International Journal of Computer Applications (0975 – 8887)
Volume 40– No.3, February 2012

Evaluating Robustness of Resource Allocation in Uniprocessor Real Time System

Pratibha Zunjare
Department of Computer Sc. and Engineering
National Institute of Technology, Rourkela
Rourkela, India

Bibhudatta Sahoo
Department of Computer Sc. and Engineering
National Institute of Technology, Rourkela
Rourkela, India

ABSTRACT

Real time systems (RTS) operate in an unpredictable changing environment that causing certain system performance features to degrade. Such systems need robustness to guarantee limited degradation despite variations in environmental parameters. EDF (Earliest Deadline First) scheduling has been used to evaluate the robustness of uniprocessor system with makespan as performance parameter. Robustness of real time system is directly proportional to the makespan of the resource allocation. EDF produces the optimal schedule that maximizes the robustness in the system.

Keywords

Robustness, QoS, Resource Allocation, Real Time System.

1. INTRODUCTION

Real Time systems are widely used in our daily life and production industry such as the robotic control, telecommunications, chemical plant control, satellite control, flight control systems, military systems, multimedia systems, and so on. A real-time system is the one whose logical correctness is based on correctness of outputs as well as timeliness. It consists of a controlling system (computer) and a controlled system (environment). Real time systems guarantee that all the timing requirements can be met by the real time scheduling and schedulability analysis. Real time systems may operate in an environment where certain system performance features degrades due to unpredictable circumstances, such as sudden machine failures, higher than expected system load, or inaccuracies in the estimation of system performance parameters like makespan, slack etc [15, 16].

Resource allocation is very important to achieve a given level of QoS (Quality of Service). The resource allocation is defined as the act of assigning (matching) each task to a machine and ordering (scheduling) the execution of the tasks on each machine. Resource allocation is generally performed based on estimated computation time of each task on each class of machines. A resource allocation is said to be robust with respect to specified system performance features against perturbations in given system parameters if degradation in these features is within acceptable limits when certain perturbations occur [2]. For example, if a resource allocation has been declared to be robust with respect to satisfying a makespan requirement against perturbations in the estimated execution time, then the system configured under that allocation should continue to operate without a makespan violation when the actual execution time is greater than estimated execution time. Makespan (the completion time for an entire set of tasks) is mostly used as the performance feature that requires to be optimized in such systems.
Robustness has been defined in different ways by different researchers. According to [5] robustness is the degree to which a system can function correctly in the presence of inputs different from those assumed. In a more general sense a robust system continues to operate correctly across a wide range of operating conditions. Robustness guarantees the maintenance of certain desired system characteristics despite variations in the behavior of its component parts or its environment [4]. A robust system is one that continues to perform at desired level of service in spite of perturbations in some components that constitute the system. In this paper resource allocation and scheduling are synonymous for uniprocessor system.

The rest of the paper is organized as follows. Section 2 describes system model and defines the resource allocation problem. Section 3 provides the work dealing with robustness and some robustness metrics. Section 4 presents some experiments and their results that highlight the usefulness of the robustness metric. Related work is given in section 5. Section 6 concludes the paper.

2. SYSTEM MODEL

It is based on periodic real-time tasks characterized by environment dependent execution time functions, as opposed to the traditional model with hard periodic real time tasks characterized by worst case execution times. In this paradigm, occasional deadline misses could be tolerated when unpredictable environmental factors drive a demand on resources beyond their limits. This section begins the system model with a traditional model; then necessary extensions are made to incorporate environmental factors.
The system model used in this work is derived from the standard real-time periodic task model by Liu (2000), where a software system consists of a set of n periodic tasks S = {T_1, T_2... T_n}. Each T_i is released periodically with a period of p_i and has a deadline equal to its period. The execution time of each task $T_i \in S$ is a constant e_i that represents the worst case execution time of T_i executed on a set of m identical processors H ={P_1, P_2,..., P_m}.
Workload Model

The traditional model does not capture dynamic and unpredictable environment since the execution times are modeled as constants. It is inadequate for some real-time systems operating in these environments. For instance, a general control system can have filter, analysis, action planning, and actuation tasks, and one or more of the tasks may contain algorithms and execution times that are affected by unpredictable environmental factors. Systems with these properties include air defense, surveillance, and intelligent vehicles. All periodic tasks arrive at the system having different period and are ready to execute any time within each period. We further assume that all periodic tasks have deadlines greater than its period and execution time and their deadlines have to be met even in the presence of uncertainties in execution time.

