

Workload Functions: A New Paradigm for Real-time Computing

David Juedes, Frank Drews, Lonnie Welch
School of Electrical Engineering and Computer Science
Ohio University
Athens, Ohio 45701 U.S.A.
{juedes, drews, welch}@ohio.edu

1 Introduction

Distributed computing is being deployed in many domains to enhance survivability, scalability and flexibility. Unfortunately, the full potential of distributed computing cannot be realized in the domain of real-time because many popular system engineering approaches characterize the resource needs of tasks with a priori worst case execution times and allocate processes to computers at design time. To achieve the aforementioned capabilities, system agility is needed in order to deal with unpredictable environments, system intrusions and harsh conditions (which result in damage to computing resources). The costs of not addressing these problems include (i) limited options for handling unanticipated domain events and anomalies, (ii) inefficient use of resources, and (iii) loss of important data.

Thus, the authors present a new paradigm for real-time systems. The paradigm overcomes some of the problems with approaches that are based on the worst-case execution time paradigm. Specifically, the resource requirements of an application are represented as a workload function that maps the input size of a real-time task to an execution time for the task. A motivating example is presented to illustrate the usefulness of the new paradigm. Additionally, a new research problem in the paradigm is formally defined and preliminary results for the problem are presented. Future work to support this new paradigm is also outlined.

2 A Motivating Example

The notion of tasks which have workload-dependent execution times originated from the study of the generic air defense system [11]. The *detect* task identifies threats to a defended entity. The task runs periodically and performs the functions of filtering and evaluating radar tracks. When a threat is detected, the *detect* task triggers the *engage* task, which fires a missile at the threat. After a missile is in flight, the *guide missile* task keeps the missile on the correct course. The *guide missile* task executes periodically, uses sensor data to track the threat, recalculates the flight path for the missile, and issues guidance commands to the missile.

All three of these tasks have resource needs that are

environment-dependent. The execution time of the *detect* task is primarily workload dependent. Since the task evaluates each radar track to determine if it is a potential threat, its execution time is a function of the number of radar tracks in the environment. The workload of the *engage* task is also variable since it is activated by events which occur at rates that are determined by the external environment. Similarly, the work performed by the *guide missile* depends on the number of missiles in flight.

Unlike traditional approaches to real-time computing, which characterize the resource needs of a task by a worst case execution time (WCET) [6], we characterize the resource needs of these tasks by execution profile functions. These functions compute the resource need as a function of workload. The execution profile of the *detect* task [10] has a resource requirement of $f(r) = 0.0869r^2 + 15.4374r + 614.8615$ microseconds, where " r " is the number of radar tracks. In the terminology used in this paper, the execution profile of the *detect* task has a well-behaved (essentially concave), input dependent component $f'(r) = 0.0869r^2 + 15.4374r$ and an input independent component $f^*(r) = 614.8615$. In contrast, the workload of the *engage* task for each activation is a function of the number of missiles m in flight. The execution profile of the *engage* task is $g(m) = 12897m + 45610$ microseconds. Finally, the *guide missile* task has an execution profile that depends both on the radar tracks r and the number of missiles in flight m . The execution profile of the *guide missile* track is $h(r, m) = 0.0869r^2 + 15.4374r + 12903.909m + 46475.609$ microseconds. These profile functions are obtained by analyzing execution profiles at various workloads and then fitting data to the appropriate curve [12].

We note that concurrency is often used in order to meet the real-time constraints of the tasks. Thus, during operation, there may be many replicas of the three tasks in the air defense system. When the number of radar tracks grows too large for a single replica of the *detect* task to process all tracks within the required time bound, one or more replicas are created and the radar tracks are partitioned among them. Additionally, pipeline concurrency is obtained by replicating the subtasks of the *detect* task. In a similar manner, the *guide missile* task is replicated as necessary to meet deadlines, and its subtasks are distributed. Replication is also used for the *engage* task when heavy workloads are antici-

