Distributing periodic workload uniformly across time to achieve better service quality

Jaeyong Koh, Kihan Kim, and Heonshik Shin

Dept. of Computer Engineering, Seoul National University,
Shinlim-dong San 56-1, Gwanak-gu, Seoul 151-742, Korea.
{jyk, shinhs}@ce2.snu.ac.kr
http://cselab.snu.ac.kr/member/alcohol/jaykoh.html

Abstract. A multimedia system consists of substantial amount of continuous media workload scheduled periodically at deterministic time points. Phased scheduling is achieved by shifting *phase* or the lag between the invocation times of independent periodic tasks. A proper phase configuration distributes workload uniformly over time and reduces task interference that may otherwise result in jitter, deadline miss, and long response time. In our previous work, an algorithm is proposed that identifies optimal phase values for given periodic task set scheduled earliest deadline first[1]. We now present another phase identification algorithm and evaluate its effectiveness. The algorithm in this paper is less accurate but faster than the previous algorithm. In addition, the new approach is applicable to wider set of problems.

1 Introduction

Phased scheduling[1] has been proposed to deliver a better service quality of multimedia end system consisting of multiple periodic and aperiodic tasks. It is achieved by shifting the phase, or points of time at which mutually non-synchronized periodic tasks are scheduled for execution. In [1], we presented a phase optimization algorithm and evaluated the performance gain that results from the optimization. It is shown that phased scheduling substantially reduces deadline miss ratio and average response time of real-time tasks. However, our previous work is applicable only to Earliest Deadline First(*EDF*) scheduling discipline. Furthermore, it requires substantial amount of time for the optimization.

This paper presents an alternative approach to improve the system performance through phased scheduling. The new algorithm attempts to distribute the periodic workload evenly across the time span using artificial phases. This *time-wise* load balancing of periodic workload reduces the *jitter* or the variance in task response time. It improves worst-case performance by removing the *critical instant* where all the periodic tasks are released simultaneously. The proper phase also improves average performance such as average response time or deadline miss probability. The new algorithm is advantageous over our previous one in that it requires less amount of time to obtain the appropriate phase configuration. However, the obtained phase is not as accurate as in the previous algorithm

which identifies the phase values optimal for the specific scheduling discipline, EDF.

2 A model of processor scheduling

We consider a single processor system where n periodic tasks $\tau_1, \ldots, \tau_i, \ldots, \tau_n$ are multi-tasked. The set of periodic tasks is denoted $\tau = \{\tau_1, \ldots, \tau_i, \ldots, \tau_n\}$. Each periodic task τ_i processes a continuous media stream and is a sequence of *task instances* produced with first release time or phase r_i and period P_i. A task instance processes each data unit of the media stream. The xth instance of task τ_i, denoted τ_{ix}, is invoked at the release time $e_{ix} = r_i + P_i(x-1)$ and is required to finish before the current period expires at time $e_{ix} + P_i$[2]. The computation time of task instances varies as the input data change. We assume, however, that the average computation time C_i for each task τ_i is known a priori. The *hyperperiod* is the least common multiple of all periods $P_1, \ldots, P_n$. Since the task set τ repeats an identical execution trace every hyperperiod, only one hyperperiod is examined to analyze the entire schedule. Finally, let $U_p = \sum_{i=1}^{n} C_i/P_i$, the utilization rate of periodic task set τ.

Aperiodic tasks may arrive in the system with unknown timing. Although their complete timing information is not available a priori, their average inter-arrival and computation times are assumed to be known. Let U_{ap} designate the utilization rate of aperiodic tasks, which equals the average computation time over the average inter-arrival time.

3 The phase improvement algorithm

Conventional schedulability analysis of real-time periodic tasks often assumes the critical instant where tasks are released all at once[3]. At this zero-phased instant, workload of the periodic tasks gets maximally condensed and the system is most likely to be overloaded. However, many periodic tasks are invoked by internal timer at deterministic time points. For these tasks, we may assign artificial phases to eliminate the condensation of workload. To identify an appropriate phase vector that improves service quality, we consider the regularity of workload distribution along time. Note that the zero phase implies the highly-irregular distribution of workload and degrades system performance. The new algorithm attempts to distribute the periodic workload evenly across the time span using artificial phases. Since the ratio of released workload to the elapsed time equals U_p within a sufficiently large time span, it is desirable that the periodic workload contained in any time interval $[t_1, t_2)$ equals $U_p(t_2 - t_1)$.

Periodic workload is inflicted into system at the release times e_{ix} of the task instances τ_{ix}. Let $< v_1, \ldots, v_j, \ldots, v_m >$ be the sorted list of the release times $\{e_{ix}\}$ within a hyperperiod. That is, v_j is the jth release time and m is the number of task instances within a hyperperiod. Relating to the sorted list $< v_1, \ldots, v_m >$, function ι is defined to map the index j of release time v_j onto the index i of periodic task τ_i released at v_j, i.e., $\iota(j) = i$ if and only if

$v_j = e_{ix}$. Similarly, for a function χ, we define $\chi(j) = x$ if and only if $v_j = e_{ix}$. Note that $v_j = e_{\iota(j)\chi(j)} = r_{\iota(j)} + P_{\iota(j)}(\chi(j) - 1)$ and $< v_1, \ldots, v_j, \ldots v_m > = < e_{\iota(1)\chi(1)}, \ldots, e_{\iota(j)\chi(j)}, \ldots, e_{\iota(m)\chi(m)} >$.