We considered each task $T_i \in S$ is represented by a tuple (p_i, t_i, d_i) where T_i is released periodically with a period of p_i and has a deadline d_i and has an estimated worst case execution time t_i. The perturbation environmental parameters that affect a system are modeled as $pp = \{pp_1, pp_2, ..., pp_n\}$. Each task T_i has actual execution time e_i that is the function of environmental variable *pp i.e.* $e_i = f(pp)$. Generally, the basic parameter used for performance metric is utilization, given as $U(pp) = \sum^{n}_{i=1} (e_i/ p_i)$.

Robust Resource Allocation Problem: A set of n periodic tasks to be allocated on uniprocessor, such that all tasks should meet their deadlines and maximizes the robustness.

3. ROBUSTNESS TECHNIQUES

3.1 Deterministic Robustness Metric

This section contains a general procedure, called FePIA, for deriving a general robustness metric for any desired computing environment [1, 2]. The name for the above procedure stands for identifying the performance features, the perturbation parameters, the impact of perturbation parameters on performance features, and the analysis to determine the robustness. Specific examples illustrating the application of the FePIA procedure to sample systems are given in the next section. Each step of the FePIA procedure [1, 2] is now described.

1) *Describe quantitatively the requirement that makes the system robust***.** Based on this robustness requirement, determine the QoS performance features like throughput, response time, makespan etc. that should be limited in variation to ensure that the robustness requirement is met. Consider an example *makespan* (the total time it takes to complete the execution of a set of applications) for a given resource allocation, the acceptable variation is up to δ % of the makespan that was calculated using estimated execution times of applications on the machines they are assigned, and the uncertainties in system parameters are inaccuracies in the estimates of these execution times. δ is user defined parameter for allowing variation, for example the acceptable variation is up to 35% of the makespan. Mathematically, let P be the set of system performance features that should be limited in variation. For each element $P_i \in P$, quantitatively describe the tolerable variation in P_i. Let $<b_i^{min}, b_i^{max}>$ be a tuple that gives the bounds of the tolerable variation in the system feature P_i. For the makespan example, P_i is the time the ith machine finishes its assigned applications, and its corresponding $<b_i^{min}, b_i^{max}>$ could be <0, 135*(estimated makespan value) >.

2) *Identify all of the system and environment parameters whose values may impact the QoS(Quality of Service) performance features selected in Step1.* These are called the perturbation parameters for example machine failure, higher than expected system load, execution time exceed than the calculated execution time, inaccuracies in the estimation of system parameters etc., and the performance features are required to be robust with respect to these perturbation parameters. For the makespan example above, the resource allocation where it is desired that the makespan be robust (stay within 135 percent of its estimated value) with respect to uncertainties in these estimated execution times. Mathematically, let Q be the set of perturbation parameters. It is assumed that the elements of Q are vectors. Let Q_j be the jth element of Q. For the makespan example, Q_j could be the vector composed of the actual application execution times, i.e., the ith element of Q_j is the actual execution time of the ith application on the machine it was assigned. In general, representation of the perturbation parameters as separate elements of Q would be based on their nature or kind (e.g., message length variables in Q_1 and computation time variables in Q_2).

3) *Identify the impact of the perturbation parameters in Step 2 on the system performance features in Step1.* For the makespan example, the sum of the actual execution times for all of the applications assigned a given machine is the time when that machine completes its applications. Note that Step1 implies that the actual time each machine finish its applications must be within the acceptable variation. Mathematically, for every $P_i \in P$, determine the relationship $P_i = f_{ij}(Q_j)$, if any, that relates P_i to Q_j. In this expression, f_{ij} is a function that maps P_i to Q_j. For the makespan example, P_i is the finishing time for machine mi, and f_{ij} would be the sum of execution times for applications assigned to machine M_i. The rest of this discussion will be developed assuming only one element in Q. The case where multiple perturbation parameters can affect a given P_i simultaneously will be examined in Section 3.

