Automated Feedback System for Multimedia in Multiprocessors

Jestin Rajamony, Dr.K.Ramar, K.P.Ajitha Gladis

Abstract

In spite of many real times scheduling algorithms available it is not clear that these scheduling algorithms support fully the problems in the real time system in multiprocessors. There are certain "open loop" algorithm that can support only some set of characteristics such as the deadlines, precedence constraints, shared resources and future release time etc. Open loop are being referred as once the schedules are fixed there is no alterations based on the continuous feedback. But open loop is fine for the static or dynamic models where the job is perfectly modeled and assigned. But when it is executed for unpredictable dynamic systems the open loop does not offer its full performance. This paper fully focuses on the "closed loop" real time scheduling algorithm for multiprocessors in the usage of multimedia systems. Here the case is studied from the worst case to the best case.

I. INTRODUCTION

Despite there are many real time scheduling algorithms available it is not clear that these scheduling algorithms support fully the problems in the real time system in a multiprocessors for the multimedia systems. There are certain "open loop" algorithms that can support only some set of characteristics such as the deadlines, precedence constraints, shared resources etc. For example the Rate Monotonic Algorithm (RMA) is one of the static scheduling algorithms that have the complete knowledge of the task set and its constraints. But the dynamic scheduling algorithm does not have a complete knowledge of the task set and its constraints. For example, if a new task is in urgent and wants to be inserted in the middle of the scheduling then the dynamic scheduler will not know of the current task and its timing. Earliest Deadline First Algorithm (EDFA) [7] is dynamic scheduling algorithm that has the complete knowledge of the task set or timing constraints for the resource sufficient environment. Resource sufficient environment is one where the systems
have the system resources in prior to the task that arrives dynamically at any time and are subjected to the scheduling. If the resource is insufficient in the environment then the EDFA performance will rapidly degrade in overload situations.

The spring scheduling algorithm [9] [11] and RED algorithm [3] also support the dynamic scheduling for a set of characteristic but they are all "open loop" scheduling algorithm. But open loop scheduling is fair for both static and dynamic systems if there is sufficient resource in a stand-alone system. In the case of multiprocessors the workload may differ from one way or the other which tends to be unpredictable for the dynamic systems. Many real world complex problems occur in the multiprocessors. For example, a system in the node of the multiprocessors may meet the different variation in the overload of the execution as in the case of multiprocessors in computers controlling the spacecraft the workload parameters that differ due to different input from the space sensors and their interpretation.

Feedback control theory has been generally applied in the field of mechanical and in the electrical systems. Now, it is applied in the field of computer systems where the use of the adaptive real time system is facing many research challenges. Researches are carried out in different places and the research is also taken to be a challenge and some of them are answered here. Almost most of the early works on the real time system are concerned with the complete avoidance of the undesirable effects such as the overloading of the task. But there are still two questions to be solved. It is difficult to predict the complete requirement of real time system in an unpredictable environment. How to design a scheduling algorithm to satisfy the complete system requirement in an unpredictable environment?

There are some environments where the performance specifications are known early. These environments are called predictable environment. But in a real world most of the performances specifications are not known priory. This type of environment are called unpredictable environment. It is very difficult to satisfy the requirements in an unpredictable environment. Most of the applications in the recent years are towards open and unpredictable environment. And when it is applied to the hard real time application in an unpredictable environment then if the performance requirement is not satisfied this may lead to mission failure or tragedy.

Now a day the soft real time application is growing rapidly in an open and unpredictable environment. Soft real time application is less restrictive type where a critical real time task gets priority over other tasks and can retain its priority until the task is completed. For example in the multimedia, virtual reality applications where the response time is not much required it is used. It has limited utility on the industrial control systems in the robots. In the online banking neither the resource requirement nor the arrival rate of the service request is known priory. But the performance guarantee are required even for these type of applications since this may lead to loss of customers, and can cause financial damages that can lead to mission failure. For these types of applications a well defined scheduler is very essential in the real world. In the case of the multimedia operating in the multiprocessors the images to be obtained should be perfect since it can be now a day in the 3D fashion too. So the images obtained should be in time and the images should be also the correct instead of any loss of data. Three important levels of scheduling are considered namely, Job scheduling that determines which jobs shall be allowed to compete actively for the resources of the system. This is sometimes called admission scheduling because it determines which jobs gain admission to the system. Once admitted, jobs become processes or groups of processes.

