Real-Time Performance Estimation for Dynamic, Distributed Real-Time Systems

Eui-Nam Huh[1], Lonnie R. Welch[2], and Y. Mun[3]

[1] Sahmyook University, Department of Computer Science
Chongyang, P.O.Box 118, Seoul, Korea
huh@syu.ac.kr
[2] Ohio University, School of Electrical Engineering & Computer Science
Athens, Ohio, USA
welch@ohiou.edu
[3] Soongsil University, School of Computer Science
Seoul, Korea
mun@computing.ssu.ac.kr

Abstract. The main contribution of this paper is accurate analysis of real-time performance for dynamic real-time applications. A wrong system performance analysis can lead to a catastrophe in a dynamic real-time system. In addition, real-time performance guarantee combined with efficient resource utilization is observed by experiments, while the previous worst-case approaches primarily focused on performance guarantee but resulted in typically poor utilization. The subsequent contribution is schedulability analysis for a feasible allocation of resource management on the Solaris operating system. This is accomplished with a mathematical model and by accurate response time prediction for a periodic, dynamic distributed real-time application.

1 Introduction

The rac-25 radiation machine killed cancer patients because of a software bug. Imagine that you have a car accident. After you have already been thrown into the windshield, the airbag inflates. This paper addresses the problem of certifying that software will respond to real-world events in a timely manner.

Use of real-time systems is being spread rapidly to many areas. Real-time services need to react to dynamic user requests. A dynamic real-time system used to offer services to the dynamic user requests should consider variable execution times and/or arrival rates of tasks during run-time.

One of the management problems of dynamic real-time systems that must be solved is to provide the quality of service (QoS) for a time-constrained task. The task commonly appears in systems such as air traffic control, robotics, automotive safety, mission control, and air defense. An example of this task is software that detects radar data, evaluates it and launches missiles if a hostile missile is detected. All of these dynamic real-time systems require consideration of variable execution times and/or arrival rates of tasks, as opposed to deterministic and stochastic real-time systems which have a priori known, fixed task execution times and arrival rates.

Resource allocation, mapping software (S/W) to hardware (H/W), for those systems is an essential component and needs to consider real-time constraints of the task and should analyze feasible resources. Thus, the feasible allocation of the available resources has to be carried out according to time-constrained requirements, failure of which might cause a catastrophe in hard real-time systems. To maintain the feasible allocation, shedulability analysis of the task is required. Prediction of the response time of the task compared to the time-constrained requirement is one of the schedulability analysis approaches. Furthermore, computing resources for those dynamic systems should be utilized efficiently. The distributed system is employed to provide scalable resources to the dynamic S/W systems that have various and unbounded execution times.

Generally, the real-time system is designed to analyze that a task can meet its time constraint before it is executed. The Rate Monotonic Analysis (RMA) introduced by Liu and Layland in [1] is used primarily to determine schedulability of an application by using a priori Worst-Case Execution Time (WCET) and the priority of the application. The priority of the application to be applied to RMA is dependent upon arrival patterns and rates. The application, which has a higher arrival rate of task or needs to be executed more frequently, has a higher priority level than any other applications.

However, as has been noted in [2], [3], [4], [5], and [6], resources are poorly utilized if the average case is significantly less than the worst case. Another drawback of RMA is that it cannot efficiently accommodate high-priority jobs that have relatively low rates. It must, however, be noted that RMA can be made to work in such cases, by transforming low-rate, high-priority jobs into high-rate jobs -- but this can be extremely wasteful in terms of resources.

It is stated in [6], [7] and [13] that accurately measuring the WCET is often difficult, and is sometimes impossible. Puschner and Burns in [8] consider WCET analysis to be hardware dependent, making it an expensive operation on distributed heterogeneous clusters.

The statistical RMA by Atlas and Bestavros in [9] considers tasks that have variable execution times and allocate resources to handle the expected case. The benefit of this approach is the efficient utilization of resources. However, there are shortcomings. Firstly, applications which have a wide variance in resource requirements cannot be characterized accurately by a time-invariant statistical distribution; and secondly, deadline violations occur when the expected case is less than the actual case. Similarly, real-time queuing theory by Lehoczky in [5] uses probabilistic event and data arrival rates for performing resource allocation analysis. On the average, this approach provides good utilization of resources. It must be noted, however, that applications which have a wide variance of resource requirements cannot be characterized accurately by a time-invariant statistical distribution. Called the dynamic real-time system by Welch and Masters in [10], there is a need for a new approach to the dynamic real-time system which would be more efficient than RMA and statistical RMA in assurance of the real-time QoS.

