Reliability-Driven Scheduling of Periodic Tasks in Heterogeneous Real-Time Systems

Wei Luo†, Xiao Qin‡*, Kiranmai Bellam‡
School of Computer Science and Technology
HuaZhong University of Science and Technology
Wuhan, Hubei, P.R.China†
Department of Computer Science
New Mexico Institute of Mining and Technology
801 Leroy Place, Socorro, New Mexico 87801-4796 USA‡
E-mail: free_xingezi@163.com, {xqin, kiran}@cs.nmt.edu

Abstract

In this paper we comprehensively investigated the issue of reliability-driven real-time scheduling for periodic tasks in heterogeneous systems. First, we built a reliability model in which the concept of reliability cost is introduced in the context of heterogeneous real-time systems. Next, we proposed a novel reliability-driven scheduling algorithm (referred to as Repars) for periodic tasks in heterogeneous systems. Third, after extending the reliability model to meet the needs of our fault-tolerant scheme, we developed a fault-tolerant scheduling algorithm or Refine. Refine aims to enhance system reliability while being able to tolerate one-processor failures in heterogeneous real-time systems. Experimental results showed that Repars is superior to RMFF in terms of both schedulability and reliability. When compared with Repars, Refine significantly reduced the reliability cost by up to 34% with graceful degradation in schedulability.

1. Introduction

With growing needs of building reliable real-time applications coupled with advancement of high-speed networks and high-performance computers, heterogeneous systems have been increasingly used for many real-time safety-critical applications like avionic control and nuclear control systems in which the correctness of the systems depend not only on the results of a computation but also on the time at which these results are produced [1].

Reliability has been a main concern of the research community for many years. Conventionally, the reliability of a system is defined as the probability that the system functions properly and continuously without any interruption [2]. With the emergence of critical business applications (i.e., e-commerce systems) and safety critical systems (i.e., space shuttle systems), the traditional definition of reliability needs to be extended to incorporate fault tolerance. However, to the best of our knowledge, most existing reliability models constructed for real-time systems have not addressed the issue of fault tolerance [3].

In the past decade, an array of heuristics scheduling algorithms have been designed and implemented to schedule periodic real-time tasks running in uniprocessor or multiprocessor systems. Liu and Layland first introduced the well-known Rate-Monotonic scheduling (RM) algorithm for preemptively scheduling a set of periodic tasks on a single processor [4]. Joseph and Pandya proposed the Completion Time Test (CTT) for checking the schedulability of a set of fixed-priority tasks on a single processor. The CTT scheme is a necessary and sufficient schedulability criterion which is stronger than the method of the least upper bound of processor utilization [5]. Dhall and Liu proposed the Rate-Monotonic First-Fit algorithm (RMFF) which is an extension of the RM algorithm [6]. The RMFF can be

* Corresponding author. *http://www.cs.nmt.edu/~xqin*
This paper appeared in the Proc. of the 4th IEEE International Symposium on Embedded Computing, Ontario, Canada, May 2007.

used to generate schedules for real-time multiprocessor systems.

Growing evidence shows that scheduling is a key factor in obtaining high reliability and performance in heterogeneous systems. The concept of reliability cost was factored in a number of heterogeneity-aware scheduling algorithms for tasks with precedence constrain [2][7]. Although reliability is a main objective of these scheduling algorithms, reliability models in these studies are geared to handle non-real-time, aperiodic and non-preemptive tasks. Hence, these scheduling approaches are inadequate for real-time applications. In addition, none of the above reliability models incorporate fault-tolerance.

Fault tolerance is an inherent requirement of real-time systems. The primary-backup scheme is an efficient fault tolerant technique in parallel and distributed systems. The three variants of this scheme include active backup copy [8], passive backup copy [9] and primary backup copy overlapping [10]. Generally speaking, backup copy is always preferred to be executed as passive backup copy because it can take the advantages of backup copy overloading and backup copy de-allocation technique to improve schedulability [9]. The common drawback of the scheduling algorithms presented in [8,9,10] is that the algorithms merely support one type of backup copy.