Intermediate level scheduling determines which processes shall be allowed to compete for the CPU. This responds to short-term fluctuations in system load by temporarily suspending and activation processes to achieve smooth system operation and to help realize certain system wide performance goals. Thus the intermediate-level scheduler acts as a buffer between the admission of jobs to the system and the assigning of the CPU to these jobs.

Low level scheduling that determines which ready process will be assigned the CPU when it next becomes available, and actually assigns the CPU to this process. It is performed by the dispatcher which operates many times per second. The dispatcher must therefore reside at all times in primary storage.

Even though the open loop scheduler spring scheduling algorithms are designed for the worst-case workload parameter they are underutilized system for workload models that are not available. The problem here is that scheduling paradigms all assume the timing requirements are to be known and also to be fixed. If there is a fixed time range for the scheduling then in the case of multiprocessors it will be more tedious because of the varying inputs. For example if an image is obtained from the satellite of a particular object and if it is not reached in time or the transmission is late or some other problem then there is possibility of some problem to occur. So it is better to fix to a range of dead lines for the job to be finished but that becomes too complex.

Due to these problems in this paper a new paradigm for the scheduling as “closed loop” scheduling for a multiprocessors of systems is been introduced. Previously there are many loop of schedules that has not been mentioned here as the key principle. In this the task is known only that time when the processor gets it in the multiprocessors is highlighted. Most of the key performances such as the timing constraint, settling time, and steady state errors are mentioned in the related section of the paper. The rest of the paper is organized as follows. Section 2 discusses about the feedback architecture, Section 3 describes about the functions of the system, Section 4 gives in brief about the experiment, section 5 shows the result and section 6 finishes with the conclusion.

II. FEEDBACK ARCHITECTURE

Our framework architecture for the real time scheduler for an unpredictable environment consist of the Feed back loop with resource. The figure1 explains the component of the closed loop real time feedback scheduler that is monitor, controller, actuator, scheduler and dispatcher in the multiprocessors.

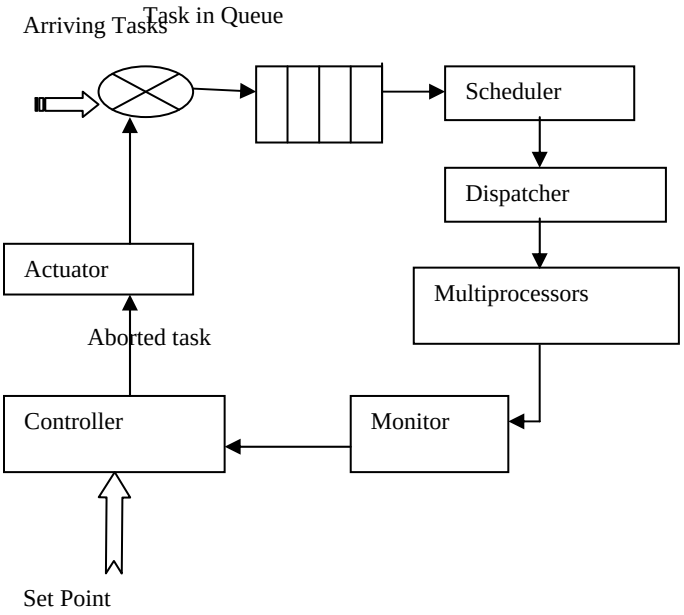

Figure 1. Real Time Feedback Scheduler

These components are been explained in brief.

i. Monitor:

The controlled variables (M (k) Miss Ratio and U (k) Processor utilization) are monitored periodically and then feed a sample back to the controller.

ii. Controller:

The performance references are compared with the corresponding controlled variables to get the current errors, and compute a change $D_B(k)$ (i.e. change in control input) to the total estimated requested utilization based on the error. A control function is used to keep the controlled variable close to the reference with the manipulated variable value in order to compensate the load variations.

