2D Approach for CPU Scheduling in Real Time Database System

Mr. Nitesh A Ghodichor, Mr. Sadanand S. Patil

Abstract — A priority table base CPU scheduling algorithm in real-time database (RTDBS) is presented. It is applicable in environments where data requests arrive with different requirements such as real-time deadline, and user priority. Previous work on CPU scheduling is based on traditional CPU scheduler for meeting the real-time deadlines. The general idea is based on modeling the CPU scheduler in RTDBS requests as points in the multi-dimensional space, where each of the dimensions represents one of the parameters i.e., one dimension represents the deadline of task, another represents the criticalness or value and a third dimension represents the priority of the request, etc. The CPU scheduling problem reduces the problem of finding a linear order to scheduling these multidimensional points. Priority table base algorithms are adopted to define a linear order for sorting and scheduling objects that lie in the multi-dimensional space. This generalizes the one dimensional CPU scheduling algorithms (e.g., EDF, MCF, and CDF). Several techniques are presented to show how a CPU scheduler deals with the progressive arrival of task over time. In this dissertation traditional CPU scheduling algorithms are investigated, implemented and compared with priority table base CPU scheduling algorithm in real time database system.

Index Terms — Time database, CPU scheduling, priority Table, Deadline.

I. INTRODUCTION

All Many real-world applications involve time constrained access to data as well as access to data that has temporal validity. For example consider telephone switching systems network management program stock trading managing automated factories and command and control systems. Radar tracking, recognition of objects and determining appropriate response as well as the automatic tracking, directing of objects on a factory floor. All of these involve gathering data from the environment processing of gathered information in the context of information acquired in the past and providing timely response as well as the automatic tracking and directing of objects on a factory floor. For instance, consider recognizing and directing objects moving along a set of conveyor belts on a factory floor. An object’s features are captured by a camera to determine its type and to recognize whether it has any abnormalities. Depending on the observed features the object is directed to the appropriate work cell. In addition the system updates its database with information about the object. The following aspects of this example are noteworthy. First of all features of an object must be collected while the object is still in front of the camera. The collected features apply just to the object in front of the camera i.e. they lose their validity once a different object enters the system. Then the object must be recognized by matching the features against models for different objects stored in a database. This matching has to be completed in time so that the command to direct the object to the appropriate destination can be given before the object reaches the point where it must be directed onto a different conveyor belt that will carry it to its next work cell. The database update must also be completed in time so that the systems attention can move to the next object to be recognized. If for any reason a time-constrained actions is not completed within the time limits alternatives may be possible. Applications such as these introduce the need for Real-time database systems [1]. There is a wide variety of algorithms is available for scheduling the CPU in traditional database systems such algorithms usually emphasize fairness and attempt to balance CPU and I/O bound transactions. These scheduling algorithms are not suitable for real-time transactions. In real time environments transactions should get access to the CPU based on criticalness and deadline not fairness. If the complete data access requirements and timing constraint are known in advance then scheduling can be done through transaction reanalysis. This is because in a RTDBS, transactions should be scheduled according to their criticalness and the tightness of their deadlines, even if this means sacrificing fairness and system throughput. We broadly categorized RTDBS CPU scheduler on the following basis [3]. Preemptive versus Non preemptive, hard versus soft real-time, Dynamic versus static.

II. CPU SCHEDULING

Since no existing operating system meets these requirements, realizing such applications requires traditional operating systems to be extended along several dimensions [2]. An operating system is a program that manages the hardware and software resources of a computer. It is the first thing that is loaded into memory when you turn your computer on. Without the operating system, each programmer would have to create a way in which a program will display text and graphics on the monitor in the beginning programmers needed a way to handle scheduled according to their criticalness and the tightness of their deadlines, even if this means sacrificing fairness and system throughput. We broadly categorized RTDBS CPU scheduler on the following basis [3]. Preemptive versus nonpreemptive,
Hard versus soft realtime, Dynamic versus static. Complex input/output operations. The evolution of computer programs and their complexities required new necessities. Because machines began to become more powerful, the time a program needed to run decreased. In order for a computer to be able to handle multiple applications simultaneously there must be an effective way of using the CPU. Several processes may be running at the same time, so there has to be some kind of order to allow each process to get its share of CPU time. In the figure below, the diagram shows the possible states of a process. When a process is created its state is set to new. Once the process is ready to use the CPU its state is set to ready. It is inserted into the ready queue awaiting its turn to be assigned CPU time so that the CPU's state is set to ready. Once the CPU is available the process next in line in the ready queue is set to running. This means that the process instructions are being executed.

