Fuzzy Real Time Scheduling on Distributed Systems to Meet the Deadline of Applications

Mrs Rekha A Kulkarni , Dr. Suhas H Patil, Dr N.Balaji

Abstract— As there is fast change in the design of computing systems, distributed computing is gaining prominence. Fast processing with large data sets is becoming need of the era. In distributed systems different kinds of hardware and software work together in cooperative fashion to solve problems involving large amount of data. In a heterogeneous environment handling real time tasks poses challenges of processor scheduling and load balancing to meet the deadlines. Implementation of real-time tasks poses a lot of problems due to the unpredictability of the tasks involved and due to the lack of complete task knowledge prior to the execution process. Fuzzy logic can be used in dynamic scheduling of these tasks based on the load information at various processing units. Fuzzy logic uses the system’s deadline miss ratio (DMR) and system throughput to calculate a value which provides a system performance metric used to drive the system to the desired level of performance.

Index Terms— Distributed systems, Fuzzy logic, Dead line miss ratio.

I. INTRODUCTION

Most of the practical real-time scheduling algorithms in heterogeneous systems present a trade-off between their computational complexity and performance. In real-time systems, tasks have to be performed correctly and timely. Finding minimal schedule in multiprocessor systems with real-time constraints is shown to be NP-hard. The practical scheduling algorithms in real-time systems don’t have deterministic response time. Deterministic timing behavior is an important parameter for system robustness analysis. The intrinsic uncertainty in dynamic real-time systems increases the difficulties of scheduling problem. To alleviate these difficulties, A fuzzy scheduling approach can be employed to arrange real-time periodic and non-periodic tasks in systems. Static and dynamic optimal scheduling algorithms fail with non-critical overload. The knowledge based algorithms can be devised to improve the performance of the heterogeneous system.

II. RELATED WORK

Cloud computing provides a distributed computing environment in which there is a pool of virtual, dynamically scale-able heterogeneous computing and storage platforms. The computing power and the storage are provided as Services on demand to external user over the internet. Cloud computing aims at making computing as a service whereby shared resources, software, and information are provided to users who are requesting the service as a utility over a network. One of the key technologies which plays an important role in Cloud data-center is resource scheduling. Real time processing on the cloud systems requires satisfying timing constraints of real time systems. In the scenario of time dependent application deadline misses are undesirable. One of the challenging scheduling problems in Cloud data center is to consider allocation and migration of re-configurable virtual machines (VMs) and integrating features of hosting physical machines.

Earliest Deadline First (EDF)[1] algorithm always chooses the task with the earliest deadline. It has been proved that this algorithm is optimal in a uni-processor system. Since it cannot consider priority and therefore cannot analyze it, this algorithm fails under overloading conditions.

Least Laxity First (LLF) [1] algorithm selects the task that has the lowest laxity among all the ready ones whenever a processor becomes idle, and executes it to completion. This algorithm is non-preemptive and avoids the well-known problem of its preemptive counterpart that sometimes degenerates to a processor-sharing policy.

While using the facilities provided by cloud environment in the area of real time processing the challenge is that of meeting the deadlines. A real-time scheduler must ensure that processes meet deadlines, regardless of system load or inherent delays caused by the various components of the cloud environment. One solution to this is by selecting the scheduling policy at run time. After studying the various real time scheduling algorithms, in this research dynamic selection of scheduling policy based on fuzzy logic has been proposed. This is possible by identifying and using the appropriate parameters in selecting the scheduling policy. In the proposed system we have tried to identify the parameters and the effect of these different parameters on the selection policy. The proposed system will try to improve the performance in real time.

III. PROPOSED SYSTEM

The performance of a scheduling algorithm is measured in terms of additional processor required to be added at a schedule without deadline violations as compared to optimal algorithm it has been proved that finding a minimal schedule for a set of real-time tasks in multiprocessor system is NP-hard.

With the above analysis, the ability to satisfy timing constraints of such real-time applications plays a significant role in distributed environment. However, the existing
Schedulers are not perfectly suitable for real-time tasks, because they lack strict requirement of deadlines. A real-time scheduler must ensure that processes meet deadlines, regardless of system load or inherent delays which are caused by various components of distributed systems.

Our Proposed system S will be represented as

\[ S = \{ I, O, F, Fa, Su \} \]

Where

\[ I = \text{Set of inputs } \{ t_1 \ldots t_n \} \{ p_1 \ldots p_n \} \]

\[ O = \text{Set of output} \] finding optimal mapping of tasks to processors.

\[ F = \{ \text{Set of functions } f_1, f_2, f_3 \ldots \} \]

Where \( f_1 \) is gathering of information about various resources in system. \( F_2 \) generation of scheduling policy based on the parameters collected from \( f_2 \). \( F_3 \) is to decide which is the best policy for given application.

\( Su \): Success case when scheduling policy improves the performance by meeting the dead line of real time task.

\( Fa \): failure case when deadline of time critical applications are not being met.

Fuzzy [2] inference is the process of formulating the mapping from a given input set to an output using fuzzy logic. Laxity is the maximum time that a task can wait before being executed (i.e., laxity = deadline - computation time).

Much of the power of fuzzy logic is derived from its ability to draw conclusion and generate responses based on vague, incomplete, and imprecise qualitative data. The job to be scheduled next according to fairness might not have data on the nodes that are currently free. In a heterogeneous environment like cloud where processing speeds are likely to vary among nodes, high processing nodes are expected to complete more tasks. In such a scenario, resource scheduling plays a critical part in meeting an application’s expectations. In large clusters, tasks complete at such a high rate that resources can be reassigned to new jobs on a timescale much smaller than job duration.

With the help of fuzzy logic it is possible to address the uncertainties involved while dealing with large amount of processors. With the help of Cloud simulators it is possible to create an infrastructure for meeting the deadlines in real time tasks.

**CONCLUSION**

Meeting the deadline for Resource and Timing Constraints is a Challenge in a cloud system. While working with shared environment users will compete for resources to meet their timing deadlines. In such a scenario, resource scheduling plays a critical part in meeting an application’s expectations. In large clusters, tasks complete at such a high rate that resources can be reassigned to new jobs on a timescale much smaller than job duration. The job to be scheduled next according to fairness might not have data on the nodes that are currently free. In a heterogeneous environment like cloud where processing speeds are likely to vary among nodes, high processing nodes are expected to complete more tasks. In such a scenario, the concept of equally distributing data in order to provide data locality is likely to create network bottleneck. For heterogeneous systems, effective data placement strategies are required in order to ensure efficient task scheduling. So the proposed system tries to select a scheduler at run time which can meet the timing requirements based on the information available locally and globally.

With the help of fuzzy logic it is possible to address the uncertainties involved while dealing with large amount of processors. With the help of Cloud simulators it is possible to create an infrastructure for meeting the deadlines in real time tasks.

**REFERENCES**