4) *The last step is to determine the smallest collective variation in the values of perturbation parameters identified in Step 2 that will cause any of the performance features identified in Step 1 to violate its acceptable variation.* This will be the degree of robustness of the given resource allocation. For the makespan example, this will be some quantification of the total amount of inaccuracy in the execution times estimates allowable before the actual makespan exceeds 135 percent of its estimated value [1]. Let $Q_j^{\mathbf{orig}}$ be the value of Q_j at which the system is originally assumed to operate. However, due to inaccuracies in the estimated parameters or changes in the environment, the value of the variable Q_j might differ from its assumed value. Let the distance $\| Q_j^*(P_i) – Q_j^{\mathbf{orig}} \|_2$ be called the robustness radius, $r_\omega(P_i ,Q_j)$, of P_i against Q_j.

$$r_\omega(P_i ,Q_j) = \mathrm{MIN} \| Q_j^*(P_i) – Q_j^{\mathbf{orig}} \|_2$$

3.2 Stochastic Robustness Metric

A stochastic robustness metric for a given distributed computing environment should reasonably predict the performance of the system. This is based on a mathematical model where the relationship between uncertainty in system parameters and its impact on system performance are described stochastically. This stochastic model is then used to derive a quantitative evaluation of the robustness of a given resource allocation as the probability that the resource allocation will satisfy the ex-pressed QoS constraints.

Assume that M copies of a application arrive at computing node at wall clock time t, and n_j is the number of applications pending execution or being executed by computing node j at that time. Let t_0 denote the wall-clock start time of execution for the query being processed by computing node j at time t. The functional dependence between the uncertainty parameters and the performance characteristic at time t, denoted as ζ (t), is

$$\varsigma(\mathrm{t}) = \max_{\mathrm{j}=1,2,..\mathrm{M}} \{T_{1j} - (t - t_0) + \sum_{i=1}^{n_j} T_{i,j}\}$$

Due to its functional dependence on the uncertainty parameters T_{ij}, the performance characteristic in above equation is itself a random variable. Let the QoS constraints be quantitatively described by the values β_{min} and β_{max} limiting the acceptable range of possible variation in system performance, i.e.,
$\beta_{min} \le \zeta \le \beta_{max}$.

The stochastic robustness metric, denoted as θ, is the probability that the performance characteristic of the system is confined to the interval $[\beta_{min}, \beta_{max}]$ i.e.

$$\theta = P[\beta_{min} \le \zeta \le \beta_{max}]$$

The stochastic robustness quantitatively measures the probability that the generated system performance will satisfy the stipulated QoS constraints. Clearly, unity is the most desirable stochastic robustness metric value, i.e., there is zero probability that the system will violate the established QoS constraints [3]. There are different methods of computing the robustness metric like Fast Fourier Transform, Bootstrap method etc. by assuming functional independence among set of local uncertainty parameters $[T_{ij} \,|1 \le i \le n_{ij}]$ and local performance parameter ζ.
Existing robust allocation approaches such as Gertphol et al. (2002), Ali et al. (2003), Juedes et al. (2004) and Gu et al. (2005) have the shortcoming that they employ coarse robustness metrics, which can result in poor allocations. The robustness metric of Ali et al. (2003) was based on the *L2* norm. It measures the radius of maximum environment perturbation environment without violating feasible boundaries. However, the metric only partially characterizes feasible regions with an inner tangent sphere, and no algorithm was developed to optimize it. The max-of-min component of a workload vector among all feasible points was adopted as robustness metric in Gertphol et al. (2002), Juedes et al. (2004), Gu et al. (2005). But it only partially characterizes feasible regions with an inner tangent rectangle. In addition, many existing approaches do not emphasize real-time scheduling and feasibility. For instance, CPUs were assumed to be fair-shared in Gertphol et al. (2002). A special scheduler based on tightness was assumed by Shestak et al. (2005), but no feasibility guarantee was made. Gu et al. (2007) introduced a new robustness metric is accurately characterized the complete space of a feasible region. The robustness of allocation M is defined to be the volume of the entire feasible space of environmental variable w. Since each environmental variable is discrete, the volume can be expressed
mathematically as

$$R(M) = \sum_{w1=0}^{\infty}\sum_{w2=0}^{\infty}..\sum_{wl=0}^{\infty}\Theta(f_m(w_{1,}w_2..w_l))$$