III. REAL-TIME DATABASE

One objective of this study work is to develop and evaluate policies that provide the necessary support for real-time transactions. In this section we explicitly address the problems of CPU scheduling, conflict resolution, and deadlock resolution. The algorithms for transaction wakeup and transaction restart [4]. Traditional databases hereafter referred to simply as databases deal with persistent data. Transactions access this data while maintaining its consistency. Serializabilities the usual correctness criterion associated with transactions. The goal of transaction and query processing approaches adopted in databases is to achieve a good throughput or response time. Real-time systems for the most part deal with temporal data i.e. data that changes over time. Due to the temporal nature of the data and the response time requirements imposed by the environment tasks in real-time systems possess time constraints e.g. periods or deadlines. The resulting important difference is that the goal of realtime systems is to meet the time constraints of the activities [5].

IV. CPU SCHEDULING ALGORITHMS

There are many scheduling algorithms. Typically, a scheduling algorithm assigns priorities to tasks. The priority assignment establishes a partial ordering among tasks. Whenever a scheduling decision is made, the scheduler selects the task(s) with highest priority to use the resource. Preemptive versus nonpreemptive. If the algorithm is preemptive, the task currently using the resource can be replaced by another task (typically of higher priority). Dynamic versus static, in static scheduling algorithms, all tasks and their characteristics are known before scheduling decisions are made. Typically task priorities are assigned before run-time and are not changed. Dynamic scheduling Algorithms allow tasks sets to change and usually allow for task priorities to change. Dynamic scheduling decisions are made at run-time.

V. TYPES OF SCHEDULING ALGORITHM IN REAL-TIMEDATABASE

CPU scheduling is the most significant of all system scheduling in improving the performance of real-time Systems [6, 7]. Conventional scheduling algorithms employed by most operating systems aim at balancing the number of CPU bound and I/O-bound jobs in order to maximize system utilization and throughput, in addition to fairness as a major design issue. On the other hand, real-time tasks need to be scheduled according to their criticalness and timeliness, even if it is at the expense of sacrificing some of the conventional design goals [8]. Therefore, real-time scheduling algorithms establish a form of priority ordering among the various tasks within the system. Priorities are either assigned statically during system design time as a measure of the task's importance to the system, or can be expressed as a function of time and dynamically Evaluated by the scheduler [15]. Such priorities are related to the attributes of the tasks. Since different applications have different attributes and characteristics, different scheduling algorithms also tend to differ in their priority assignment regimes [9]. There are three CPU scheduling algorithm in RTDBS as following:-

A. Earliest Deadline First

As the name implies, this algorithm uses the deadline of a task as its priority. The task with the earliest deadline has the highest priority. The task with the latest deadline has the lowest priority. The following theorem gives the condition under which feasible schedule exists under EDF Theorem, EDF Bound. The advantage to this is that one can achieve a 100% CPU schedule bound and that it supports dynamic priorities. The disadvantage of EDF is there is no way to guarantee which tasks will fail during transient overload [10]. The most common real-time schedulers are Earliest Deadline First (EDF) and Rate Monotonic Analysis, which are also best suited for scheduling multimedia applications but not good enough. In EDF, processes with the earliest deadline are scheduled first (given highest priority). RMA gives priority to the task with highest frequency (shortest period). These schedulers need admission control in order to refuse scheduling request from applications that are not possible to meet; the refused requests must wait for a later time to be scheduled. Operating systems schedulers do not need admission control. Applying EDF and RMA for scheduling tasks requires the tasks to be periodic and processing length to be known and fixed. These scheduling requirements could be known for multimedia applications but not always. However, the requirements are unknown for conventional applications. Page faults, cache misses lead to unknown variable timing costs. In addition, EDF does not support system overload when tasks cannot be properly scheduled [11, 12]. In Real – Time algorithms are intended for real time environment, serving disk request on deadline. Examples here include EDF, Scan-EDF and Priority Scan (Pscan) [13].