Bertossi *et al.* proposed a scheduling algorithm where both active and passive backup copies are incorporated into the well-known Rate-Monotonic First-Fit assignment algorithm to provide fault tolerance [11]. Although their scheduling algorithm overcomes the drawbacks of timing constraints on backup-copies, it does not consider heterogeneity or reliability. The issues of scheduling for heterogeneous systems have been addressed in many papers. In our previous studies, we extensively studied the issue of real-time and fault-tolerant scheduling for heterogeneous systems [9,12]. Our algorithms aim to improve the reliability of heterogeneous systems without any additional hardware cost. Although both the real-time and reliability issues were addressed in the studies, our previous reliability model is only suitable for aperiodic, non-preemptive tasks. Recently, we constructed a novel reliability model for real-time periodic tasks running in fault-tolerant heterogeneous systems [13]. In the light of this reliability model, we developed a real-time fault-tolerant algorithm (referred to as DFTAHS) for heterogeneous systems. RDFTAHS aims at boosting reliability while guaranteeing timing requirements of periodic tasks. There are two assumptions for RDFTAHS. First, only one processor failure is tolerated in the worst case. Second, there are sufficient processors in a heterogeneous system. Moreover, the reliability model in RDFTAHS does not deal with any processor failure.

Although numerous algorithms have been developed with respect to real-time scheduling for heterogeneous systems, much less attention has been devoted to reliability-driven real-time scheduling for periodic tasks. To bridge this gap in real-time scheduling technology, in this study, we first built a novel reliability model in the context of real-time periodic tasks for both non-fault-tolerant and fault-tolerant systems. Next, we developed two reliability-driven scheduling algorithms for periodic tasks in heterogeneous systems (referred to as *Repars* and *Refine*). *Repars* manages to assign tasks in a way to improve the reliability of heterogeneous systems while meeting real-time constraints of real-time periodic tasks. The primary-backup copy scheme is incorporated into our algorithms to make heterogeneous system fault-tolerant. *Refine* aims at increasing the reliability of heterogeneous systems by introducing a primary-backup scheme while providing fault tolerance.

This paper is organized as follows. In section 2, we present the system and reliability models. The *Repars* is described in Section 3. Section 4 proposes the *Refine* algorithm-a reliability-driven fault-tolerant scheduling algorithm. In Section 5 experiments results are analyzed and discussed. Finally, Section 6 concludes the paper with a summary and future work.

2. The System Model

In this section we describe a system model and then present a reliability model that captures the typical reliability characteristics of real-time periodic tasks in heterogeneous systems characterized by the system model

2.1 System Model

Our model considers a typical heterogeneous system structure, i.e., processors accessing their local memory modules are connected to shared-memory modules via an interconnection network. Formally, a heterogeneous system in this study comprises of a set $\Gamma = \{\tau_1, \tau_2, \tau_3, \ldots, \tau_N\}$ of tasks in addition to a set $\Omega = \{P_1, P_2, \ldots, P_M\}$ of processors executing the task set. The *i*th periodic preemptive task τ_i is characterized by two parameters: period T_i and execution time vector $C_i = [c(i,1), c(i,2), \ldots, c(i,M)]$. A measure of *computational heterogeneity* is modeled by a function, $C: V \times P \rightarrow Z^+$, which represents the execution time of each task on each processor in the system. Thus, *c(i,j)* denotes the execution time of task τ_i on processor P_j. We apply the rate-monotonic algorithm to schedule tasks allocated to any processor in the heterogeneous system. It is assumed that that there are enough processors to execute the tasks set. This assumption is reasonable as the extra processors can be readily added to a system if additional processors are required to shorten schedule lengths. Without loss of generality, we assume that processor failures are independent of each other, and the reliability of the heterogeneous system is modeled by a set of failure rates $R = \{\lambda_1, \lambda_2, \ldots,\lambda_M\}$.

2.2 Reliability Model

Srinivasan and Jha defined reliability of a system with respect to a task set as the probability that the system can run the task set without any failure [2]. The concept of reliability in their study relies on reliability cost. It should be noted that the reliability cost of a task on a processor is a product of the task execution time and the failure rate of the processor. The reliability cost used in their study is defined for a set of aperiodic tasks that are non-preemptive in nature. Therefore, their reliability model is inadequate for a set of periodic and preemptive tasks. To remedy this problem, in what follows, we build a new reliability cost model for periodic, preemptive tasks.