iii. QoS Actuator:

According to the change in the control input D (k + 1) that is adjusted by the QoS levels of the tasks the actuator dynamically changes the total estimated requested utilization. The main aim in designing the actuator is to ensure that new total estimated requested utilization B (k+1) = B(k) + $D_B(k)$. A QoS actuator is just an admission controller that uses the QoS optimization algorithm. Here in this paper the QoS actuator is initiated upon the arrival of each task and also isolates the disturbances caused by the variation in the task arrival rates.

iv. Basic Scheduler & Dispatcher:

With the scheduling policy such EDF or RMA the tasks are scheduled and admitted into the multicore heterogeneous processor. In this paper the FCS architecture permits plugging in different polices for the basic scheduler and then designing the entire feedback control scheduling system. The problem that is challenged in this paper is dynamically correct the scheduling errors at run time based on the inputs from the monitor.

v. Multiprocessor.

Since the multicore processor executes at different speed and has different performance if all the processor requires a loop then the speed of monitoring becomes low.

The scheduler receives the tasks in the queue and is based on the algorithm, the task are rescheduled to the dispatcher. The dispatcher finds out the idle Central Processing Unit (CPU) in the multiprocessors and the task is executed there. The aborted task is entered in to the feedback loop and then is identified by the monitor that passes to the controller and then to the actuator.

III. SYSTEM FUNCTION

The performance of the real time system usually depends on the task that make up to the dead line. The percentage of tasks that miss the deadline can be calculated as the system deadline miss ratio. This can be represented as the controlled variable. Controlled variable here is mentioned as the miss ratio $M_R\ (I)$ and the CPU utilization $P_U\ (I)$. This system deadline missed ratio depends highly on the system load. Here the manipulated variable is the total estimated processor utilization for the entire task in the processors of systems.

Suppose if one of the processors input task is interrupted in the middle while on the execution it is monitored by the monitor. The Monitor monitors the variables $M_R\ (I)$ and $P_U\ (I)$ and then sends it to the controller. The controller uses the control function to compute the variable and then compares the current value of the controlled variable with the set point with a referenced tolerance and gets into an error. It then makes $E_R\ (I)$ to the total requested CPU utilization based on the computed errors.

The output is then passed to the actuator. The actuators differ in term of execution time and arrival time and change the manipulated variable to control the system. The actuator then change the total estimated CPU utilization for each sampling instant *I* for the control input $D\ (I+1)$ by adjusting each level of the task that comes in. This now estimates the new total requested CPU utilization $C\ (I+1) = C\ (I) + E_R\ (I)$.

But most of the works or research concentrates on the task that is fixed and is well known. Here in this paper the task are unpredictable and is dynamic and time varying. In the multiprocessors, the systems could get workload where it is shared with initial start as the nominal assumption. The system could then monitor the actual performance of the schedule and compare with the actual requirements and find out the differences. The system will then call the control functions and then apply the correction to keep the system within the acceptable range of the performance.

A. Constraints

To have a closed loop scheduling in the multiprocessors it is necessary to consider the resources and the components in the entire system. The first thing to do is set the set point for the scheduler in the system. The controlled variable has to maintain a very low miss ratio among admitted task. To do this the miss ratio *t* is maintained as a small value but not zero. If in the multiprocessors if one system maintains a miss ratio of 0 then the CPU utilization and the throughput are ignored.

But if the scheduler has a set point with the missed ratio $\neq 0$ it will try to over load any system CPU in the network slightly that is free to achieve high utilization. But in an unpredictable environment of large number of processors or collection of processors in a real time environment it is very difficult to achieve 100% CPU utilization at all the time. Suppose if a high miss ratio occurs the controller will correct the system and make it to low miss ratio if possible to achieve some what high CPU utilization and through put in the network.