B. Most Critical First

Define the critical / important task set as the first N tasks such that worst-case CPU utilization does not exceed 100%. Each task must specify its desired start time, deadline time, and worst-case execution time. If a critical task does not fall
within the critical task set then a period transformation, as with fixed priority or static algorithm, can be used.

Priority = Level of importance

Assign high criticality to all the tasks in the critical set, and low criticality to all the other tasks. Optionally, assign a unique user priority to every task in the system. This is not change during execution. But again there is no way to guarantee which task will fail due to transient overload.

C. Critical and Deadline First

The following type of information about transactions is available to use in CPU scheduling:
1. Timing Constraints E.g. deadlines.
2. Criticalness It measures how critical it is that a transaction meets its deadline.
3. Value function related to a transaction’s criticalness is its value function.

CDF combine the first two EDF and MCF priority scheduling. The assignment of the criticality and user priority and the actions of the EDF scheduler, done at runtime. Select task with highest criticalness. If two or more tasks share the same criticalness, choose the one with the highest dynamic priority. If all parameter are equal, the tasks are served in a first-come, first-served basis. Both algorithms apply EDF for under-load conditions. However, in overload situations, ALG1 switches to MCF, whereas ALG2 switches to another policy that is an artificial combination of EDF and MCF. If scheduling was based on EDF and MCF together at the same time then it is a natural combination. However, since the scheduling decision under an overload is based on MCF first, then on EDF, the two policies are not actually integrated into one measure and therefore we believe it is an artificial combination of the two policies. Combining the deadlines and criticalness together in one policy; e.g., (ALG1 and ALG2) can outperform both deadline based and criticalness-based algorithms [15]. But both schemes are shown in this study not to be very suitable for an RTDB environment due in turn to their relatively high preemption rate, aside from their inferior performance when operating under overload situations [14]. Priority assign to the job is calculated by using given formula:

\[ P = \frac{RD}{V} \]

Advantages of CDF 1) guarantee for tasks will meet deadlines even during transient overloads.
2) Scheduling of CPU is dynamic. 3) Improved performance. 4) Easy detection of failures for rescheduling. But above techniques are not enough for CPU scheduling using their algorithm instead of that it is degrade the performance because of the only one parameter followed by the algorithm that is 2D new approach for the same parameters but in a different manner. Follow it is…

VI. 2D APPROACH FOR CPU SCHEDULING IN RTDBS

A. First Approach

The earlier the deadline of a task is or the larger the value is, the higher the priority of the task is. But above techniques are not enough for CPU scheduling using their algorithm instead of that it is degrade the performance because of the only one parameter followed by the algorithm that is 2D new approach for the same parameters but in a different manner. Follow it is…

Where, i and j denote positions of deadline and value of the task in their corresponding task lists respectively the smaller the value of \( P \) is, the higher the priority level of tasks is. The tasks on an inclined line belong to the same assigned according to the following formula:

\[ P = (i+j) \]

The smaller the value of \( P \) is, the higher the priority of a task. For tasks with the same priority level, this scheme prefers to executing tasks with earlier deadlines this kind of design scheme of priority table on deadline J value results in a scheduling algorithm called as Earliest Deadline first with large Value [16]. Figure 1 (a) [10]

B. Second Approach

Another algorithm can be derived from a variation of the foregoing previous technique in this technique, the functioning opposite direction. Different from previous, this scheme prefers to executing tasks with larger values, and the corresponding scheduling algorithm is called as least Value with Earlier Deadline, whose priority assignment formula as follows:

All Rights Reserved © 2012 IJARCSEE
Both algorithm has improve the performance of CPU of executing task and avoid the allotment same priority to any two task. Figure 1 (b) [10]

\[ p = (i + j - 1) * (i + j - 2) / 2 + i \] [10]

VII. IMPLEMENTATION

To demonstrate the concept of CPU scheduling, five non-preemptive CPU scheduling algorithms such as Earliest Deadline First (EDF), Most Critical Transaction First (MCF), Criticalness and Deadline First (CDF), and diagonal priority base scheduler (i.e. Earliest Deadline with Value First (EDV) and Value First with Earlier Deadline (VED)) are implemented in Matlab with backhand database (oracle).