It is assumed that processor failures follow a Poisson Process with an arrival rate λ referred to as failure rate. Thus,

$$\Pr_n(t) = \frac{e^{-\lambda t}(t * \lambda)^n}{n!}$$

gives the probability of *n* events during the time interval *[0, t]*. Let *W* be a random variable for the number of faults occurred in a heterogeneous system. We denote $NF_i(t)$ as the probability that processor P_i is running without any failure by time *t*. Therefore, $NF_i(t)$ can be expressed as follows:

$$NF_i(t) = \Pr[W = 0] = e^{(-t\lambda i)} \tag{1}$$

where λ_i $(1\leq i\leq M)$ is P_i's failure rate.

Hence, the reliability of the whole system during time interval [0, t] can be written as:

$$Reliability(t) = \prod_{i=1}^{M} NFi(t) = \prod_{i=1}^{M} \Pr[W = 0]$$

$$= \prod_{i=1}^{M} e^{(-t\lambda i)} = \exp(-\sum_{i=1}^{M}(-t\lambda_i)) \tag{2}$$

A processor might fail during an idle time, but it is assumed that processors' failures during an idle time interval are not considered in our reliability model.

The characteristic of a periodic task set is captured by hyper period *H* defined as the least common multiplier of periods of all tasks in the set, i.e., $H = LCM\{T_i \mid \tau_i \in \Gamma\}$. We denote $A_i = H/T_i$ as the number of instances of a task in the hyper period.

Since the Poisson Process is a stable incremental process, the processors in a heterogeneous system have the same probability to fail during any amount of time interval. Thus, we have:

$$Reliability(H) = Reliability(t+H) - Reliability(t) \tag{3}$$

Eq.3 indicates that we can study the reliability of a system by measuring the reliability of a set of tasks running in the system within the hyper period of the task set. We can derive the reliability of a heterogeneous system executing periodic tasks as below:

$$Reliability(\Omega,\Gamma,H) = \exp(-\sum_{i=1}^{M}(-t\lambda_i)))$$

$$= \exp(-\sum_{k=1}^{M}(\sum_{P(\tau_i)=P_k}\lambda_k * C[i,k] * A_i))]$$

$$= \exp(-\sum_{k=1}^{M}(\sum_{P(\tau_i)=P_k}\lambda_k * C[i,k] * H/T_i)) \tag{4}$$

where H/T_i is the number of instances of the *i*th task in the hyper period, and the product of C[i,k] and H/T_i is the total execution time of the *i*th task on the *j*th processor in the hyper period. We define the reliability cost of a heterogeneous system with respect to a set of periodic tasks as:

Definition 1: Given a set Γ of real-time periodic tasks running on a system with a set Ω of heterogeneous processors, we compute the system reliability cost of the system as

$$Reliability\ Cost(\Gamma,\Omega) = \sum_{k=1}^{M}\lambda_k(\sum_{P(\tau_i)=P_k} C[i,k]/T_i) * H \tag{5}$$

Eq.5 signifies that to improve the reliability of the system, one has to find an efficient way to reduce reliability cost. We can minimize reliability cost by allocating tasks with long execution times to more reliable processors

3. The *Repars* Algorithm

Like RMFF [11], our *Repars* algorithm performs the first fit strategy, assigning tasks to the first processor that could meet the needs of the tasks. Different from conventional scheduling algorithms, the *Repars* algorithm chooses the best candidate processor with respect to a task that is it selects a processor providing the task with the smallest reliability cost which in turn results in the highest reliability for the task.

The schedulability test in *Repars* goes in a similar way as the one described in [5] except that in the test for *Repars,* the execution times of a task on heterogeneous processors are different.

Definition 2: Given a set Γ of real-time tasks and a set Ω of processors, a reliability cost vector of task τ_i is defined as $RCV_i = [rcv_{i1}, rcv_{i2}, ..., rcv_{iM}]$, where the element rcv_{ij} in RCV_i is modelled as a pair of two parameters, i.e., $rcv_{ij} = (rc_{ij}, \rho_{ij})$. Here, ρ_{ij} is the processor sequence number. rcv_{ij} is derived from vector C_i and Ω. That is, $rc_{ij} = (C[i, \rho_{ij}]/T_i) \times \lambda \rho_{ij}$. Elements in RCV_i are sorted in the order of non-decreasing reliability cost. Thus, we have

$$\forall i \in [1, N], \forall j, k \in [1, M] : (j < k \rightarrow (\mathrm{rc}_{ij} \leq \mathrm{rc}_{ik}))$$

Prior to allocating tasks, all the tasks are sorted in the increasing order of period. This is because the priority of a task equals to the inverse of its period. Thus, tasks are assigned to processors following the order τ_1 ,τ_2 … τ_N. Assigning tasks in accordance to task's RM priorities greatly simplifies the implementation of our algorithms.
A detailed pseudo code of the *Repars* algorithm is presented as follows.