pated. Thus, an important problem to solve for this system is how to allocate the tasks and subtasks in a manner that allows real-time constraints to be met and that minimizes the need for reallocations (which create overhead in the system). In contrast to much of the existing work on resource allocation in real-time systems, the research described here does not assume an upper bound on the size of the input data or an upper bound on the worst-case execution time for each task. Instead, it assumes the existence, for each task T_i , of an (arbitrary) function $T_i.r(w)$ that gives an upper bound on the running time of task T_i on inputs of size w . As described in detail below, our objective is to produce an allocation of resources to tasks that remains feasible for the largest possible system workload w . This is the *maximum allowable workload* (MAW) problem, and it has several variants based on the underlying process scheduling algorithm. MAW-RMS is the version that uses fixed priority rate monotonic scheduling. MAW-EDF is the version that uses the earliest deadline first scheduling approach.

3 The Optimization Problem

Here we use the following model for real-time systems. Each real-time system consists of a collection n processors $P = \{P_1, \dots, P_n\}$ and a collection of m independent periodic tasks $T = \{T_1, \dots, T_m\}$. We are given d different sources of workload that are incurred by the environment. The corresponding input sizes are denoted by $\vec{w} = (w_1, w_2, \dots, w_d)$. Furthermore, we introduce values $c_i, i = 1, 2, \dots, d$ which express the relative importance of the i -th workload source. Each task T_i has a period $T_i.p$ and a deadline $T_i.d = T_i.p$. Furthermore, each task T_i has a collection of functions $T_i.r_j(\vec{w})$ that describe the running-times of task T_i on each processor P_j with workloads $\vec{w} = (w_1, w_2, \dots, w_d)$. It is assumed that $T_i.r_j(\vec{w})$ is a function, that is component-wise monotonically non-decreasing in each direction w_i . For shorthand, we describe a real-time system by the tuple $\langle T, P \rangle$. The maximum allowable workload problem can be formalized as follows.

Optimization Problem: (Maximum Allowable Workload) MAW-RMS/MAW-EDF

Input: $\langle T, P, \vec{c} \rangle$

Output: An allocation of tasks to processors, $alloc : T \rightarrow P$ and a workload vector \vec{w} such that the allocation $alloc$ is *feasible*. Here, we say that an allocation is feasible if, given the workload sizes $\vec{w} = (w_1, w_2, \dots, w_d)$, all tasks on a single processor meet their deadlines when scheduled using RMS [6]. Let $\mathcal{T}^j = \{T \in \{T_1, T_2, \dots, T_n\} | alloc(T) = P_j\}$, then the feasibility is observed if

$$\sum_{T \in \mathcal{T}^j} \frac{T_i.r_j(w_1, w_2, \dots, w_n)}{T_i.d} \leq |\mathcal{T}^j| \cdot (2^{|\mathcal{T}^j|} - 1). \quad (1)$$

The optimization problem MAW-EDF can be defined by replacing RMS with EDF in the above definition, and the

right-hand side in equation (1) to 1.

Objective Function: Maximize $\min_{i=1}^d \{c_i \cdot w_i\}$.

The proposed objective function aims at maximizing the minimal workload that can be handled by the system, weighted by the importance of the workload source.

Note that an optimal solution for MAW-RMS or MAW-EDF has the property that the allocation is feasible with workload (w_1, w_2, \dots, w_d) but not feasible with workload $(w_1 + 1, w_2 + 1, \dots, w_d + 1)$. Also note that it is easy to see that the optimization problem MAW-EDF is NP-hard.

4 Preliminary Results

In this chapter, we will briefly describe preliminary results from our work on the one-dimensional MAW-RMS problem, i.e., the problem with $d = 1$. We will describe theoretical results as well as experimental results.