In order to implement these CPU scheduling algorithms, a form in Matlab is designed using GUI, where user can select single or all of the above scheduler from the existing given five CPU scheduler. A task set consists of 500 tasks ti (i= 1, 2, 3... 500), and their parameters are generated according to following methods:

- Execution time d will be chosen as a random number between 1 and 3 time units for each task.
- Value of a task / importance’s is modeled as a random number from 1 to 7 which are divided into 7 criticalness levels from 1 to 7.
- Maximum window / buffer size is 7, will be chosen as a random number between 1 and 7.
- On basis of above parameter the priority of the task is set for execution.

Output grid shows the sequences of execution of database task with start time, Finish time, Execution time / Service Time (Ts), Turnaround (Tq) time, tq/ts and status of execution i.e. Successful or miss deadline.

This work demonstrates the feasibility of converting non-real-time DBMS into real-time DBMSs for soft deadline transactions. There are several issues in building the real-time database system from conventional non-real-time DBMSs, and then scheduling the CPU for incoming task. Deadlines associated with task differentiate a real-time database system from a normal database system. The SQL interface must be extended to specify the deadlines. The unit of the real-time database system, which recognizes the task, assigns priorities to task based on their deadlines. The goal of task and query processing in real-time databases is to maximize the number of successful transactions in the system. The different modules and algorithms that are considered in the design phase are explained below:

- Selection of CPU scheduling algorithm
- Assign priorities to the task according to CPU scheduling algorithm
- Execute query by applying scheduling algorithm.
- Show result / generating statistics
- Exit Module

There are five modules in the design of system and each one is explored in detail with the necessary data structure and the algorithm below.

A. Selection of CPU Scheduling Algorithm

CPU Scheduling algorithm module displaying different CPU Scheduling algorithm such as Earliest Deadline First (EDF), Most Critical Transaction First (MCF), Criticalness and Deadline First (CDF), and diagonal priority base scheduler (i.e. Earliest Deadline with Value First (EDV) and Value First with Earlier Deadline (VED)) User can select Individual as well as all above mode in which all above scheduling algorithm executed one by one. User can also select total number of task for execution .A task set consists of 500 tasks ti (i= 1, 2, 3... 500)

B. Assign priorities to the task according to CPU Scheduling algorithm

Priorities assignment to task is performed according to following Algorithm

\[
\begin{align*}
\text{If (algo = Earliest Deadline First (EDF))} & \\
\text{Assign priority to each task (According to dead line of task)} & \\
\text{If (algo = Most Critical First (MCF))} & \\
\text{Assign priority to each task (According to Importance’s of task)} & \\
\text{If (algo = Criticalness and deadline first (CDF))} & \\
\text{Assign priority to each task (According to (deadline / Importunes) of task)} & \\
\text{If (algo = Earliest Deadline with Value First (EDV))} & \\
\text{Assign priority to each task (According to (p = (i + j - 1) * (i + j - 2) / 2 + j))} & \\
\text{If (algo = Value First with Earlier Deadline (VED))} & \\
\text{Assign priority to each task (According to (p = (i + j - 1) * (i + j - 2) / 2 + i))} & \\
\end{align*}
\]
C. Executing Query by Applying Scheduling Algorithm

Once the task arrives through related modules different data required to find out deadlines can be achieved. In the context of real-time database systems, task are assumed to have deadlines associated with them. A task is said to be completed successfully only if it has finished executing within deadline. In a RTDBS, the priority assignment scheme used to schedule task often plays an important role in deciding the effectiveness of the system. This effectiveness is measured in terms of the percentage of transaction completed within their deadlines. This policy does not look at the expected processing time for task and hence has been used to schedule task of soft deadline based application. Another important factor in above CPU scheduling algorithms is that it does not allow preemption of task. However, a task can be blocked until their requested locks are granted. A higher priority transaction that is waiting for data objects may have to be processed after one with a lower priority, which has all the required data cached at the client. In this module, soft deadlines are used; therefore the commit / Rollback decision is to be taken at the end of execution.