Next the CPU utilization of the entire accepted task in the group of system is considered as the manipulated variable. A dispatcher now allocates the system that is idle where the miss ratio $\neq 0$. If the miss ratio = 0 then the processor is in utilization and it can't be overloaded. If needed an admission controller (that can control the flow of workload into the system) and a service controller (that can adjust the work load in the system) can also be included in the scheduler as the mechanism to manipulate the requested utilization. The controller that computes the action of the total amount of the CPU load [5] that need to be added into i.e. $\Delta P_U\ (t) > 0$ or reduced from the system $\Delta P_U\ (t) < 0$.

B. System Utilization

Assume that each processor in the multiprocessors has different workload and each task is independent. Several different forms such as milestone method, sieve function method or multiple version method are imprecise computation [8]. A job with longer execution time and another job with smaller execution time is called. But the task $T_{i\ will}$ have a deadline as D_i and a start time S_i. Each task T_i in the system has the set I, ET, VAL, S, D in it where I represents logical version, ET represents the execution time, VAL represents the values of different types of implementation, S is the start time and D is the deadline. Each task can be adjusted within the range that is specified for given deadline. Each task has one or more logical versions $I = (T_{i1,}\ T_{i2\ \ldots}\ T_{ik).}$ When applied in the multiprocessors these tasks are sent to different CPU so the task does not seams to have multiple implementations. Generally in digital control systems the task timing constraints are allowed to adjust within a specified range without affecting the system stability.

Each task in different CPU has different execution time $ET = \{ET_{11,}\ ET_{12,\ \ldots}\ ET_{kj}\}$ of different versions and they get into different values. This ET specified here is for one CPU. But in different CPU the ET get split to $ET_{21,}\ ET_{22,\ \ldots}\ ET_{kj.}$ This specifies the CPU and the task that is split for that CPU. The nominal execution time is used for the

requested CPU utilization. For example if the $ET_{11} = 0.01$, then the CPU1 utilization is 1% from the miss ratio for the first version. For the same CPU if the $ET_{12} = 0.25$ then the processor utilization is 2.5% for the second version and for the third $ET_{13} = 0.15$ means 1.5% and so on. So the total execution time results in just 0.35 that is about 3.5% CPU utilization.

IV. EXPERIMENT

Consider 5 processors, a monitor, scheduler, actuator, dispatcher and a controller in the system of processors. There is a set point set in the controller that can adjust to a limit range. Assume 10 jobs have arrived in the queue, the scheduler schedules the job in the queue using any of the scheduling algorithms and send it to the dispatcher. The dispatcher then identifies the processor that is idle in the set of processors and sends the task to it. The processor will execute the job if it is a predictable one. Consider if the 2 CPU is given task from the dispatcher and the task is an unpredictable one it is unable to execute the job. Since it is a real time system any inputs from the related sensors may occur. For example consider the workload is interrupted and the monitor monitors the flaw in the workload.

The work of the monitor is to continuously monitor the output of the processor. It is then passed to the controlled unit and the controller compares with the set point along with the prescribed tolerance and sets it to be executed and send to the actuator. The actuators then change the total estimated CPU utilization for each sampling instant I for the control input *D (I +1)* by adjusting each level of the task that comes in. It is again then passed to the scheduler and then to the dispatcher and to any idle processor in the set of processors. If in case the task is totally unpredictable even by the monitor then it is removed from the scheduler and corrective measures are taken. This part is not discussed in depth in this paper.

V. RESULT

From the above experiment it was very clear that the CPU was in full utilization as shown in the figure 2 for a multiprocessor working in the real time environment.

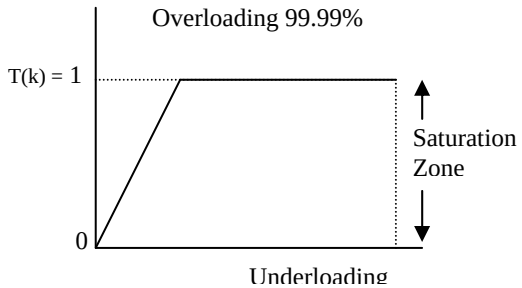

Figure 2. Multiprocessor performance in the real time environment.

VI. CONCLUSION

In this work, the closed loop concepts for a real time scheduling systems in multiprocessors of systems that communicate with one another have been explored. This would give a new idea on the real time scheduling in the field of multiprocessors. Any algorithms used, experiment results that is used for the scheduling are not mentioned. This gives the complete automated feedback loop based on the input that is dynamically changing.