3.3 Other robustness Metrics

Distributed real time system must operate in an environment replete with uncertainty while proving a required level of quality of service (QoS). The System undergoes unpredictable changes causing certain system performance features to degrade. Such system needs robustness to guarantee limited degradation despite fluctuation in the behavior of its components or environment. The robustness gives an idea of the stability of the solution with regards to another performance metric such as schedule length, load balance of an application, queue waiting time of batch scheduler etc.
Some required definitions used in robustness metrics:

i. *Makespan (M):* The total execution time of application.
ii. *Slack of task (S) :* A time window within which the task can be delayed without affecting the makespan
iii. *Entropy of schedule:* It is based on the probability of an execution path that will become critical.
iv. *Expected makespan (E(M)):* The average value of the makespan.

Based on these definitions we define the following robustness metrics [15].

Makespan standard deviation. The standard deviation of the makespan distribution tells how narrow this distribution is. The narrower the distribution, the smaller the standard deviation is. This metric related to robustness because when there are two schedules the one for which the standard deviation is the smaller is the one for which realizations are more likely to have makespan close to the average value. Mathematically

$$\sigma_M = \sqrt{E(M^2) - E(M)^2}$$

Makespan Differential entropy. The differential entropy of distribution of a distribution measures the uncertainty of that distribution. If there is less uncertainty there is more chance than two realizations give a close result hence that schedule is robust.

$$h(M) = \int_{-\infty}^{+\infty} f(x)\log f(x)dx$$

Average slack. The slack gives the sum of space time in the schedule. It is related to robustness of makespan as a schedule with large slack is able to absorb a lot of uncertainty. For deterministic schedule the slack is defined as

$$s = \sum_{i \in V} M - Bl(i) - Tl(i)$$

Where *Bl(i)* is the bottom level of task *i* (the length of the longest path from *i* to an exit node including *i*) and *Tl(i)* is the top level of node *i* (the length of the longest path from entry node to node *i* excluding *i*).

Slack standard deviation. Each task has its own slack. Some tasks have a very large slack and other a slack of zero. As shown in [24] only task with non-zero slack can absorb uncertainty without delaying the makespan. Hence the standard deviation of all slack needs to be as small as possible. For computing the standard deviation of all slack, we use average slack S as shown above and slack of every node $i \in V : s_i = M - Bl(i) - Tl(i)$.
Then we have standard deviation of the slack:

$$\sigma_s = \sqrt{\sum_{i \in V}(s_i - S)^2}$$

Average lateness. A schedule is said late if its makespan exceeds the average makespan. The average lateness as defined in [24] is the average of the difference between the makespan of the late realization and the average makespan. If this metric is large this means that the makespan tends to be far from the average and then that the robustness is low. It is defined as:

$$L = E(M') - E(M)$$

where *M'* is the random variable describing the realization that have makespan larger than *E(M).*

Probabilistic metric. This metric has been defined in [15] and gives the probability that the makespan is within two bounds. If this probability is high, this means that the makespan of a given realization is likely to be close to the average makespan and hence that the robustness is high. An absolute probabilistic metric [15] that measures the probability of the makespan to be within [E(M)-δ, E(M)+δ] where δ a positive constant given by the user. The absolute probabilistic metric is defined as:

$$A(\delta)=P(E(M) - \delta \leq m \leq E(M)+ \delta)$$

The relative probabilistic metric [15] that measure the probability of the makespan to be within [E(M)/ γ, E(M) * γ] where γ is a real number greater than 1. The relative probabilistic metric is defined as:

$$R(\gamma) = P(\frac{E(M)}{\gamma} \leq m \leq \gamma E(M))$$

4. EXPERIMENTS

4.1 Robust EDF Algorithm

The scheduling problem is to find the schedule S that is feasible (meets all task deadlines) and maximizes robustness. Liu and Layland [21] presented a necessary and sufficient schedulability condition for EDF(Earliest Deadline First) scheduling under the assumption that all tasks' relative deadlines are equal to their periods, the schedulability condition is that the total utilization of the task set is less than or equal to 1. It has been proven to be optimal among all scheduling algorithms on a uniprocessor, in the sense that if a real-time task set cannot be scheduled by EDF, then this task set cannot be scheduled by any algorithm. We have considered, arrival time for all tasks is same. Each task is characterized by arrival time, Estimated Time to Compute (ETC), deadline and actual execution time. EDF generates the schedule for tasks according to their closest deadline first. EDF produces the optimal schedule that maximizes the robustness metric R(M). The basic idea of the algorithm (Alg. 1) generates a feasible schedule S which represents sequence of tasks. S is used to derive the robustness of the system.

Algorithm 1 Robust Resource Allocation for Uniprocessor

```
Input: MaxTask, P, T = [T1 T2 . . . T_MaxTask]
Output: Robustness
1: for i = 1 to MaxTask do
2: Compute Utilization $U = \sum_{i=1}^{MaxTask} \frac{t_i}{p_i}$
3:        if U ≥1 then
4:        Tasks are not schedulable
5:    else
6:        Sort Tasks according to Earliest Deadline First
7:        Make Schedule S= [1:MaxTask]
8:        $M = \sum_{i=1}^{MaxTask} t_i$   // Compute Makespan M for S
9:    // Compute a tuple that gives the bounds of the tolerable
      variation in M
10:       vM = [0, 1.35* M]
11:       Compute Finishing time F of Processor
12:       Compute Robustness
13:    end if
14: end for
```

ETC and deadline for each task is uniformly distributed. The task arrival pattern follows Poisson's distribution with ETC. The simulation study has been performed with the simulator designed using Matlab 7.0.1. Performance of EDF has been studied with averaging 10 instances of the allocation. The robustness value for each sample was given *point to plane* formula. Here robustness requirement is that the makespan M of the resource allocation should not increase more than 35% beyond its predicted value which is computed based on the expected execution times of tasks [1].