The *Repars* Algorithm:

```
1)  Set the number of processors to be one, $Min\_pro \leftarrow 1$, reorder the real-time tasks in $\Gamma$ according to decreasing of their priority;
2)  for i $\leftarrow$ 1 to $N$ do /* Schedule Real-Time tasks by the increasing order of their priorities */
3)      Calculate the Reliability Cost Vector of $\tau_i$ on current $M$ processors as $RCV_i$;
4)      Try to assign the task to the processor on which the reliability cost is minimal, that is Schedule tasks to the first process that could be fitted in by $RCV_i$;
5)      if all current $Min\_pro$ processors can not accommodate $\tau_i$ then
6)          $Min\_pro \leftarrow Min\_pro + 1$;
7)          $P(\tau_i) \leftarrow P_{Min\_pro}$;  /* Schedule $\tau_i$ on a new processor $P_{Min\_pro}$ */
8)      end if
9)  end for
10) Return $Min\_pro$
```

4. The *Refine* Algorithm

To make heterogeneous system fault-tolerant, we incorporate the primary-backup copy scheme into our algorithm. A reliability model is extended in light of the fault-tolerant model that takes into account, the reliability in presence of one-processor failure. Based on the new reliability model, we develop in this section, a real-time and fault-tolerant algorithm (or *Refine* for short) for heterogeneous systems.

4.1 Employing Primary Backup Copy

Now, we introduce a set of backup copies of real-time tasks, $B\Gamma = \{\beta_1 , \beta_2 , \beta_3 ,…, \beta_N\}$, according to given primary copies. $\beta_i = (D_i, T_i)$ $(i = 1,2,…,N)$ is a backup copy with respect to τ_i . D_i is the execution time vection of τ_i. Without loss of generality, it is assumed in our model that the backup and primary copies of a task are completely identical, i.e., $D_i = [c(i,1), c(i,2),…,c(i,M)]$. Note that T_i denotes the period of β_i. In our model each backup copy has two forms: passive and active backup copies. It is imperative to assign the primary copy of a task before assigning the corresponding backup copy. The status of backup copy is determined by the following:

$$Status(\beta_i) = \begin{cases} passive & T_i - R(i,j) > D_{ik} \\ active & T_i - R(i,j) \leq D_{ik} \end{cases} \quad (6)$$
$$where\ \ P(\tau_i) = P_j\ \ and\ \ P(\beta_i) = P_k$$

R(i,j) in Eq. (6) denotes WCRT (worst case response time) of τ_i assigned to P_j. For ease of presentation, γ_i represents a primary copy or a backup-copy, namely, $\gamma_i = \tau_i$ or β_i.

To facilitate the description of the *Refine* algorithm, we introduce the following notation: *Primary*(P_j), *Backup*(P_j), *active*(P_j), *passiveRecovery*(P_j,P_f)

activeRecover(P_j,P_f), and *recover*(P_j, P_f). Note that these were similar as the notations used in [11].

4.2 Fault-Tolerant reliability model based on Periodic Tasks

It is, of course, non-trivial to address the issue of reliability analysis coupled with fault tolerance. A k-timely-fault-tolerant (k-TFT) schedule is defined as a schedule in which no task deadlines are missed, despite k arbitrary processor failures [14]. Our goal in this study is to achieve 1-TFT by using processor and task redundancies in the scheduling algorithm. We define the reliability of systems as below:

Definition 3. Given a set of real-time tasks set Γ along with their corresponding backup copy set BΓ scheduled on a system with a set Ω of heterogeneous processors. The reliability of the system is the probability of all the tasks in the task set can be completed before their deadlines within their hyper period despite of one-processor failures.