REFERENCES

[1] [Blev 76] P.R. Blevins and C.V.Ramamoorthy, " Aspects of a dynamically adaptive operating systems", IEEE Transactions on Computers, Vol. 25, No.7, pp. 713-725, July 1976.

[2] [Bradt 98] S. Brandt and G.Nutt, "A Dynamic Quality of Service Middleware Agent for Mediating Application Resource Usage ", IEEE Real-Time Systems Symposium, December 1998.

[3] [Butt 95] G. Buttazzo and J.A. Stankovic, " Adding Robustness in Dynamic Preemptive Scheduling" Responsive Computer Systems: Steps Toward Fault – tolerant Real-Time Systems (D.S. Fussell and M. Malek Ed.), Kluwer Academic Publishers, 1995.

[4] [Butt 98] G. Buttazzo, G. Lipari and L.Abeni, " Elastic Task Model for Adaptive Rate Control", IEEE Real-Time Systems Symposium, Madrid, Spain, pp. 286-295, December 1998.

[5] [Chen 98] Chenyang Lu, J.A. Stankovic, Gung Tao, Sang H. Son " Design and Evaluation of a feedback Control EDF Scheduling Algorithm", Department of Computer Science, University of Virgina.

[6] [Hari 91] J.K. Haritsa, M.Livny and M.J.Carey, "Earliest Deadline Scheduling for Real-Time Database Systems", IEEE RTSS, 1991.
[7] [Liu 73] C. L. Liu and J.W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment", JACM, Vol 20, No.1, pp 46-61, 1973.
[8] [Liu 91] J.W.S. Liu.et.al "Algorithms for Scheduling Imprecise Computations", IEEE Computer, Vol 24, No.5, May 1973.
[9] [Rama 84] K.Ramamritham and J.A. Stankovic, "Dynamic task scheduling in distributed hard real time systems", IEEE software, Vol1, No. 3, July 1984.
[10] [Stee 99] D.C. Steere. et. al., " A feedback-driven Proportion Allocator for Real-Rate Scheduling", Operating Systems Design and Implementation, Feb 1999.
[11] [Zhao 87] W. Zhao, K. Ramamritham and J.A. Stankovic, "Preemptive Scheduling Under Time and Resource Constraints", IEEE Transactions on Computers 36(8), 1987.

Jestin Rajamony, is currently working as a Senior Lecturer in the department of Information and Technology, CSI Institute of Technology, Thovalai, Kanyakumari district, TamilNadu, India. Before this he has worked in Software company and steel industry. He has completed his Bachelor degree in Instrumentation and control from Madurai Kamaraj University, Madurai in 1995 and Master degree in Computer Science and engineering from Bharathiar University, Coimbatore in 2000. Currently, he is pursuing his Ph.D degree in Vinayaga Mission University Research Center, Salem,TamilNadu, India. His interest mainly on the parallel computing and distributed computing in real time systems, multimedia.

Dr.K.Ramar, is currently a Principal of Sri vidhya college of Engineering and Technology, Virudhunager, TamilNadu, India. He has been in the teaching and research field for more than 25 years. He has received the Ph.D degree in Computer Science and Engineering from Mononmaniam Sundarnar University, Thirunelveli, TamilNadu,India. He has published many Journals, conferences and guided many Ph.D scholars. His area of interest is distributed computing in real time systems, Image Processing.

K.P.Ajitha Gladis, is currently working as a Assistant Professor and Head of the department of Information and Technology, CSI Institute of Technology, Thovalai, Kanyakumari district, TamilNadu, India. She has completed her Bachelor degree in Computer Science and Engineering from Bharathiar University, Coimbatore in 1995 and Master degree in Computer Science and engineering from Bharathiar University, Coimbatore in 2000. Currently, She is pursuing her Ph.D degree in Vinayaga Mission University Research Center, Salem,TamilNadu, India. Her interest is mainly on the Image Processing, Computer Vision and multimedia.