2 Problem Statement

This section uses mathematical notation to describe the problem of certifying that a real-time application meets (or does not meet) its real-time requirement on a host that is running other applications. The notation will be used in subsequent sections to concisely define certification methods. For convenience, the Data Flow Diagram (DFD) is used for description of problems.

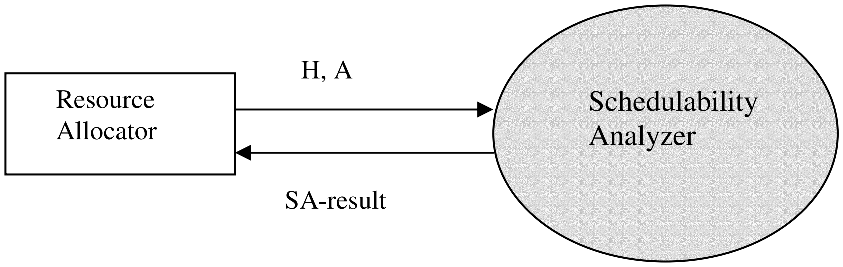

Fig. 1. Level 0 DFD of "Schedulabilty Analyzer System"

As shown in Fig.1, the Schedulability Analyzer certifies (or says that it cannot certify) that the application ('A') will meet its real-time requirement on the host (H). The application 'A' is represented as period, priority, execution time, segment and real-time requirement. The host 'H' is represented as its name, time quanta, and a list of applications. SA-result is returned to indicate the certification result, which consists of a boolean value and a predicted response time.

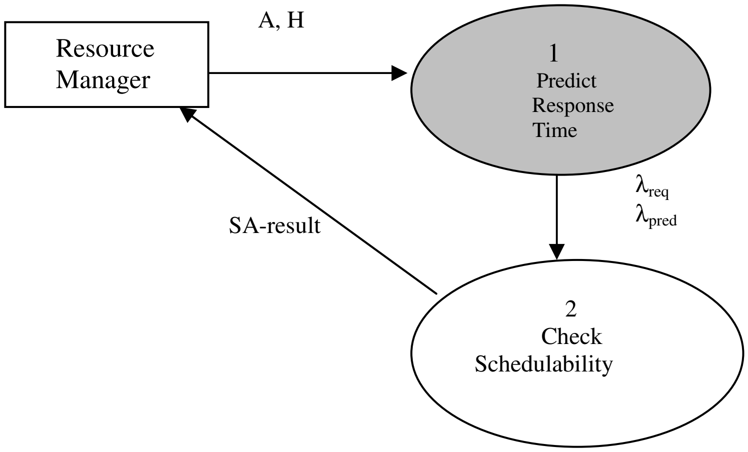

Fig. 2. Level 1 DFD of "Schedulability Analyzer"

The Schedulability Analyzer from Fig. 1 is depicted in Fig. 2. This paper presents a novel approach to schedulability analysis, which predicts response time (λ_{pred}) of 'A' and checks schedulability to certify that 'A' will meet its real-time requirement (λ_{req}) on H before allocation. 'Predict Response Time' reads application list ('A*') on H and computes λ_{pred} of 'A'. 'Check Schedulability' compares λ_{pred} with λ_{req} and returns 'true' if $\lambda_{pred} < \lambda_{req}$ and returns 'false' otherwise; the predicted response time; λ_{pred}, is also returned. 'A', H and the elements of A* are defined as tuples below.

Definition 2.1: An application tuple A = (T, p, *C, U, S,* λ_{req} *)*
where

T is period of A ,
p is priority of A,
C is execution time of A,
U is utilization of A ,
S is the number of time quanta used by A per period.
(it is computed as C/TQ .)
λ_{req} is the requirement of A. (note: we assume that $\lambda_{req} \leq$ T)

Definition 2.2: A host tuple H=(name, TQ, A*)
where

name is name of a host,
TQ is a vector defining the time quantum for each priority on a host
A* is a list of the applications:<a_1, a_2,,, a_n> that are currently allocated to H.