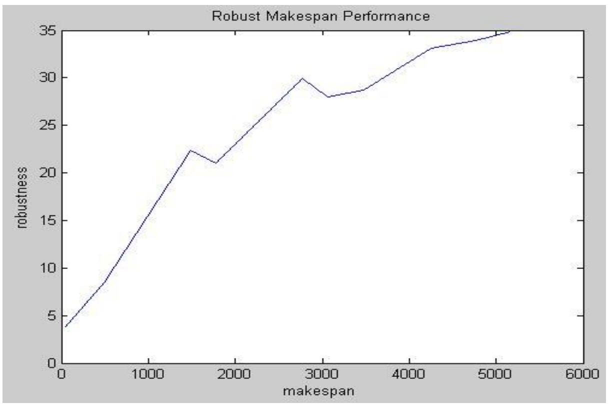

Figure 4.1(a) Performance of EDF with makespan for 1000 tasks

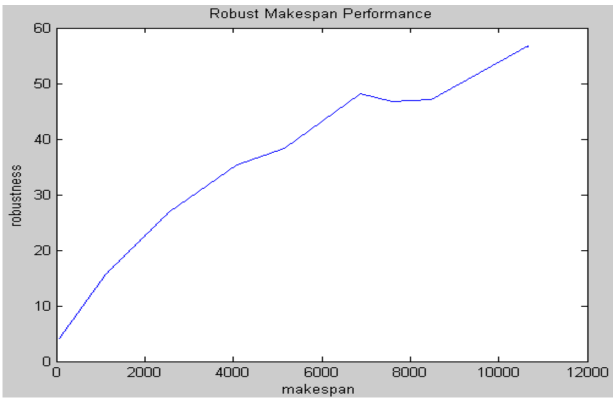

Figure4.1(b) Performance of EDF with makespan for 2000 tasks

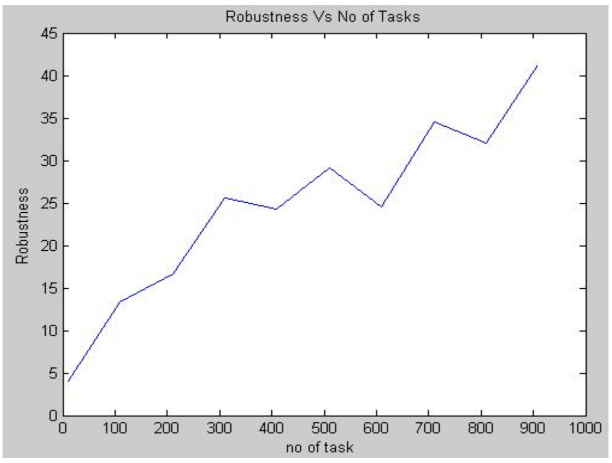

Figure 4.2(a) Performance of EDF with 1000 tasks

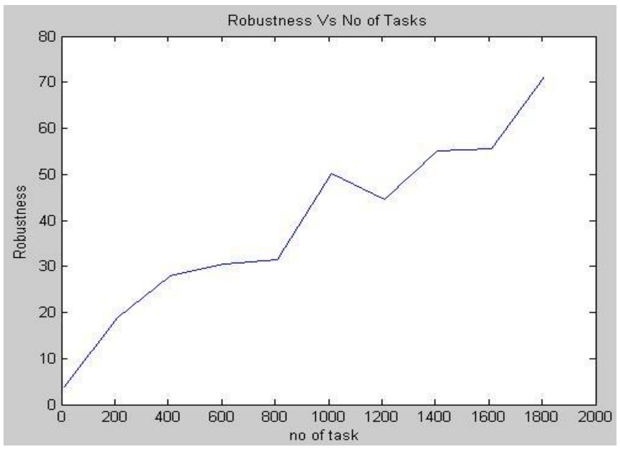

Figure 4.2(b) Performance of EDF with 2000 tasks

Figure 4.1(a, b) shows the performance of EDF with uniformly distributed service time of the different task set on single processor. The simulations indicates the dependency of robustness on makespan; as makespan increases the corresponding robustness also increases. The dependency of robustness on *no of the tasks* arrived with Poisson distribution is shown in Figure 4.2(a, b).