D. Show Result / Generating statistics

In this module different statistic results are generated on the basis of the performance of each CPU Scheduler will be measured according to the following metrics:

- Turnaround Time (Tq), turnaround time, total time in system, waiting plus execution.
- Tq/Ts: is the normalized turnaround time. This value indicates the relative delay experienced by a process. Typically, the longer the process execution time, the greater the absolute amount of delay that can be tolerated. The minimum possible value of this ratio is 1.0; increasing values correspond to a decreasing level of service.
- Value-sum / Hit Value Ratio %, the percentage of value that the system was able to collect, relative to the total value of all transactions, those that were completed and those that were aborted. Thus,

\[ \text{Value-Sum} \% = \frac{\text{total value collected} \times 100}{\text{total value of all transactions submitted to the system}} \]

- Success %, the percentage of transactions that are able to complete, whether tardy or on time, relative to the total number of transactions submitted to the system. Thus,

\[ \text{Success} \% = \frac{\text{total transactions completed} \times 100}{\text{total transactions submitted to the system}} \]

- Tardy %, the percentage of completed transactions that is tardy, relative to the total number of transactions that are completed. Thus,

\[ \text{Tardy} \% = \frac{\text{total number of tardy completed transactions} \times 100}{\text{total number of transactions completed}} \]

Finally by comparing the behaviors and performances of these algorithms, the graph are plotted for all five scheduler on the basis of

- Turnaround Time (tq)
VIII. FUTURE SCOPE

The diagonal priority table design can also be extended to integrate three characteristic of tasks such as deadline, slack time and value. This work can be extended to process multimedia based data such as in Video based data over Internet. In soft / firm real-time application such as Ecommerce, ATM network the performance can be improved using diagonal priority table base scheduling concept.

IX. CONCLUSION

The real-time database model on conventional database system has been simulated here. Five Non Preemptive CPU scheduling algorithms out of which three are conventional CPU scheduling algorithms such as Earliest Deadline First (EDF), Most Critical Transaction First (MCF), Criticalness and Deadline First (CDF), and two are diagonal priority table base scheduling algorithms such as Earliest Deadline with Value First (EDV) and Value First with Earlier Deadline (VED) are implemented and compared with each other. The result of comparison shows that Diagonal Priority table design for CPU scheduling algorithm in RTDBS for serving requests which uses parameters i.e. one dimension represents the deadline of task , another represents the criticalness or value and a third dimension represents the priority of the request, etc. This kind of design method of priority table can be used to integrate different characteristics of task. The idea is to map the multiple parameters into the 1-dimensional space. Moreover, an earliest deadline first (EDF) algorithm is used as a baseline. Especially, under overload situation, these algorithms degrade graceful. Comparatively the performances of Earliest Deadline with Value First (EDV) and Value First with Earlier Deadline (VED) are better than conventional CPU scheduling algorithms. Successfully tested in client – server environment of oracle and getting the comparison of all cpu scheduling algorithm.

REFERENCES

[5] Dept. of Computer and Information Science, University of Massachusetts, Amherst, Mass. February Invited Paper
Figure 3. (a) Comparison of CPU Scheduling algorithms for HVR%  

Figure 3 (b) Comparison of CPU Scheduling algorithms for Success% vs. Failure %  

Figure 4. (a) Comparison of CPU Scheduling algorithms for tardy %  

Figure 4. (b) CPU Scheduling in RTDBS Main Menu  

Figure 5. (a) CPU Scheduling in RTDBS selection of number of task for scheduling  

Figure 5. (b) CPU Scheduling in RTDBS task using CDF algorithms  


Mr. Sadanand S. Patil, Asst. Professor in Computer Technology, B.E. (IT) ME (CSE) since seven year in teaching field proficiency in Computer Network & Database, Published 6 paper National & International Conferences