Definition 2.3: Each element of A* is an application, represented as a tuple = (T, p, *C, U, S)*:
where

T is period of A ,
p is priority of A,
C is execution time of A,
U is utilization of A , and
S is the number of time quanta used by A per period.

The convention used in this paper to denote an element 'E' of tuple 'T' is E(T). Thus, for example, the execution time of an application 'A' is denoted as C(A).

The novel approach to schedulability analysis, prediction of response time, from Fig. 2 is described in Fig. 3 in detail. The prediction of response time technique considers estimation of queuing delay of 'A' due to contention with other applications ('A*') on H. An important consideration of prediction of response time is estimation of queuing delay due to the same priority (p) applications (D_{pred1}) as shown at 1.1 bubble in Fig.3. Queuing delay due to higher priority (p) applications (D_{pred2}) as shown as bubble 1.2 in Fig. 3 is also estimated as higher priority tasks always hold resources of H. Finally, Calculate response time of the application as shown as bubble

1.3 in Fig. 3 computes the estimated response time of 'A': $\lambda_{pred} = C + D_{pred1} + D_{pred2}$, where D_{pred1}: delay experienced by a due to waiting for $a_i \in$ 'A*' such that $p(a_i)=p(A)$, and D_{pred2}: delay experienced by waiting for $a_j \in$ 'A*' such that $p(a_j) > p(A)$.

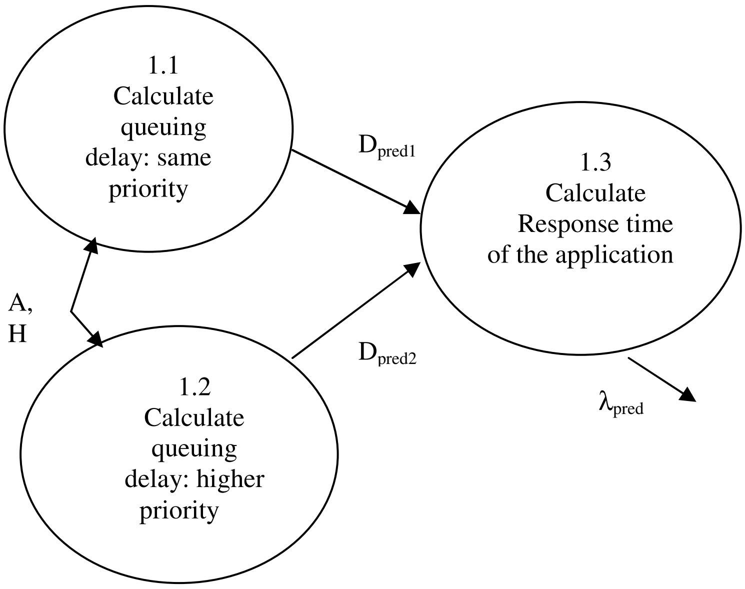
1.1 Calculate queuing delay: same priority
D_{pred1}
1.3 Calculate Response time of the application
A, H
D_{pred2}
λ_{pred}
1.2 Calculate queuing delay: higher priority

Fig. 3. Level 2 DFD of "Predict Response Time"

3 Real-Time Performance Analysis

This section presents approaches for performing estimation of real-time performance of dynamic real-time applications introduced as bubbles, 1.1, 1.2, and 1.3 as shown in Fig. 3. These approaches based on probabilistic techniques are extended to be applicable to time-sharing round-robin scheduler, which considers queuing delay due to the same priority tasks. As mentioned in section 1, the worst-case response time analysis approaches, [11] and stochastic approaches, [5] and [9] are not appropriate for response time analysis in dynamic real-time systems.

3.1 A Execution Rate (ER) Technique

This approach calculates queuing delay of 'A' due to the same priority tasks, and considers applications' periods to find total contention among them. Least Common Multiple (LCM) of periods of 'A' and 'A*' on H is used in this approach. To extend this approach to round-robin scheduler, the execution of ‘A’ is analyzed in the unit of time quantum (TQ) and segment (S). Following steps show how it is processed to compute $D_{pred1}(A)$ due to the same priority tasks.

Step 1. Compute the resource requirement of ‘A’ during the interval, [0,LCM].
Step 2. Compute the resource requirement of the other tasks A* during the interval, [0,LCM].
Step 3. Compute executive rate at which requests of ‘A’ are serviced.
Step 4. Compute queuing delay $D_{pred1}(A)$.

