A time-driven scheduling model for real-time operating systems*, in: Proceedings of the IEEE Real-Time Systems Symposium, IEEE press, Washington, 1985, 112-122.

[4] D. Choffnes, M. Astley, and M. J. Ward, Migration policies for multi-core fair-share scheduling, *ACM SIGOPS Operating Systems Review*, 42(2008):92–93.

[5] T. Li, D. Baumberger, D. Koufaty, and S. Hahn, Efficient operating system scheduling for performance-asymmetric multicore architectures, in: *ACM/IEEE Conference on Supercomputing*, New York, ACM press, 2007, 90-101.

[6] J. Anderson, J. Calandrino, Real-time scheduling on multicore platforms, in: *Proceedings of IEEE Real-Time Embedded Technology and Applications Symposium*, IEEE press, San Jose, 2006, 179–190.

[7] J. Calandrino and J. Anderson, Cache-aware real-time scheduling on multicore platforms: Heuristics and a case study, In: *Euromicro Conference on Real-Time Systems*, IEEE press, Prague, 2008,209–308.

[8] A. Srinivasan and J. Anderson, Optimal rate-based scheduling on multiprocessors. *Journal of Computer and System Sciences*, 72(6)(2006), 1094-1117.

[9] P. Valente and G. Lipari, An upper bound to the lateness of soft real-time tasks scheduled by EDF on multiprocessors, in: *Proceedings of the 26th IEEE Real-Time Systems Symp.*, IEEE press, Miami, 2005, 311-320.

[10] J. Renau. SESC website. http://sesc.sourceforge.net.

Benhai Zhou is a Ph.D student at the college of information science and engineering, Northeastern University, Shenyang, China. He received BS degree in computer science and technology from Shenyang University in 2004. In 2007, he earned M.S. Degree in computer software and theory from Shenyang University of Technology in 2007. His research interests are in the areas of computer architecture, and distributed systems.