It is intuitive that the reliability of a heterogenous system relies on two factors: the reliability in case of no permanent processor failures and the reliability in the midst of any permanent processor failures. The state of the system is represented by a random variable *G*, whose value is in the set {0, 1, 2… *Min_pro*}. More precisely, *G = 0* means that no processor permanently fails, and *G = i* (1 ≤ i ≤ *Min_pro*) signifies that P_i encounters permanent failures. The probability of *G* in a hyper period is expressed by Eq. (7).

$$\Pr[X=g]=\begin{cases}\prod_{i=1}^{Min\_pro} NFi(\theta_i)=\prod_{i=1}^{Min\_pro} e^{-\lambda_i\theta_i} & g=0\\ (1-NFi(\theta_i))\prod_{j=1,j\neq i}^{Min\_pro} NFj(\theta_j)=(1-e^{-\lambda_i\theta_i})\prod_{j=1,j\neq i}^{Min\_pro} e^{-\lambda_i\theta_i} & \text{for } g=\mathrm{i}(1\le \mathrm{i}\le Min\_pro)\end{cases} \quad (7)$$

where θ_i denotes the total execution of tasks allocated to processor P_i. The value of θ_i can be derived as

$$\theta_i=\begin{cases}\sum_{P(\gamma_j)\in\sigma}(C[j,i]\times(\frac{H}{T_j})) & \text{if no processor fails}\\ \quad\text{where}\quad \sigma=Primary(P_i)\cup active(P_i) & \\ \sum_{P(\gamma_j)\in\sigma}(C[j,i]\times(\frac{H}{T_j})) & \text{if processor } P_f \text{ fails}\\ \quad\text{where}\quad \sigma=Primary(P_i)\cup recovery(P_f,P_i) & \end{cases} \quad (8)$$

According to the definition of reliability, the reliability of the heterogeneous system can be written as below:

$$Reliability_{Ft}(\Omega,\Gamma,B\Gamma,H)=\sum_{g=0}^{Min\_pro}\Pr\{X=g\}=\prod_{i=1}^{Min\_pro} e^{-\lambda_i\theta_i}+\sum_{i=1}^{Min\_pro}(1-e^{-\lambda_i\theta_i})\prod_{j=1,j\neq i}^{Min\_pro} e^{-\lambda_i\theta_i} \quad (9)$$

It should be noted that reliability can be expressed as a function of reliability cost, which is a concept widely used in the literature [2][12]. In our reliability model, the reliability cost is defined as a log function of Reliability:

$$Reliability\ \ Cost(\Omega,\Gamma,B\Gamma,H)=-\log(Reliability) \quad (10)$$

4.3 Description of the *Refine* algorithm

Now, we are positioned to propose an algorithm for scheduling a periodic task set along with its corresponding set of backup copies in a heterogeneous system. The *Refine* algorithm goes in a similar way as the FTRMFF [11]. The difference is that when assigning a primary copy or a backup copy, Refine tends to assign a task copy to a processor with smallest reliability Cost. Due to space constraints of this paper, please refer to [11] for a detailed description of the algorithm.

5 The Performance Evaluation

We conduct extensive simulations to evaluate the performance of our algorithms. To reveal performance improvements gained by our proposed algorithms, we first compare the *Repars* with the well-known scheduling algorithms, namely, RMFF (Rate-Monotonic First-fit) [11].

5.1 Experimental platform

In our experiments, we simulate large task sets of periodic tasks, which are generated according to the following parameters:

1. Periods of tasks (T_i) is a value generated randomly distributed in the interval [250, 125, 50, 25, 20, 10, 5, 2];
2. Execution time of any task on any processor (C[i, j])—a value taken from a random distribution in the interval 0<C[i, j] $\leq \alpha$ T_i, parameter

$$\alpha = \max_{i=1,\ldots,N,\, j=1,\ldots,M} C[i,j]/T_i$$

3. Size of any task set (L) — a value selected from a specific set, [200, 400, 600, 800, 1000].
4. The failure rate (FR) for each processor is uniformly selected between the ranges 0.75 to 1.25*10^{-6}/Sec.

Two performance measures namely *schedulability* and reliability cost are used to capture two important but different aspects of real-time scheduling.

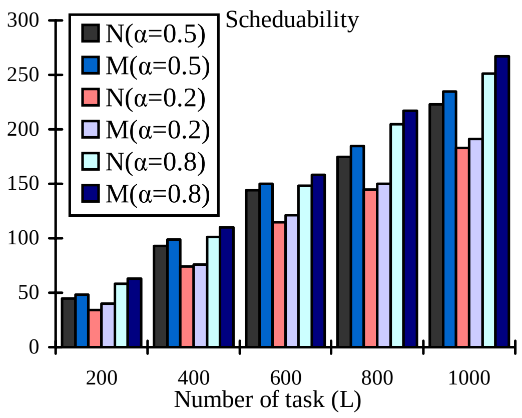

Fig. 1. Comparison between *Repars* and RMFF of Schedulability as a function of L.