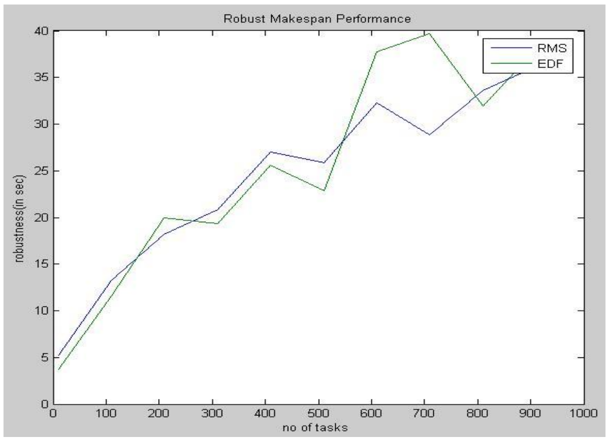

Figure 4.3(a) Robustness for EDF and RM with 1000 tasks

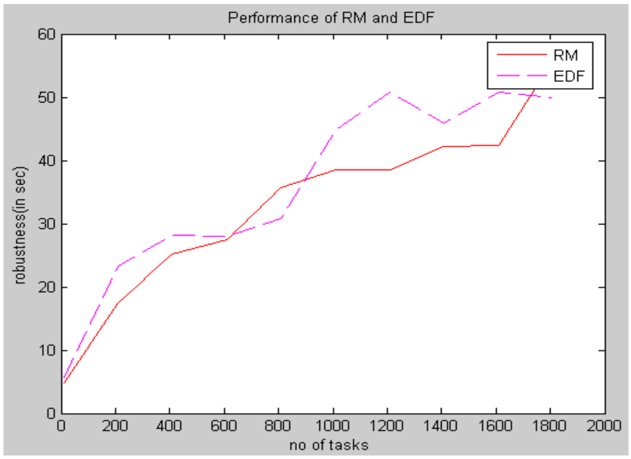

Figure 4.3(b) Robustness for EDF and RM with 2000 tasks

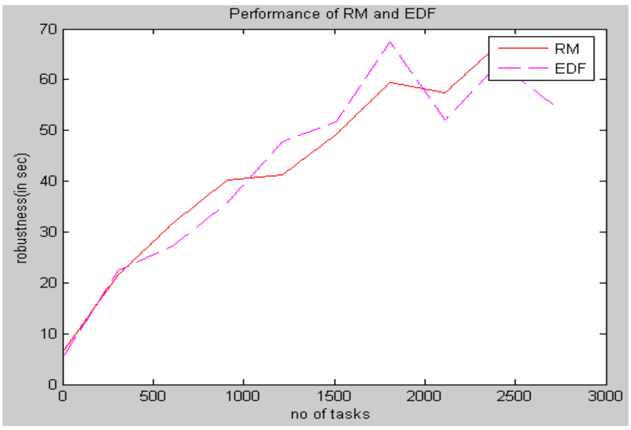

Figure 4.3(c) Robustness for EDF and RM with 3000 tasks

Robustness of EDF on different task sets is compared with the Rate Monotonic (RM) Scheduling on uniprocessor system. The simulation results indicates on better performance of EDF over RM when the no of task on the system greater than 600 or above. A dependency of makespan with no of task can also be observed.

5. RELATED WORK

A universal framework for defining robust resource allocations in heterogeneous computing systems was addressed in [2]. This work referred to a resource allocations tolerance to uncertainty as the robustness of that resource allocation. In [1, 2], a four-step procedure is established for deriving a deterministic robustness metric. The first step is defining robustness metric requires quantitatively describing what makes the system robust. This description establishes the required QoS level that must be delivered to refer to the system as robust essentially bounding the acceptable variation in system performance. In the second step, all modeled system and environmental parameters perturbation parameters that may impact the system's ability to deliver acceptable QoS are identified.

In stochastic approach [11], each perturbation parameter, or uncertainty parameter, is modeled as a random variable fully described by a probability mass function (pmf). Our second approach differs from that in [1], where a single deterministic estimated value for each of the identified perturbation parameters is used. In the third step, the impact of the identified perturbation parameters on the systems performance features is defined. This requires identifying a function that maps a given vector of perturbation parameters to a value for the performance feature of the system. Similarly in [12] stochastic environment, this involves defining the functional dependence between the input random variables and the given performance feature. However, the model this involves more complex computations to combine random variables. Finally, in the fourth step, the previously identified relation is evaluated to quantify the robustness.

As a measure of robustness, the authors in [1] use the minimum robustness radius that relies on a deterministic performance characteristic so used in a deterministic worst-case analysis.

In stochastic model [5, 6, 11, 12], more information regarding the variation in the perturbation parameters is assumed known. Representing the uncertainty parameters of the system as stochastic variables enables the robustness metric in the stochastic model to account for all possible outcomes for the performance of the system. The stochastic robustness metric [9, 11, 12] requires more information and is far more complex to calculate than its deterministic counterpart. To handle the computational complexity, [3] considered the FFT and bootstrap approximation methods that greatly simplify the required calculations. The scheduling problem in real time systems has been explained in [8, 13, 14]. A heuristic dynamic scheduling scheme for parallel real-time jobs, modeled by directed acyclic graphs (DAG) where parallel real time jobs arrive at a heterogeneous system following a Poisson process, in a heterogeneous system is presented in [12, 15].