The workload released within the time interval $[0, v_j)$ for $j = 1, \ldots, m$ amounts to $\sum_{l=1}^{j-1} C_{\iota(l)}$. Uniform distribution of the workload implies that, within any time interval $[0, v_j)$, the ratio of total inflicted workload $\sum_{l=1}^{j-1} C_{\iota(l)}$ to the length of the interval v_j is identical across all $j = 2, \ldots, m$. Within a sufficiently large time span, the ratio $\sum_{l=1}^{j-1} C_{\iota(l)}/v_j$ converges to U_p. It follows that any interval lengths v_j should be as close to $\sum_{l=1}^{j-1} C_{\iota(l)}/U_p$ as possible for $j = 2, \ldots, m$. Let $v_j^0 = \sum_{l=1}^{j-1} C_{\iota(l)}/U_p$, the ideal length of the interval $[0, v_j)$. We intend to identify the phase values that approximate the interval lengths v_j as close to the ideal values v_j^0 as possible.

Without loss of generality, the value of each phase r_i is assumed to fall in the interval $[0, P_i)$. Since the phases are relative among themselves, we choose $r_n = 0$ as reference point to all the other phases. Let $\boldsymbol{r}_\tau$ be the n-dimensional vector space composed of possible phase values of task set τ, that is, $\boldsymbol{r}_\tau = (r_1, \ldots, r_i, \ldots, r_{n-1}, 0)$ for real numbers r_i such that $0 \leq r_i < P_i$. Finally, let $\boldsymbol{r}$ denote a phase vector belonging to the vector space $\boldsymbol{r}_\tau$. For some task sets, the precisely-uniform workload distribution is not possible. As an example, consider the task set $\{\tau_1 = (P_1 = 10, C_1 = 2, r_1 > 0), \tau_2 = (P_2 = 20, C_2 = 10, r_2 = 0)\}$. Ideally, the length of the interval $[0, v_2) = [0, r_1)$ should equal $C_2/U_p = 14.3$. However, since $r_1 < P_1 < 14.3$, it is impossible to set the interval length r_1 to the ideal value 14.3. The ideal workload distribution is not achievable for some periodic task sets as illustrated in this example. In this case, we try to keep the length of the interval $[0, v_j)$ as close to the ideal value v_j^0 as possible. The 'error' $v_j - v_j^0$ should be minimized for all $j = 2, \ldots, m$. More precisely, the phase configuration $\boldsymbol{r}$ is identified that minimizes the total amount of the variances $(v_j - v_j^0)^2$ or

$$\sum_{j=2}^{m} (v_j - v_j^0)^2 = \sum_{j=2}^{m} (r_{\iota(j)} + P_{\iota(j)}(\chi(j) - 1) - \sum_{l=1}^{j-1} C_{\iota(l)}/U_p)^2. \qquad (1)$$

Finding such a phase vector is a well-known convex quadratic programming problem of $\boldsymbol{r}$.

To formulate the objective function as shown in Formula (1), we first need to determine the sequence of release times $< v_1, \ldots, v_m >$. The sequence of release events $\{e_{ix}\}$ varies as the value of $\boldsymbol{r}$ changes within the phase vector space $\boldsymbol{r}_\tau$ as illustrated in Figure 1a-b. To consider each of the different release sequences separately, we partition the phase vector space $\boldsymbol{r}_\tau$ into subspaces in each of which the release sequence is uniquely determined. It is clear that the sequence of first $j \leq m$ release events $< v_1, \ldots, v_j >$ is obtained in the subspace $\{\boldsymbol{r} | \boldsymbol{r} \in \boldsymbol{r}_\tau, v_1 < \ldots < v_l < \ldots < v_j\}$ where v_l is a linear expression of a phase variable $v_l = r_{\iota(l)} + P_{\iota(l)}(\chi(l) - 1)$. Figure 1a illustrates two subspaces of $\boldsymbol{r}_\tau$ corresponding to two different release sequences $< e_{11}, e_{21} >$ and $< e_{21}, e_{11} >$ respectively.