3.2 A Probabilistic Rate (PR) Technique for Same Priority Tasks

An ***improved approach*** called PR calculates queuing delay of ‘A’ due to the same priority tasks for the bubble 1.1 as shown in Fig. 3 by considering the probability of the target task (‘A’) being blocked by any other tasks (‘A*’).

The basic concept is the same as ER introduced in section 3.1. The improved probabilistic rate, pr(A), considers all possible probability that the target task could be executed including computation of progress rate within contention with other tasks as follows: (1) the probability of being with no usage of the resource, (2) the chance that only ‘A’ uses the resource, and (3) the chance that there is progress rate of ‘A’ within contention. This approach also implemented like the ER approach.

3.3 Response Time Prediction

This section simply combines C(A), $D_{pred1}(A)$, and $D_{pred2}(A)$ to compute the predicted response time of the task, $\lambda_{pred}(A)$ as shown as the bubble 1.3 in Fig. 3. That is, $\lambda_{pred}(A) = C(A) + D_{pred1}(A) + D_{pred2}(A)$.

4. Experiments and Summary

This section shows how the approaches accurately predict response time of an application using DynBench, path based s/w systems, introduced by Welch and Shirazi (see [12]).

The ‘filter1’ application (which filters noise from sensed data) in the first sensing path (denoted as “path 1”) as the task is examined on the variable size of workload scenario. Thirty samples of the response time of ‘filter1’ are collected and averaged, when workloads of the path 2 are changed dynamically and monotonically. The workload of sensing path 3 is fixed by 800, and that of sensing path 2 is suddenly increased to 1300 by adding 800 workloads at 1 minute 20 seconds; it drops by 600 at 2 minutes 40 seconds; it adds 700 again at 4 minutes, and drops 500 at 5 minutes 20 seconds. See Fig. 4.

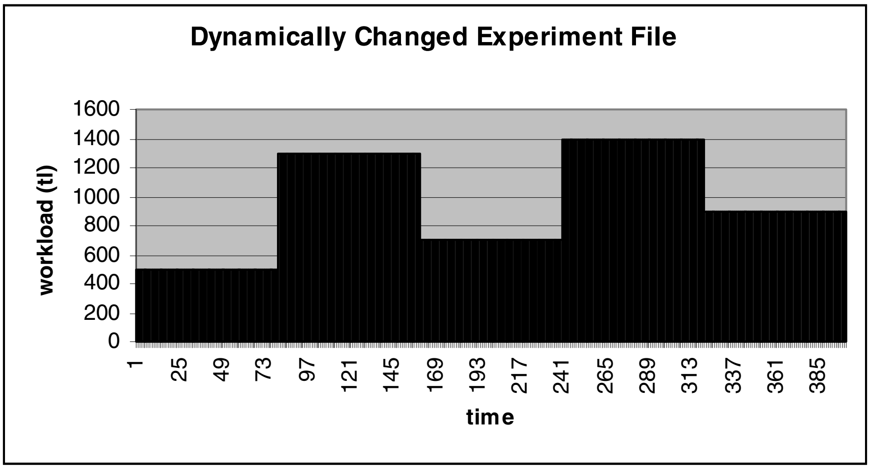

Fig. 4. Dynamically changed workload scenario

The response time, λ_{pred}=D_{pred1}+D_{pred2}+C(filter1), depicted in Fig. 5, is measured using the same approach for each contention case by ER and PR. And the worst-case analysis (denoted as WC) by Audsley in [11] is employed as well to see how the new technique is accurate.

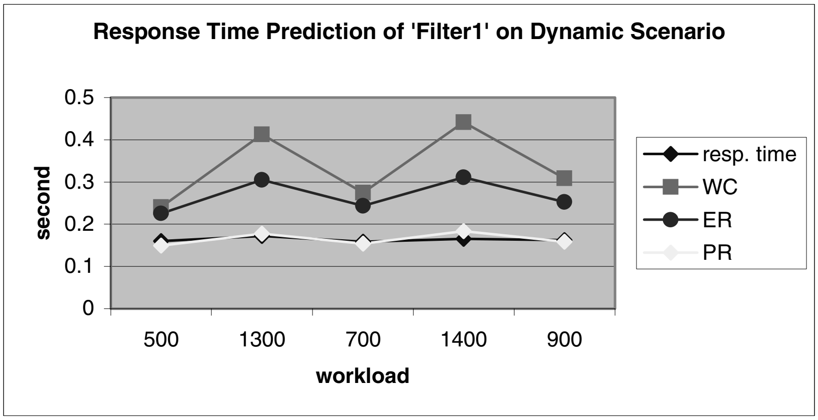

Fig. 5. Predicted Response time comparison

Table 1. Error comparison by each technique

workload	WC	ER	PR
500	0.081	0.066	0.011
1300	0.241	0.133	0.005
700	0.117	0.086	0.004
1400	0.277	0.146	0.018
900	0.147	0.091	0.004