4.1 Preliminary Analytical Results

In our initial investigation of MAW-RMS problem, we noted that in order to analyze the performance of algorithms for MAW-RMS or any other resource allocation problem using functions to describe the running-times of tasks, it is necessary to make certain assumptions about the functions that describe the running-times of the tasks in the system. In particular, it seems necessary to require that such functions be “well-behaved” in the sense that (1) their rate of increase is non-decreasing, and (2) they grow at least linearly in the size of the input. While we will not formally define the notion of “well-behaved” in this work-in-progress paper, we use this term to describe our results. It turns out that many functions of interest are well-behaved in the above sense. In particular, this set contains all polynomials with non-negative coefficients and no constant terms.

In our preliminary work, we example the MAW-RMS problem under a single workload parameter. In this special case, we were able to analyze the performance of a first-fit style (FF) algorithm based on an earlier approach by Oh and Baker [7]. The analytic result that we were able to obtain were as follows. Our adapted version of the First Fit algorithm achieves an asymptotic approximation ratio of 2.42 when all processors are identical and all of the running times of the tasks in the system are well-behaved. This means that the algorithm will produce a solution that is guaranteed to be at least $(41.32 - \epsilon)\%$ of optimal, where ϵ approaches 0 as the value of optimal solution becomes large. In particular, the algorithm produces a solution that is at least 37.5% of optimal when the value of the optimal solution is at least 100, and 40.98% of optimal when the value of the optimal solution is at least 1000. Moreover, we show that if at most 12% of the system utilization is consumed by input independent tasks (e.g., constant time tasks), then FF is guaranteed to produce a solution that is at least 33% of optimal, asymptotically. This suggests that the FF algorithm may be a reasonable approach for resource allocation in real-time systems with variable workloads.

Scenario	FF	RS	SA(FF)	HC(R)
I	180	175	190	162
II	210	198	217	187
III	128	119	131	118

Table 1. Maximum number of radar tracks for the three scenarios.

4.2 Experimental Results

In our previous work, we proposed and evaluated a collection of heuristic algorithms to both the MAW-RMS and the MAW-EDF problem, including random search, simulated annealing, first-fit/binary search, and hill climbing. We restricted ourselves to the one-dimensional MAW-RMS problem, i.e., the problem with $d = 1$. For this collection of algorithms, we performed a collection of tests. The test instances consisted of groups of $m \in \{20, 40, \dots, 100\}$ tasks running on 10 processors. Each processor was described by its profile in terms of its speed factor. The values of the speed factors for each processor were chosen from the range $[10, 30]$ at random. Each periodic task was given a period in the range $[2500, 5000]$ milliseconds, selected uniformly at random. The running-time function (in terms of the number of milliseconds) for each task was determined by generating a random pseudo-polynomial with coefficients chosen uniformly at random from the range $[0, 100]$. Random polynomial workload functions were generated randomly as follows. All terms in the pseudo polynomials came from the set $TP = \{x^2 \log x, x^2, x \log x, x\}$. The running time profile of each Task is related to a profile host with speed factor 1. Consequently, the running time of a given task was translated between machines by dividing the running time of the task in accordance with its running time profile by the speed factor of the assigned machine.

Figure 1 shows the performance of our algorithms on the collection of test data based on 100000 iterations for simulated annealing and random search. Simulated annealing and random search tend to outperform first fit when the number of applications is small. By increasing the number of iterations performed by simulated annealing and random search from 100000 to 400000, both algorithms found better solutions for the 20 and 40 application scenarios. However, since the number of iterations of simulated annealing and random search for scenarios with more than 60 applications is not increased further, first fit eventually outperforms all the heuristics presented in this paper, except SA(FF). Hill climbing turns out to be the worst allocation algorithm regardless of the scenario. All simulations were performed on a 440Mhz Sun Ultra 10 workstation

4.3 A Motivating Example Revisited

In this subsection, we examine the performance of the aforementioned heuristics on example data gathered from the motivating example of a generic air defense system