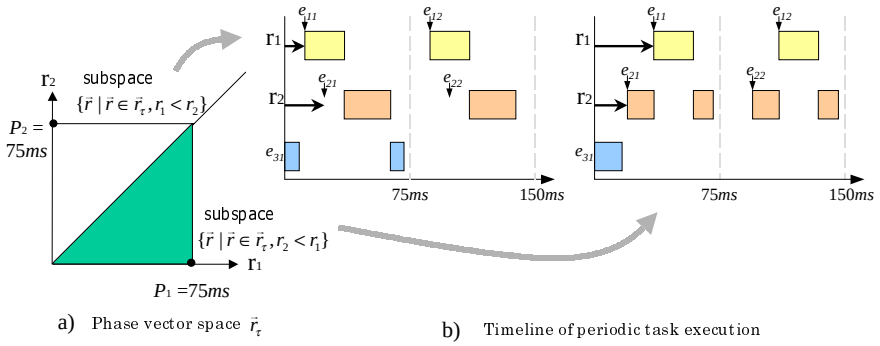


Fig. 1. Phase configuration and task release sequences illustrated for task set $\{(P_i, C_i)|(75, 21), (75, 27), (150, 16)\}$

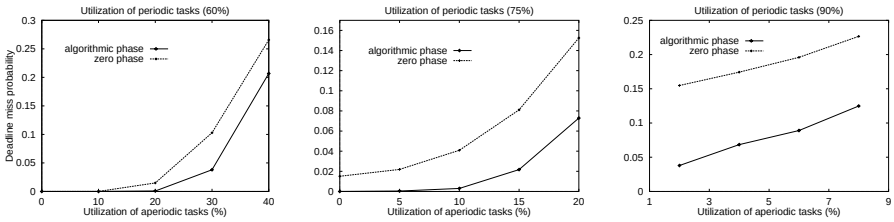
Each release sequence $< v_1, \ldots, v_m >$ achieved in the subspace $\{\boldsymbol{r}|\boldsymbol{r} \in \boldsymbol{r}_\tau, v_1 < \ldots < v_m\}$ is considered separately. [4] presents one method to generate all possible release sequences. Within each of the subspaces, the objective function $\sum_{j=2}^{m}(v_j - v_j^0)^2$ is obtained with the constraints $v_1 < \ldots < v_m$ and $0 \leq r_i < P_i$ for $i = 1, \ldots, n$. To that quadratic objective function and linear constraints, we apply the quadratic programming method to identify the phase vector $\boldsymbol{r}$ that achieves minimum $\sum_{j=2}^{m}(v_j - v_j^0)^2$ within each subspace. The best one of these local optima is then chosen to be the solution for our optimization. It should be noted that the complexity of quadratic programming in this case is $O(n^2)$ where n is the number of periodic tasks.

4 Performance evaluation

We show the impact of phased scheduling on deadline miss probability. In the experiment, periodic and aperiodic tasks are scheduled by proportional share discipline. We use the same set of periodic tasks as illustrated in Figure 1 except that their average computation times C_i are scaled up or down uniformly to achieve the desired utilization U_p. Three different periodic utilization rates are considered separately; $U_p = 60\%, 75\%$, and 90%. The aperiodic load varies across the utilization unused by periodic tasks, i.e., $0 \leq U_{ap} \leq 1 - U_p$. The weight is assigned to each task, which equals the task's utilization rate. Both periodic and aperiodic tasks are considered *time-critical*: A task instance is accepted for execution only if it can finish by deadline. Otherwise it is discarded[5]. For more detailed description on experimental settings, refer to [4].

The deadline miss probabilities under the two phase configurations are compared in Figure 2. It is shown that the phased scheduling is most effective when the system is heavily loaded with periodic tasks. Note that the task set in Figure 1 has 75×75 enumerable integer phase vectors. Our previous algorithm[1] identifies the optimal phase vector by inspecting 66 phase configurations. The

new algorithm proposed in this paper identifies the appropriate phase vector by applying quadratic programming 12 times.



(a) (b) (c)

Fig. 2. Deadline miss probability

The algorithm presented in this paper is applicable to other scheduling disciplines and objective performances as well[4]. The phased scheduling shows similar effectiveness for various scheduling disciplines and performance metrics as well.

5 Conclusion

We have discussed a new phased scheduling algorithm which distributes periodic workload uniformly across the time span. The simulation results showed that this 'time-wise' load balancing of periodic workload significantly improves service quality. We are working on a phase identification algorithm with polynomial time complexity. Also, devising more efficient algorithms based on domain specific information such as scheduling discipline and objective performance remains further study.

References

1. J. Koh and H. Shin. Phased scheduling of continuous media tasks to improve quality of service. In *IEEE International Conference on Multimedia Computing and Systems*, pages 108–117, 1998.
2. R. Steinmetz. Analyzing the multimedia operating system. *IEEE Multimedia*, 2(1):68–84, Spring 1995.
3. C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard-real environment. *Journal of the Association for Computing Machinery*, 20(1):46–61, 1973.
4. J. Koh and H. Shin. Distributing periodic workload uniformly across time to achieve better service quality. Technical Report http://cselab.snu.ac.kr/member /alcohol/rts.ps, Computer Engineering Dept., Seoul National University, 1998.
5. Karsten Schwan and Hongyi Zhou. Dynamic scheduling of hard real-time tasks and real-time threads. *IEEE Trans. on Software Engineering*, 18(8):736–748, Aug. 1992.