5.2 Performance comparisons between *Repars* and RMFF

We compare *Repars* with RMFF in terms of both schedulability and reliability. The failure rates of processors are uniformly distributed in [0.95, 1.05]×10^{-6}/sec. α is fixed to 0.2, 0.5 and 0.8. For briefness, we denote N as the number of processor required by *Repars*, and M as number of processors required by RMFF for any task set.

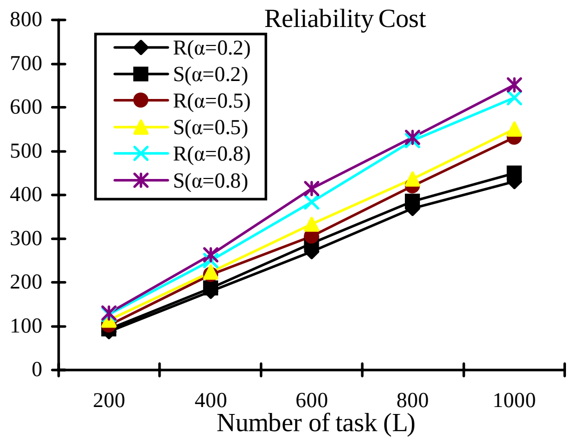

Figure 2. Comparison between *Repars* and RMFF of Reliability Cost as a function of L.

It is clear that Fig 1 show that *Repars* noticeably reduces the number of processors needed to complete task sets before specified deadlines. This performance improvement becomes more pronounced when α is set to 0.2.

Fig. 2 shows the reliability cost yielded by the two evaluated algorithms. For simplicity, we denote R as the reliability cost of the system using *Repars* as a scheduler, and S as the reliability cost of the system using RMFF to schedule task sets.

5.3 Performance comparisons between *Repars* and *Refine*

Now we are in a position to compare *Refine* with *Repars.* For the ease of presentation, we denote N and K as the numbers of processors for task sets scheduled by *Repars* and *Refine* respectively. Fig. 3 indicates that compared with *Repars*, *Refine* needs more processors in order to meet the real-time requirements of periodic tasks.

The reliability costs of *Refine* and *Repars* are plotted in Fig.4 as function of the number of tasks in a set. We denote R and T as the reliability costs of the same task sets scheduled by *Repars* and *Refine*, respectively. We observe from Fig. 5 that for both *Refine* and *Repars* reliability cost increases with the increasing number of tasks in a task set. Most importantly, it is observed that *Refine* performs better *Repars* in terms of reliability cost.

Moreover, the performance benefit gained from *Refine* becomes more pronounced as the values of L and α are larger. We conclude that *Refine* can achieve high reliability for heterogeneous real-time systems by employing the reliability-driven scheduling technique.

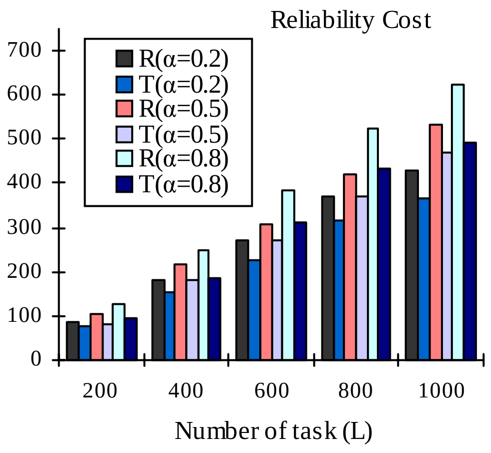

Figure 3. Comparison between *Repars* and *Refine* of Schedulability as a function of L.

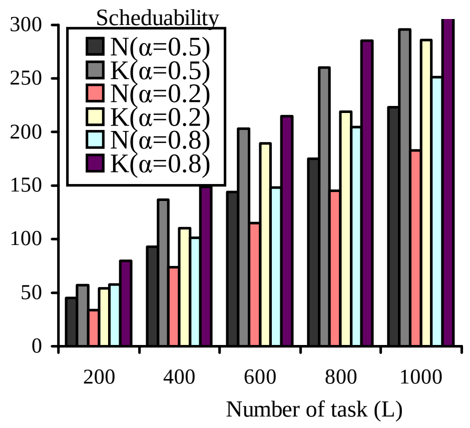

Figure 4. Comparison between *Repars* and *Refine* of Reliability Cost as a function of L.

