Two-stage scheduling algorithm based on priority table for clusters with inaccurate system parameters

LIU An-feng(刘安丰), CHEN Zhi-gang(陈志刚), XIONG Ce(熊 策)
(School of Information Science and Engineering, Central South University, Changsha 410083, China)

Abstract: A new two-stage soft real-time scheduling algorithm based on priority table was proposed for task dispatch and selection in cluster systems with inaccurate parameters. The inaccurate characteristics of the system were modeled through probability analysis. By taking into account the multiple important system parameters, including task deadline, priority, session integrity and memory access locality, the algorithm is expected to achieve high quality of service. Lots of simulation results collected under different load conditions demonstrate that the algorithm can not only effectively overcome the inaccuracy of the system state, but also optimize the task rejected ratio, value realized ratio, differentiated service guaranteed ratio, and session integrity ensured ratio with the average improvement of 3.5%, 5.8%, 7.6% and 5.5%, respectively. Compared with many existing schemes that cannot deal with the inaccurate parameters of the system, the proposed scheme can achieve the best system performance by carefully adjusting scheduling probability. The algorithm is expected to be promising in systems with soft real-time scheduling requirement such as E-commerce applications.

Key words: task scheduling; priority table; inaccurate information; quality of service(QoS)

1 INTRODUCTION

Conducting research on scheduling of Web cluster is quite valuable and has already attracted much research work[1-6], but the above algorithms have the following deficiencies.

1) The system status discussed in previous work is always assumed to be definite and accurate. In fact, the execution status of cluster system is always inaccurate due to some facts.

2) Web-based service has its own characteristics, for example, the integrity of sessions must be maintained, otherwise the interrupt of any task in a transaction will make the finished tasks futile[2].

3) Most algorithms do not take memory access locality into account.

In order to overcome the above shortcomings, in this paper, a new priority-table-based two-stage soft real-time scheduling algorithm was proposed for task dispatch and selection in cluster systems with inaccurate parameters.

2 TASK MODEL AND TARGET

The definition of the task model and system parameters is as follows.

Given $n$ independent tasks, $T = \{T_1, T_2, \cdots, T_n\}$ is a set of tasks, and there are $m$ servers and $M = \{M_1, M_2, \cdots, M_m\}$ is a set of servers. Task $T_i$ contains the following structure:

```c
typedef struct TaskNode {
    int Ti; // task ID
    int a; // task arrival time
    int c; // task execution time
    int d; // task absolute deadline
    int v; // task value
    int target; // task access target
    int session; // sessionID to decide whether the different tasks belong to the same session
    struct TaskNode *pdnext; // pointer to the next TaskNode
} TaskNode, *NodeLinkList; // a linked list with the TaskNode as its element
```

Only those tasks with definite deadline are considered in the following discussion of this paper. Whenever a task misses its deadline, it will be terminated with its value decreasing to zero[3].

The formal description of the scheduling algorithm and its target is given below:

1) Task dispatch: dispatching the tasks from a task set $T = \{T_1, T_2, \cdots, T_n\}$ to $m$ real servers.

2) Task selection: scheduling those tasks dispatched to a real server for processing in some order, with a scheduling task result set, denoted as $J$. $J$ should meet the following target that the
scheduling probability of each task in set $J$ is greater than $S_p$, and the total value produced by $J$ approaches the maximum, as described below:

$$\forall P_i(T_i) \geq S_p \quad (T_i \in J) \quad (1)$$

$$\sum_{i=1}^{n} T_i \cdot v \rightarrow \max \quad (T_i \in J) \quad (2)$$

3 TASK SCHEDULING ALGORITHM

3.1 Description of system parameters

The inaccuracy of system status is mainly reflected by the imprecision of the task processing time. However, the task execution time in a period often statistically follows some distribution. To be general, the task execution time in discussion is assumed to observe the lognormal distribution. The analysis result can be easily generalized to other distributions.

A given task set $T = \{T_1, T_2, \ldots, T_n\}$ contains the following parameters:

1) $C(i)$ represents the processing time of task $T_i$.

2) $AC(i)$ represents the maximal variation of processing time for task $T_i$. Modern research demonstrates that network traffic appears to be approximately self-similar, which means that the processing time of tasks over a period is interrelated. The formula proposed in Ref. [6] for estimating $AC(i)$ is used:

$$AC(i) = \Delta C(i-1) + f(a, q) \cdot AU(i) \quad (3)$$

where $\Delta U(i)$ represents the mean of processing time variation of the similar tasks that have been finished, $\Delta C(i-1)$ represents the maximal variation of processing time of the prior task, factor $f(a, q)$ is the forgetting rate on history information $AU(i)$.

Let $C_k = \sum_{i=1}^{k} C(i)$ ($1 \leq i \leq k \leq n$) be the total execution time of tasks from the head to the $k$th in the task scheduling queue $Q$, $n$ be the number of all tasks in the queue. Let $\Delta C_k = \sum_{i=1}^{k} \Delta C(i)$ ($1 \leq i \leq k \leq n$) be the sum of maximal variation of processing time for tasks from the first to the $k$th in the scheduling queue. As to the $k$th task $T_k$ in the queue $Q$, its probability of being scheduled is a random variable. Assume processing time of task $T_k$ and those preceding it in the task queue follow the lognormal distribution with parameters $(\mu_k, \sigma_k^2)$, where parameters $\mu_k$ and $\sigma_k^2$ are determined as follows:

$$\mu_k = C_k, \text{ and according to the principle } "3\sigma",\n\sigma_k^2 = |\Delta C_k/3|^2, 1 \leq i \leq n.$$  

Then the scheduling probability of task $T_k$, $P_s(d \leq d_k)$, where $d_k$ is the deadline, is as follows:

$$P_s(d < d_k) = \begin{cases} \frac{1}{\sqrt{2\pi}\sigma} \exp \left(\frac{-(x-\mu)^2}{2\sigma^2}\right)dx, & \mu - 3\sigma < d_k \\ 0, & \mu - 3\sigma > d_k \end{cases}$$

(4)

3.2 Data structures required

The front dispatcher is mainly responsible for performing task dispatch algorithm. The main data structure are designed as follows.

typedef struct computerList {
    int $h_i$ /* total time for processing tasks in the current scheduling queue */
    int cap // server processing capacity;
    computerList * next;
    ... // several other parameters;
} computerLink * computerLinkHead.

Here, each computerList node corresponds to a real server in a Web cluster, with several system parameters and the current status of the real server stored. Meanwhile, let $C_{sw} = \sum_{i=1}^{n} C_i$ represent the total execution time of all tasks in the current system.

As for tasks running on background real servers, priority-table-based scheduling scheme is adopted, which is most widely used in real-time system scheduling based on priority. A queue $Q$ is constructed to represent the task scheduling table, in which the head task is always selected for execution of scheduling tasks. The data structure for a task node is defined as before.

3.3 Algorithm survey

General principles observed by scheduling are as follows.

1) On background servers, with the prediction of satisfying the scheduling probability, the task with earlier deadline has higher priority. As to tasks with the same deadline, those which come earlier have higher priority.

2) Principle of maximizing system value is followed, which means when system fails to fulfill all tasks, tasks with the least value are dropped first.

3) Scheduling of related tasks is considered. At the front dispatcher, the integrity of sessions is guaranteed as possible. At the background real server, a priority accumulation rule is designed. That is to say, for every task in the same session, its value = original value + prior task's value X accumulation factor, that is, $T_i \cdot v = T_i \cdot prior \cdot v \times r$.