6. CONCLUSION

A robust schedule can absorb some degree of uncertainty in tasks duration while maintaining a stable solution. This paper discussed the different type of robustness metrics in real time system. The robustness of real time system becomes the vibrant research issue to provide the stability of system in presence of unpredictable operational environments. A number of experiments were conducted to examine the performance and scalability of the robust allocation algorithm by using EDF. From the experiment results we can conclude that robustness of real time system is directly proportional to the makespan. The Majority of real time system operates on multiprocessor system those used in applications like chemical plant control, satellite control, flight control systems, military systems etc. The performance of EDF can tested for the multiprocessor system on different real time system.

7. REFERENCES

[1] S. Ali, A. A. Maciejewski, H. J. Siegel, and J. K. Kim. Measuring the robustness of a resource allocation. IEEE Transactions on Parallel and Distributed Systems,vol. 15, no. 7, pp. 630641, Jul.2004.

[2] S. Ali, A. A. Maciejewski, H. J. Siegel, and J. K. Kim. Definition of robustness metric for resource allocation. Proceedings of International Parallel and Distributed Processing Symposium, page 10, 2003.

[3] Dazhang Gu , Lonnie Welch, Frank Drews, Klaus Ecker. Characterizing robustness in dynamic real time systems. The Journal of Systems and Software 80, pp 1005-1014, Nov 2007.

[4] S. Ali, J.K. Kim, H. J. Siegel, A. A. Maciejewski, Y. Yu, S. B. Gundala, S. Gertphol, and V. Prasanna. Greedy Heuristics for Resource Allocation in Dynamic Distributed Real-Time Heterogeneous Computing Systems. In-ternational conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 02),Vol II,Pages 519-530, June 2002.

[5] Z. Shi, E. Jeannot, and J. Dongarra, Robust task scheduling in non-deterministic heterogeneous computing systems, in Proceedings of the IEEE International Conference on Cluster Computing, Sep. 2006, pp.135-143.

[6] D. Juedes, L. Welch, F. Drews, and D. Fleeman. Resource allocation algorithms for maximizing allowable workload in dynamic, distributed real-time systems. Technical report, Center for Intelligent, Distributed, and Dependable Systems, Ohio University, 2003.

[7] D. Gu, F. Drews, and L. Welch. A characterization of task allocation problems for dynamic distributed real-time sys-tems. IASTED International Conference on Parallel and Distributed Computing and Systems, Cambridge, M A, 2004.

[8] C.M. Krishna and Kang G. Shin. Real Time Systems. McGrawHill, 1997.[cited at p. 5, 17, 57, 63, 64, 66]

[9] F. Drews, L. Welch, D. Juedes, and D. Fleeman Utility-Function based Resource Allocation for Adaptable Applications in Dynamic, Distributed Real Time Systems Proceedings of the 18th International Parallel and Dis-tributed Processing Symposium (IPDPS04) 2004.

[10] Sethavidh Gertphol a nd Viktor K. Prasanna Iterative Integer Programming Formulation for Robust Resource Allocation in Dynamic Real-Time Systems Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS04)2004.

[11] Jay Smith, Luis D. Brice, Anthony A. Maciejewski, Howard Jay Siegel Measuring the Robustness of Resource Allocations in a Stochastic Dynamic Environment IEEE Transaction on Parallel and Distributed System 2007.

[12] Vladimir Shestak, Jay Smith, Howard Jay Siegel, and Anthony A. Maciejewski. A Stochastic Approach to Measuring the Robustness of Resource Allocations in Distributed Systems. Proceedings of the International Conference on Parallel Processing (ICPP06), pages 459-470, Columbus, Ohio, USA, August 2006.

[13] A. Burns. Scheduling hard real-time systems: a review. Software Engineering, 6(3):116-128, May 1991.

[14] J. Goossens, Shelby Funk, and Sanjoy Baruah. Priority-driven scheduling of periodic task systems on multiprocessors. Technical Report TR01-024, 14 2001. [cited at p. 5]

[15] L. Canon, E. Jeannot. A comparison of Robustness Metrics for Scheduling DAGs on Heterogeneous Systems. Proceedings of the IEEE International Conference on Cluster Computing, pages 558-567, 2007.

[16] Z. Shi E. Jeannot. Robust task scheduling in nondeterministic heterogeneous computing systems. Proceedings of IEEE International Conference on Cluster Computing, pages 630-641, 2006.