6. Summary and Future works

In this study we first built a novel reliability model in the context of real-time periodic tasks. The reliability model considers both non-fault-tolerance and fault-tolerance in heterogeneous systems. Next, we developed two reliability-driven scheduling algorithms (referred to as *Repars* and *Refine*) for periodic tasks in heterogeneous systems. *Repars* manages to assign tasks in a way to improve the reliability of heterogeneous systems while meeting real-time constraints of periodic tasks.

Unlike *Repars*, *Refine* aims to make heterogeneous system fault-tolerant by incorporating the primary-backup scheme in to the process of scheduling. Compared with existing real-time and fault-tolerant scheduling schemes, *Refine* is more flexible in the sense that it considers backup copies in both active and passive forms. Experimental results show that *Repars* is superior to RMFF in terms of both schedulability and reliability, while compared with *Repars*, *Refine* can significantly reduce the reliability cost by up to 34% with graceful degradation in schedulability.

Acknowledgements

The work reported in this paper was supported in part by the New Mexico Institute of Mining and Technology under Grant 103295, by Intel Corporation under Grant 2005-04-070, and by Altera Corporation under an equipment grant.

References

[1] W.A. Halang, R Gumzej, M. Colnaric, and M. Druzovec, "Measuring the Performance of Real-Time Systems", *Journal of Real-Time Systems*, Vol.11 No.1, Jan. 2000

[2] S. Srinivasan, and N.K Jha, "Safety and Reliability Driven Tasks Allocation in Distributed Systems", *IEEE Trans. Parallel and distributed systems*, Vol.10 No.3, Mar. 1999

[3] A. Heddaya, and A Helal, "Reliability, Availability, Dependability and Performability: A User-centered View", Technical Report, College of Arts and Science, Boston University, May 15, 1997

[4] C.L. Liu, and J.W. Layland,"Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment", *Journal of ACM*, Vol 11 No 1, pp.46~61, 1973.

[5] M.H. Klein, J.P. Lehoczky, R. Rajkumar, "Rate-Monotonic Analysis for Real-Time Industrial Computing", *Computer*, Vol.27, No.1, Jan.1994

[6] S.K. Dhall, C.L. Liu, "On a real-time scheduling problem", *Operations Research*, Vol.26, No 1, Jul. 1978

[7] A. Dogan, F. Ozguner, “Reliable matching and scheduling of precedence-constrained tasks in heterogeneous distributed computing,” *In Proc. of the 29th International Conference on Parallel Processing*, pp. 307-314, 2000.

[8] C.H. Yang, G. Deconinck, W.H Gui, “Fault-tolerant scheduling for real-time embedded control systems”, *Journal of Computer Science and Technology*, Vol.19, No.2, Mar. 2004

[9] X. Qin, H. Jiang, D. R. Swanson, “An Efficient Fault-tolerant Scheduling Algorithm for Real-time Tasks with Precedence Constraints in Heterogeneous Systems,” *Proc. 31st Int’l Conf. Parallel Processing*, pp.360-368. Aug. 2002.

[10] A. Omarir, A.K. Somani, and G. Manimaran, “An adaptive scheme for fault-tolerant scheduling of soft real-time tasks in multiprocessor systems”, *Journal of Parallel and Distributed Computing*, Vol 65, No 5, May. 2005

[11] A.A. Bertossi, L.V. Mancini, and F. Rossini, ”Fault-tolerant rate-monotonic first-fit scheduling in hard-real-time systems”, *IEEE Trans. Parallel and Distributed Systems*, Vol.10, No.9, Sep.1999

[12] X. Qin and H. Jiang, “Dynamic, Reliability-driven Scheduling of Parallel Real-time Jobs in Heterogeneous Systems,” *Proc. 30th Int’l Conf. Parallel Processing*, pp.113-122, Sept. 2001

[13] W.Luo, F.M. Yang, L.P Pang, and X, Qin. “Fault-tolerant Scheduling Based on Periodic Tasks of Heterogeneous Systems”, *Proc of the 3rd International Conf. Autonomic and Trusted Computing*, pp571~580, Sept 2006.

[14] Y. Oh, and S. H. Son, "Scheduling Real-Time Tasks for Dependability," *Journal of Operational Research Society*, Vol.48, No. 6, pp 629-639, June 1997