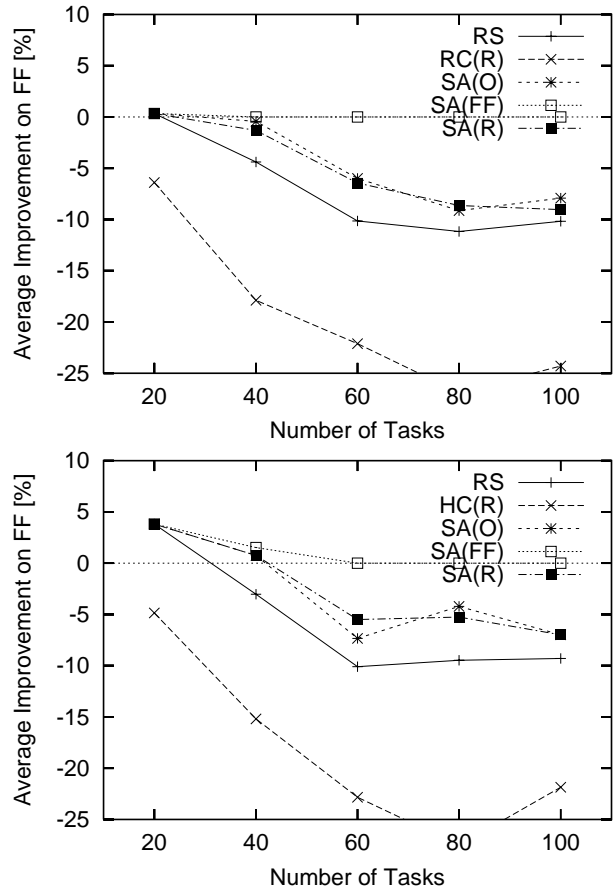


Figure 1. Average percental improvement of workload on the test collection for 100000 iterations (above) and 400000 iterations (below).

given in section 2. Recall that our example contained three basic types of tasks: the *detect* task, the *engage* task, and the *guide missile* task. As mentioned earlier, the profile functions for these three basic types of tasks depend on r , the number of radar tracks, and m , the number of missiles in flight. The profile function for the engage task was $g(m) = 12987m + 45610$ microseconds. The profile function for the detect task was $f(r) = 0.0869r^2 + 15.4374r + 614.8615$ microseconds. Finally, the profile function for the guide missile task is $h(r, m) = 0.0869r^2 + 15.4374r + 12903.909m + 46474.609$ microseconds. To match the parameters of our approach, we will assume that $r = m$, and hence that $g(m) = g(r) = 12987r + 45610$, and $h(r, m) = h(r) = 0.0869r^2 + 15.4374r + 12919.3464r + 46474.609$.

Here we examine an example real-time system where these three basic tasks are replicated. For comparison, we consider three possible scenarios on a system with 20 processors: first, when we have 20 detect tasks, 5 engage tasks, and 10 guide missile tasks; second, when we have 15 detect tasks, 2 engage tasks, and 5 guide missile tasks; and, third, when we have 30 detect tasks, 10 engage tasks, and 10 guide

missile tasks. Without loss of generality, we assume that deadline and period for the detect task is 0.1 sec, the deadline and period for the engage task is 0.01 sec, and the deadline and period for the guide missile task is 0.05 sec. The following table compares the results for the three scenarios for simulated annealing, random search, hill climbing, and first-fit.

5 Related Work

Traditional methods for achieving real-time performance do not always work well in dynamic environments because they have overwhelmingly concentrated on periodic tasks with *a priori* workloads. In early work, Abdelzaher and Shin [1, 2] examined a branch and bound algorithm for off-line scheduling of communicating tasks and messages with known arrival rates and resource requirements, where the optimization objective was the minimization of the maximum task lateness. Peng, Shin, and Abdelzaher [8] gave another branch-and-bound algorithm for allocating communicating periodic tasks with the objective of minimizing the maximum normalized system response time, called the system hazard. The Q-RAM project uses a similar objective function for optimizing resource allocations; however, it works for utility-based soft real-time applications [9, 5].