Table 1 shows the errors can be evaluated to find which technique is significant. The PR technique is significant. Overall error in terms of resource utilization is as follows: 17.3 percenatge for the worst-case (WC) , 10.4 percentage for ER, and 0.08 percentage for PR are observed.

From the thirty times experiments, there is no measurement error observed under the 95% confidence interval. The PR, probabilistic contention analysis technique, can accurately predict response time of an application on a host, while the worst-case analysis poorly predicts response times. Therefore, these experiments for the dynamic real-time systems give strong analysis that the worst-case analysis poorly utilizes computational resources, and new approaches can predict response time accurately with the dynamic environment constraints using current utilization. The probabilistic response time prediction method for dynamic real-time systems rather than the worst-case is very useful in terms of resource utilization and certification of real-time performance.

5. Future Study

This research opens the possibility for yet future interesting research. First, there is the topic of certification for an event-driven real-time application which is periodic as well as aperiodic. Second, a new research area derived from this study is the determination of confidence levels of certification, which will be analyzed by mathematical methods with an upper bound, or error.

Acknowledgements

This work is supported in part by the Ministry of Information Communication of Korea, under the "Support Project of University Information Technology Research Center(ITRC)" supervised by KIPA".

References

1. Liu, C. L., and Layland , J.W.: Scheduling algorithms for multi-programming in a hard-real-time environment. Journal of the ACM, Vol. 20. (1973) 46-61
2. Ramamritham, J.A. Stankovic and Zhao, W.: Distributed scheduling of tasks with deadlines and resource requirements. IEEE Transactions on Computers, Vol. 38. (1989) 110-123
3. Haban, D. and Shin, K. G.: Applications of real-time monitoring for scheduling tasks with random execution times. IEEE Transactions on Software Engineering, Vol. 16. (1990) 1374-1389
4. Tia, T. S., Deng, Z., Shankar, M., Storch, M.,Sun, J., Wu, L.C., Liu, J.W.S.: Probabilistic performance guarantee for real-time tasks with varying computation times. Proceedings of the 1st IEEE Real-Time Technology and Applications Symposium. IEEE Computer Society Press (1995) 164-173
5. Lehoczky, J.P.: Real-Time Queueing Theory. Proceedings of IEEE Real-Time Systems Symposium. IEEE Computer Society Press (1996) 186-195
6. Abeni, L. and Buttazzo, G.: Integrating multimedia applications in hard real-time systems. Proceedings of the 19th IEEE Real-Time Systems Symposium. IEEE Computer Society Press (1998) 3-13
7. Stewart, D.B. and Khosla, P.K.: Mechanisms for detecting and handling timing errors. Communications of the ACM, Vol. 40. (1997) 87-93
8. Puschner, P., Burns, A.: A Review of Worst-Case Analysis. The International Journal of Time-Critical Computing Systems, Vol. 18. (2000) 115–128
9. Atlas, A., and Bestavros, A.: Statistical rate monotonic scheduling. Proceedings of the 19th IEEE Real-Time Systems Symposium. IEEE Computer Society Press (1998) 123-132
10. Welch, L.R and Masters, M.W.: Toward a Taxonomy for Real-Time Mission-Critical systems. Proceedings of the First International Workshop on Real-Time Mission-Critical Systems (1999)
11. Audsley, Neil C.: Deadline Monotonic Scheduling. Report YCS-90-146. Department of Computer Science, York University (1990)
12. Welch, L. R., Shirazi, B.: A Dynamic Real-Time Benchmark for Assessment of QoS and Resource Management Technology. IEEE Real-Time Application System (1999)
13. Burns, F., Koelmans, A., Yakovlev, A.: WCET Analysis of Superscalar Processors Using Simulation With Coloured Petrinets. The International Journal of Time-Critical Computing Systems, Vol. 18. (2000) 275–288