Maximizing the allowable workload produces resource allocations that are robust with respect to increases in workload. Robustness metrics have been a source of recent work [4, 3]. In particular, the recent work by Gertphol *et al.* [4] presents a robustness metric that is very close to maximum allowable workload. They define a performance metric “MAIL,” which stands for “maximum allowable increase in load” and present several heuristic algorithms for finding resource allocations that allow a maximum increase in load. Briefly, they attempt to produce an allocation of resources to tasks so that the system can absorb the largest increase in load level α in the sense that the allocation of resources remains feasible when the “load level” of each task is increased by a factor of $(1 + \alpha)$. The MAIL metric can be seen as roughly equivalent to maximum allowable workload if we measure the running-times of tasks in terms of a single load increase parameter α . Although their work is based on a more general system model that allows dependency relationships, the model used for this paper allows us to prove theoretical results for the described algorithms.

6 Conclusions and Further Work

This paper introduced and motivated the multidimensional *maximum allowable workload* (MAW) problem for real-time systems with tasks having variable workload sizes. We described some preliminary theoretical results for the problem and presented optimization results on some test scenarios for a collection of algorithms in the special case of only one workload source. A practical application of a generic air defense system was described to motivate the model assumptions and illustrate the applicability of the model. Furthermore, we presented optimization results for the air defense system. Our further work will focus on

extending our theoretical results to the multi-dimensional case. Moreover, we will develop, examine and evaluate heuristic algorithms for the multi-dimensional case. Subsequently, we will test our algorithms on various types of real-time applications.

References

- [1] T. F. Abdelzaher and K. G. Shin. Optimal combined task and message scheduling in distributed real-time systems. In *Proceedings of the 16th IEEE Real-Time Systems Symposium*, pages 162–171. IEEE Computer Society Press, 1995.
- [2] T. F. Abdelzaher and K. G. Shin. Combined task and message scheduling in distributed real-time systems. *IEEE Transactions on Parallel and Distributed Systems*, 10(11):1179–1191, 1999.
- [3] S. Ali, A. A. Maciejewski, H. J. Siegel, and J. K. Kim. Definition of a robustness metric for resource allocation. In *Proceedings of the 17th International Parallel and Distributed Processing Symposium (IPDPS 2003)*, 2003.
- [4] S. Gertphol, Y. Yu, S. B. Gundula, V. K. Prasanna, S. Ali, J. K. Kim, A. A. Maciejewski, and H. J. Siegel. A metric and mixed-integer-programming-based approach for resource allocation in dynamic real-time systems. In *Proceedings of the 16th International Parallel and Distributed Processing Symposium (IPDPS 2002)*, 2002.
- [5] C. Lee, J. Lehoczy, R. Rajkumar, and D. Siewiorek. On quality of service optimization with discrete qos options. In *Proceedings of the Fifth IEEE Real-Time Technology and Applications Symposium (RTAS '99)*, pages 276–286, Washington - Brussels - Tokyo, June 1999. IEEE.
- [6] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard real-time environment. *Journal of the Association for Computing Machinery*, 20(1):46–61, 1973.
- [7] D-I. Oh and T. P. Baker. Utilization bounds for N -processor rate monotonic scheduling with stable processor assignment. *Real Time Systems Journal*, 15(1):183–193, 1998.
- [8] D. T. Peng, K. G. Shin, and T. F. Abdelzaher. Assignment and scheduling of communicating periodic tasks in distributed real-time systems. *IEEE Transactions on Software Engineering*, 23(12), 1997.
- [9] R. Rajkumar, C. Lee, J. P. Lehoczy, and D. P. Siewiorek. A QoS-based resource allocation model. In *Proceedings of the IEEE Real-Time Systems Symposium*, 1997.
- [10] Z. Tan. Producing application CPU profiles in DynBench via curve fitting, 2003. manuscript.
- [11] Lonnie R. Welch and Behrooz A. Shirazi. A dynamic real-time benchmark for assessment of qos and resource management technology. In *Proceedings of the IEEE Real-Time Technology and Applications Symposium*, pages 36–45, 1999.
- [12] Y. Zhou. Execution time analysis for dynamic real-time systems. Master’s thesis, School of Electrical Engineering and Computer Scienc, Ohio University, 2002.